МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«**ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ
УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ**»

А.И. Солдатов, С.Н. Торгаев, И.А. Лежнина, М.Л. Громов, В.Хан, Костина М.А.

ОСНОВЫ ПРОГРАММИРОВАНИЯ НА ЯЗЫКЕ СИ

Издательство
ТУСУР
2018

УДК 681.322 (075.8)
ББК 32.973.26-04я73
Т60

Солдатов А.И.
Т60 Основы программирования на языке СИ / А.И. Солдатов, С.Н. Торгаев, И.А. Лежнина, М.Л. Громов, В.Хан, Костина М.А.; Томский государственный университет систем управления и радиоэлектроники. − Томск: Изд-во ТУСУР, 2018. – 96 с.

Пособие содержит базовый материал по основам программирования на языке С и предназначено для студентов ВУЗов, обучающихся по специальностям «Электроника и наноэлектроника», «Биотехнические системы и технологии», «Радиофизика», «Мехатроника и робототехника». Представленный в пособии теоретический материал содержит большое количество примеров программ. Также в пособии представлен курс лабораторных работ, направленный на получение практических навыков написания программного кода на языке С. Основная цель материала, изложенного в пособии – получение навыков написания программного кода с использованием базовых операторов и функций языка С для дальнейшего применения этих навыков при работе с современными микроконтроллерами и микропроцессорами.

УДК 681.322(075.8)
ББК 32.973.26-04я73

Рецензенты
кандидат технических наук, доцент кафедры медицинской и биологической кибернетики с элементами информатики СибГМУ
И.Толмачев
Кандидат технических наук,
начальник отдела информационных технологий АО «НПЦ «Полюс»
И.С.Костарев

Содержание

Введение

Язык Си является универсальным языком программирования. В связи с этим именно он используется при программировании большинства современных микроконтроллеров и микропроцессоров. В связи с этим изучение основ написания программ на данном языке программирования студентами технических специальностей является необходимым.

Данное пособие направлено на освоение принципа написания программ на языке Си и их структуры. В представленных материалах рассматриваются только основные операторы и функции языка Си: арифметические и логические операции; операторы циклов; операторы условных и безусловных переходов; функции и структуры. Знание студентами материала, представленного в пособии, достаточно для разработки программ микроконтроллеров и устройств на их основе.

В пособии представлено большое количество примеров программ на языке Си, что позволит студентам более глубоко освоить принципы разработки программ и усвоить предложенный материал.

Глава 1. Позиционные системы счисления

В позиционных системах счисления числовое значение цифры зависит от ее местоположения или позиции в последовательности цифр, изображающих число. К широко распространённым систем счисления относятся такие системы как десятичная, двоичная и шестнадцатеричная. Именно данные системы счисления находят наибольшее распространение в микропроцессорной технике.

1.1. Форматы чисел

В любой позиционной системе счисления число записывается как некоторая последовательностью цифр:

$$x = a_{n-1}a_{n-2}.....a_1a_0,$$

где, ***b*** – основание системы счисления. В данном случае говорится, что число представлено в системе счисления с основание ***b*** [1,2].

Если основание системы больше ***b***>10 (например, шестнадцатеричная система), то вводятся специальные символы, соответствующие числам более 10. Например, значение чисел ***a***, которые больше 10, в шестнадцатеричной системе будут записываться следующим образом:

$$A \to 10$$

$$B \to 11$$

$$C \to 12$$

$$D \to 13$$

$$E \to 14$$

$$F \to 15.$$

Пример 1.1: Пример записи чисел в различных системах счисления.

535_{10} – десятичная система

$0001\ 0111_2 \to 0001\ 0111b \to bx0001\ 0111$ – двоичная система

537_8 – восьмеричная система

$8CA_{16} \to 8CAh \to 0x8CA$ – шестнадцатеричная система

Положительное целое число ***x***, записанное в ***b***-ричной системе счисления, представляется в виде линейной комбинации степеней числа ***b*** [1,2]:

$$x = \sum_{k=0}^{n-1} a_k b^k,\ 0 \le a_k \le b-1.$$

Пример 1.2: Пример вычисления числа.

$539_{10} = 5 \cdot 10^2 + 3 \cdot 10^1 + 9 \cdot 10^0 = 500 + 30 + 9 = 539$

$0001\,0111_2 = 0 \cdot 2^7 + 0 \cdot 2^6 + 0 \cdot 2^5 + 1 \cdot 2^4 + 0 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 = 16 + 4 + 2 + 1 = 23$

$537_8 = 5 \cdot 8^2 + 3 \cdot 8^1 + 7 \cdot 8^0 = 320 + 24 + 7 = 351$

$8CA_{16} = 8 \cdot 16^2 + 12 \cdot 16^1 + 10 \cdot 16^0 = 2048 + 192 + 10 = 2250$

Вся микропроцессорная техника оперирует с числами, представленными в двоичных кодах. При написании программного кода пользователь, как правило, вправе использовать числа в любой системе счисления. Однако, независимо от изображения чисел и цифр в программе пользователя, микропроцессор всегда преобразует их в последовательность двоичных цифр: 0 и 1 [1,2].

В микропроцессорной технике используют следующие сокращения для обозначения форматов двоичных чисел:

- **Бит** – двоичная цифра, имеющая два значения (0, 1).
- С помощью двух бит можно представить четыре числа (00, 01, 10, 11).
- С помощью *n* бит можно представить $\mathbf{2^n}$ чисел.

Тетрада – это комбинация из четырех бит. Посредствам тетрады можно описать 16 комбинаций чисел. Любую комбинацию бит тетрады можно записать либо одной цифрой или буквой шестнадцатеричной системы счисления:

Таблица 1.1.

Запись тетрады в различных системах счисления

10-теричная	двоичная	16-теричная
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8

9	1001	9
10	1010	*A*
11	1011	*B*
12	1100	*C*
13	1101	*D*
14	1110	*E*
15	1111	*F*

- **Байт** – это группа из 8 бит. Байт позволяет описать 256 различных чисел. Биты в байте номеруются справа, налево начиная с нуля. Самое младшее число в 16-теричной системе счисления, формата байт, записывается в виде – 00_{16}, а самое старшее число – FF_{16} (рис. 1.1).

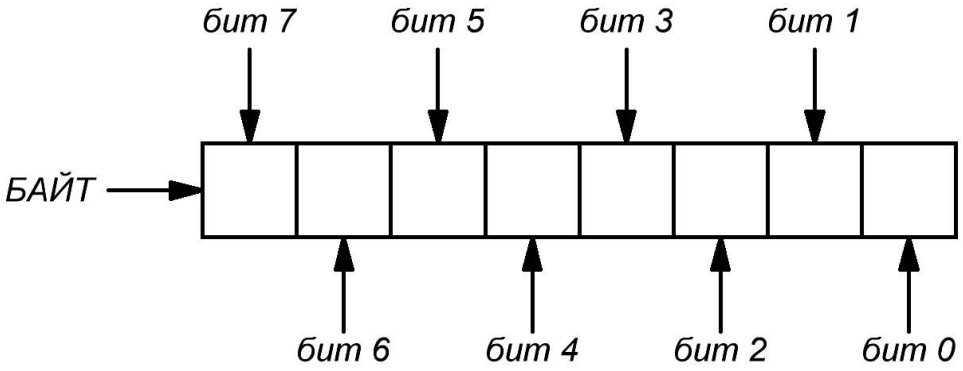


Рис. 1.1. Структура байта

- **Слово** – это комбинация из 16 или 32 бит. В микропроцессорной технике разрядность слова определяется разрядностью программируемого микропроцессора или микроконтроллера. Слово содержит:

2^{16} – 65536 комбинаций

или

2^{32} – 429496796 комбинаций.

- Для краткой записи больших степеней числа 2 величину 2^{10} обозначают буквой ***К*** (килобайт), число 2^{20} обозначают ***М*** (мегабайт), число 2^{30} обозначают ***Г*** (гигабайт).

Используя таблицу 1.2 можно достаточно быстро осуществлять перевод чисел из двоичной системы счисления в шестнадцатеричную и 17_{16} обратно.

Пример 1.3: Пример перевода чисел.

$$0001\,0111_2 \to 17_{16}$$

$$0001\,1100_2 \to 1Ch$$

$$1010\,1111_2 \to AFh$$

$$35_{16} \to 0011\,0101_2$$

$$CD_{16} \to 1100\,1101_2$$

$$F1_{16} \to 1111\,0001_2$$

1.2. Перевод десятичного числа в другую систему счисления

Для перевода целого десятичного числа ***x*** в систему счисления с основанием ***b*** необходимо последовательно делить исходное число *x* и образующиеся частные на основание ***b***. Деление необходимо выполнять до момента получения частного равного нулю. Искомое представление числа записывается как последовательность остатков от деления. При этом первый остаток дает младшую цифру искомого числа, т.е. запись остатков от деления осуществляется справа налево.

Приведём ряд примеров перевода десятичных чисел в двоичную и шестнадцатеричную системы счисления.

Пример 1.4: Перевод числа 23 в двоичную систему.

$$23/2 = 11(остаток 1)$$

$$11/2 = 5(остаток 1)$$

$$5/2 = 2(остаток 1)$$

$$2/2 = 1(остаток 0)$$

$$1/2 = 0(остаток 1)$$

$$23_{10} = 0001\,0111_2$$

Пример 1.5: Перевод числа 125 в двоичную систему.

$$125 / 2 = 62(остаток 1)$$
$$62 / 2 = 31(остаток 0)$$
$$31 / 2 = 15(остаток 1)$$
$$15 / 2 = 7(остаток 1)$$
$$7 / 2 = 3(остаток 1)$$
$$3 / 2 = 1(остаток 1)$$
$$1 / 2 = 0(остаток 1)$$
$$125_{10} = 01111101_2$$

Пример 1.6: Перевод числа 125 в шестнадцатеричную систему.

$$125 / 16 = 7(остаток 13)$$
$$7 / 16 = 0(остаток 7)$$
$$125_{10} = 7D_{16}$$

Пример 1.7: Перевод числа 2250 в шестнадцатеричную систему.

$$2250 / 16 = 140(остаток 10)$$
$$140 / 16 = 8(остаток 12)$$
$$8 / 16 = 0(остаток 8)$$
$$2250_{10} = 8CA_{16}$$

1.3. Знаковые двоичные числа

Запись знаковых двоичных чисел можно осуществлять двумя способами: прямой код и дополнительный код. На сегодняшний день, в микропроцессорной технике используется только дополнительный код записи знаковых чисел. Прямой код использовался только на ранних этапах развития микропроцессоров.

Для изображения знака числа при прямой кодировке используется старший двоичный разряд. Для отображения положительного числа старший разряд двоичного кода полагается равным 0, а для отрицательных чисел этот разряд будет равен 1. Таким образом, в прямой кодировке старший разряд используется как знаковый разряд, а оставшиеся младшие разряды используются для обозначения модуля числа.

На рис. 1.2 представлены границы знаковых и без знаковых чисел, записанных в прямом коде для 8-битного (однобайтного) числа.

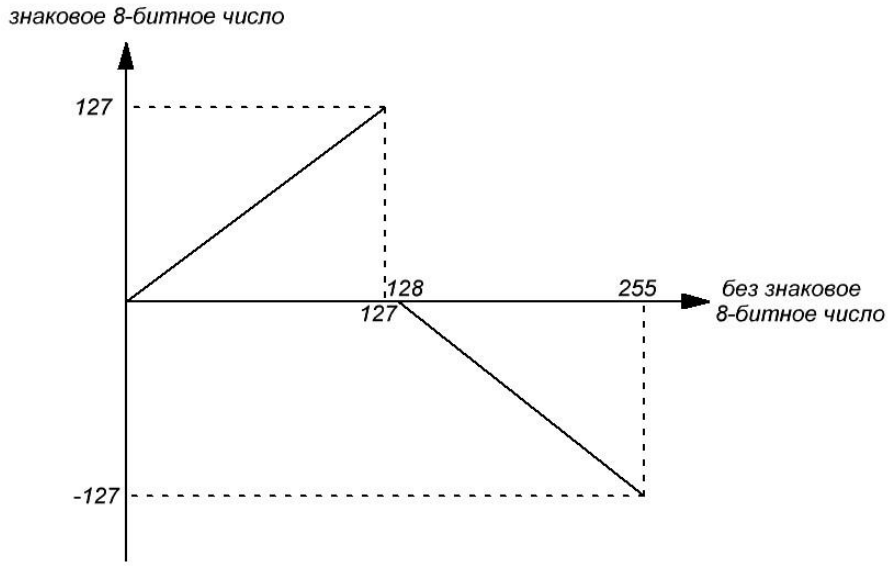


Рис. 1.2. Прямой код получения знакового числа

Максимальное число в прямой кодировке в двоичном коде записывается как $01111111=127_{10}$. Минимальное число в прямой кодировке в двоичном коде записывается как $11111111=-127_{10}$. Приведём пример записи двоичных чисел в прямом коде:

Пример 1.8: Пример записи знаковых двоичных чисел в прямом коде.

$$0001\,0111_2 = 23_{10}$$

$$1001\,0111_2 = 175_{10} - \textit{беззнаковое}$$

$$1001\,0111_2 = -23_{10} - \textit{знаковое}$$

В дополнительном коде кодировка положительных чисел совпадает с прямой кодировкой, а кодировка отрицательных чисел осуществляется по следующему алгоритму:

- записать n-битный модуль числа;
- выполнить побитовую инверсию;
- прибавить к полученному числу единицу.

На рис. 1.3 представлены границы знаковых и без знаковых чисел, записанных в дополнительном коде для 8-битного (однобайтного) числа.

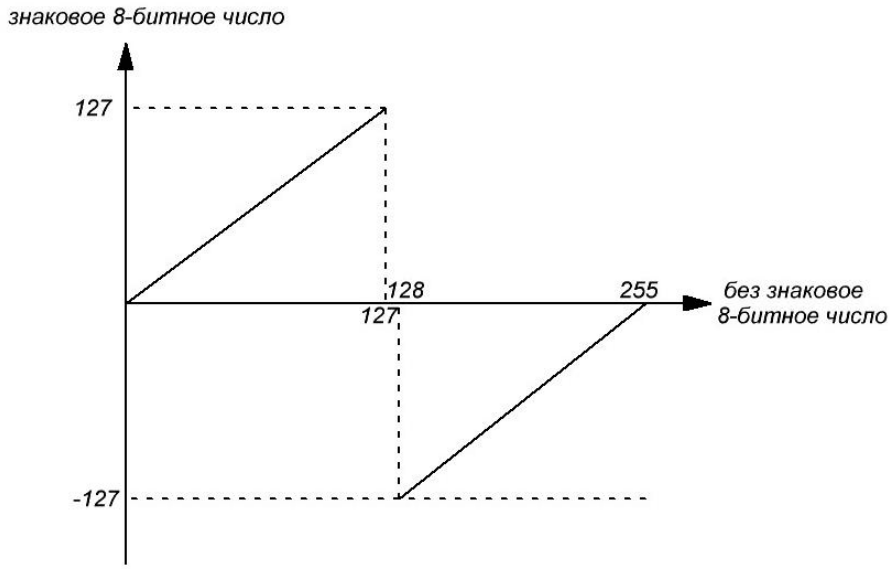


Рис. 1.3. Дополнительный код получения знакового числа

Приведем примеры записи отрицательных числе в дополнительном коде.

Пример 1.9: Пример записи числа -23 в дополнительном коде.

$0001\,0111_2 = 23_{10}$

Инверсия $-1110\,1000_2$

Прибавление $1-1110\,1001_2 = -23_{10}$

$1110\,1001_2 = 233_{10} - $ *беззнаковое*

$1110\,1001_2 = -23_{10} - $ *знаковое*

Пример 1.10: Пример записи числа -1 в дополнительном коде.

$0000\,0001_2 = 1_{10}$

Инверсия -11111110_2

Прибавление $1-11111111_2 = -1_{10}$

$11111111_2 = 255_{10} - $ *беззнаковое*

$11111111_2 = -1_{10} - $ *знаковое*

Пример 1.11: Пример записи числа -255 в дополнительном коде.

$11111111_2 = 255_{10}$

Инверсия $-0000\,0000_2$

Прибавление $1-0000\,0001_2 = -255_{10}$

$0000\,0001_2 = 1_{10} - $ *беззнаковое*

$0000\,0001_2 = -255_{10} - $ *знаковое*

1.4. Числа с фиксированной точкой

Запись дробной части числа ***x*** в позиционной системе счисления с основанием ***b*** с фиксированной точкой основывается на представлении этого числа в следующем виде [1]:

$$x = a_{-1}b^{-1} + a_{-2}b^{-2} + a_{-3}b^{-3} + ... + a_{-m}b^{-m}.$$

Для перевода дробной части чисел из десятичной системы счисления в систему счисления с основанием ***b*** необходимо использовать следующий алгоритм:

1. Дробная часть умножается на основание системы счисления ***b***, после чего отбрасывается целая часть результата умножения.
2. Полученная дробная часть снова умножается на ***b***.
3. И т.д. Процедура умножения продолжается до тех пор, пока дробная часть не станет равной нулю.
4. Далее в двоичное число после запятой выписываются целые части результатов умножения в порядке их получения (слева направо). Причем, если целая часть равно 0, то выписываем “0”, а если равна 1, то выписываем “1”.
5. Результатом может быть либо конечная, либо периодическая дробь. В случае, если дробь является периодической, необходимо обрывать умножение на каком-либо шаге. В данном случае полученное число в двоичном коде будет иметь некоторую погрешность, величина которой зависит от количества сделанных процедур умножения.

Приведем примеры записи различных чисел в двоичном коде с фиксированной точкой.

Пример 1.12: Пример записи в двоичном коде с фиксированной точкой числа 521.4375.

Перевод целой части числа:

$$521/2=260(остаток 1)$$
$$260/2=130(остаток 0)$$
$$130/2=65(остаток 0)$$
$$65/2=32(остаток 1)$$
$$32/2=16(остаток 0)$$
$$16/2=8(остаток 0)$$
$$8/2=4(остаток 0)$$
$$4/2=2(остаток 0)$$
$$2/2=1(остаток 0)$$
$$1/2=0(остаток 1)$$
$$521_{10}=10\ 0000\ 1001_2$$

Перевод дробной части числа:

$$0.4375\cdot 2=0.875 \text{ } записываем \text{ } 0$$
$$0.875\cdot 2=1.75 \text{ } записываем \text{ } 1$$
$$0.75\cdot 2=1.5 \text{ } записываем \text{ } 1$$
$$0.5\cdot 2=1 \text{ } записываем \text{ } 1$$
$$.4375_{10}=.0111_2$$

Таким образом, полная запись числа 521.4375 в двоичном коде будет иметь следующий вид:

$$0010\ 0000\ 1001.0111_2=209.7_{16}.$$

Произведем проверку результата:

$$0010\ 0000\ 1001.0111_2$$
$$x=1\cdot 2^9+1\cdot 2^3+1\cdot 2^0+0\cdot 2^{-1}+1\cdot 2^{-2}+1\cdot 2^{-3}+1\cdot 2^{-4}=$$
$$=512+8+1+0.25+0.125+0.0625=521.4375.$$

Пример 1.13: Пример записи в двоичном коде с фиксированной точкой числа 1.94.

Перевод целой части числа:

$$1/2=0(остаток 1)$$
$$1_{10}=0001_2$$

Перевод дробной части числа:

$$0.94 \cdot 2 = 1.88 \; \textit{записываем} \; 1$$
$$0.88 \cdot 2 = 1.76 \; \textit{записываем} \; 1$$
$$0.76 \cdot 2 = 1.52 \; \textit{записываем} \; 1$$
$$0.52 \cdot 2 = 1.04 \; \textit{записываем} \; 1$$
$$0.04 \cdot 2 = 0.08 \; \textit{записываем} \; 0$$
$$0.08 \cdot 2 = 0.16 \; \textit{записываем} \; 0$$
и т.д.
$$.94_{10} = .111100_2$$

Таким образом, полная запись числа 1.94 в двоичном коде будет иметь следующий вид:

$$1.94_{10} = 0001.1111_2 = 1.F_{16}.$$

Произведем проверку результата:

$$0001.1111_2$$
$$x = 1 \cdot 2^0 + 1 \cdot 2^{-1} + 1 \cdot 2^{-2} + 1 \cdot 2^{-3} + 1 \cdot 2^{-4} =$$
$$= 1 + 0.5 + 0.25 + 0.125 + 0.0625 = 1.9375.$$

1.5. Числа с плавающей точкой

Для того чтобы обрабатывать очень большие или наоборот очень маленькие числа использование двоичных кодов с фиксированной точкой является неэффективным, ввиду ограничения разрядности. Для вычислений с такими числами используются так называемые числа с плавающей точкой [1,2].

В десятичной арифметике для записи таких чисел используется алгебраическая форма. При этом число записывается в виде мантиссы, умноженной на 10 в степени, отображающей порядок числа:

$$0.2 \cdot 10^{25} \; \textit{или} \; 0.16 \cdot 10^{-30}.$$

Существует стандарт IEEE754 для представления чисел с плавающей точкой одинарной точности (***float***) и двойной точности (***double***). Для записи числа в формате с плавающей точкой одинарной точности требуется 32-битовое слово. Для записи чисел с двойной точностью требуется 64-битовое слово. Структуры чисел в формате с плавающей точкой одинарной точности и в формате с плавающей точкой двойной точности представлены на рис. 1.4.

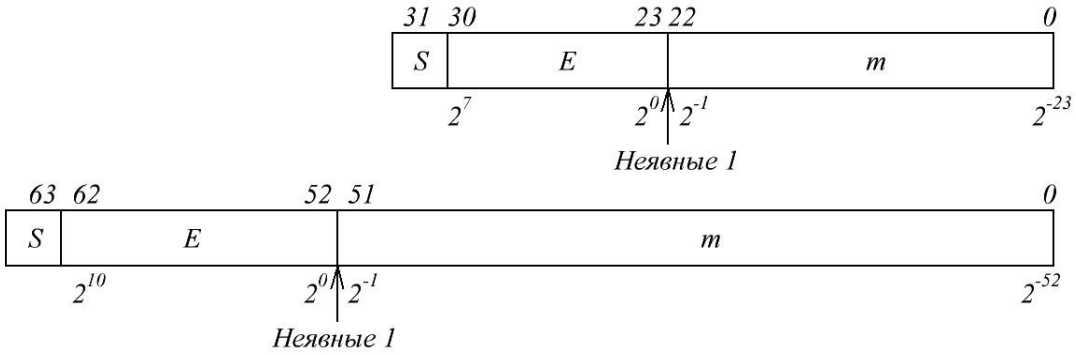


Рис. 1.4. Структура чисел с плавающей точкой одинарной и двойной точности

В структуре чисел с плавающей точкой, представленной на рис. 1.4:

- ***S*** – это знак числа: 0 - это положительное число, 1 - отрицательное число.
- ***E*** – это смещённый порядок числа. Смещение требуется, чтобы не вводить в число еще один знак. Смещённый порядок всегда положительное число. Для одинарной точности для порядка выделено восемь бит, при этом смещение принято равным 127. Для смещённого порядка двойной точности отводится 11 бит, при этом смещение принято равным 1023.
- ***m*** – это мантисса. В двоичной мантиссе после запятой могут присутствовать только цифры 1 и 0. В формате чисел с плавающей точкой принято, что мантисса всегда больше 1. То есть диапазон значений мантиссы лежит в диапазоне от 1 до 2. При этом под целую часть мантиссы не выделяется дополнительного бита – всегда подразумевается, что мантисса имеет вид: 1.***m***.

Расчет десятичного значения числа по его двоичному коду с плавающей точкой можно выполнить по формуле:

$$x = (-1)^S \cdot (1.m) \cdot 2^{E-127}.$$

Приведем примеры расчета чисел по двоичному коду с плавающей точкой.

Пример 1.14: Пример расчета значения следующего числа:

11000001 01001000 00000000 00000000

- Знаковый бит, равный 1 показывает, что число отрицательное.
- Экспонента 10000010 в десятичном виде соответствует числу 130. Вычтя число 127 из 130, получим число 3.

- Теперь запишем мантиссу: 1.100 1000 0000 0000 0000 0000.
- Смещаем точку в мантиссе на 3 знака вправо (так как 130-127 – положительное смещение).
- Таким образом, десятичное число: $1100.1_2 = 12.5_{10}$.

Также можно рассчитать значение числа с использованием соотношения, приведенного выше:

$$x = (-1)^1 \cdot (1.100\,1000) \cdot 2^{130-127} = -1.5625 \cdot 2^3 = -12.5.$$

1.6. Контрольные задания

1. Переведите в двоичную систему счисления число 31 (ответ записать в виде восьми битов, т.е. при необходимости добавить «0» к старшим битам).
2. Переведите в двоичную систему счисления число 5389 (ответ записать в виде шестнадцати бит, т.е. при необходимости добавить «0» к старшим битам).
3. Переведите в двоичную систему счисления отрицательное число -5 (ответ записать в виде восьми битов, т.е. при необходимости добавить «0» к старшим битам, использовать запись числа в дополнительном коде).
4. Переведите в двоичную систему счисления отрицательное число -100 (ответ записать в виде восьми битов, т.е. при необходимости добавить «0» к старшим битам, использовать запись числа в дополнительном коде).
5. Переведите в шестнадцатеричную систему счисления число 105 (ответ записать в виде двух символов цифрами и латинскими заглавными буквами, например, 0A, AB, 17 и т.д.).
6. Переведите в шестнадцатеричную систему счисления число 1005 (ответ записать в виде трех символов цифрами и латинскими заглавными буквами, например, 01A, A0B, 17B и т.д.).
7. Переведите в двоичную систему счисления число 4Ch, записанное в шестнадцатеричном коде (ответ записать в виде восьми битов, т.е. при необходимости добавить «0» к старшим битам).
8. Переведите в двоичную систему счисления число 5035h, записанное в шестнадцатеричном коде (ответ записать в виде шестнадцати бит, т.е. при необходимости добавить «0» к старшим битам).
9. Записать число в двоичной системе счисления с фиксированной точкой: 12.3789_{10}.

10. Записать число в двоичной системе счисления с фиксированной точкой: 2.00125_{10}.

11. Перевести двоичное число с плавающей точкой в десятичную систему счисления (2 способа): 11000011 01001000 00000000 00000000.

12. Перевести двоичное число с плавающей точкой в десятичную систему счисления (2 способа): 11000000 01101100 00000000 00000000.

Глава 2. Основы разработки алгоритмов

2.1. Основные понятия и определения

Существуют различные определения понятия алгоритма:

- **Алгоритм** – это описание последовательности действий для решения задачи или достижения поставленной цели.
- **Алгоритм** – это правила выполнения основных операций обработки данных.
- **Алгоритм** – это описание вычислений по математическим формулам.

Использование алгоритмов позволяет оценить эффективность предлагаемого способа решения поставленной задачи, результативность данного решения, исправить возможные ошибки, сравнить его еще до применения на практике с другими алгоритмами решения этой же задачи. А также, алгоритм является основой для составления программы, которую пишет программист на каком-либо языке программирования. Т.е. алгоритм не привязан к конкретному языку программирования и пользователь вправе сам выбирать с использованием каких программных средств будет решена задача.

Алгоритм должен обладать следующими свойствами:

- **Понятность для исполнителя**. Т.е. алгоритм должен быть достаточно подробным и доступным для его понимания. Необходимо, чтобы в нем отсутствовали символы и описания понятные только составителю данного алгоритма.
- **Дискретность.** Алгоритм должен представлять процесс решения задачи как последовательное выполнение простых шагов.
- **Определённость**. Каждый шаг в алгоритме должен быть четким и однозначным. Т.е. процесс выполнения алгоритма не должен требовать каких-либо дополнительных указаний или сведений о задаче.
- **Результативность**. Данное свойство состоит в том, что алгоритм должен приводить к решению задачи, либо остановки, в случае невозможности решения поставленной задачи, за конечное число шагов.
- **Массовость**. Т.е. алгоритм решения задачи разрабатывается в общем виде и не имеет привязки к средствам его реализации.

На территории РФ действует единая система программной документации (ЕСПД), частью которой является ГОСТ 19.701-90 “Схемы алгоритмов программ, данных и систем”. Данный ГОСТ

практически полностью соответствует международному стандарту ISO 5807:1985.

2.2. Основные блоки для построения алгоритмов

В данном разделе представлены основные блоки, применяемые при составлении алгоритмов.

2.2.1. Блок начала и конца работы функций

Данным блоком начинается и заканчивается любая функция. Он имеет следующий вид:

Рис. 2.1. Блок начала и конца работы функций

Тип возвращаемого значения и аргументов функции обычно указывается в комментариях к данному блоку.

2.2.2. Блок объявления переменных

Блок объявления переменных имеет следующий вид:

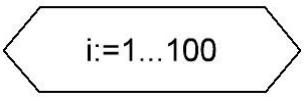


Рис. 2.2. Блок объявления переменных

В данном блоке указывается перечисление переменных, которые необходимы для выполнения поставленной задачи. В данном блоке можно как просто объявить переменные, так и указать их начальные значения.

2.2.3. Блок операций ввода и вывода данных

В ГОСТ определено много различных символов ввода/вывода данных, которые отличаются источником и приемником данных. Если источник или приемник данных не принципиален, обычно используется символ параллелограмма. В случае необходимости подробности ввода/вывода можно указать в комментариях.

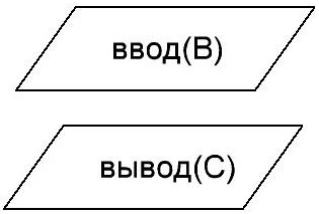


Рис. 2.3. Блок ввода/вывода данных

2.2.4. Блок выполнения операций над данными

Данный блок имеет следующий вид:

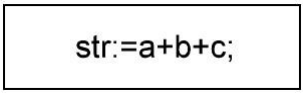


Рис. 2.4. Блок выполнения операций

В блоке операций обычно размещают одну или несколько операций, не требующих вызова внешних функций. Тут могут быть прописаны операции присвоения, копирования, арифметические и логические операции и т.д.

2.2.5. Блок ветвления алгоритма

Блок ветвления выглядит следующим образом:

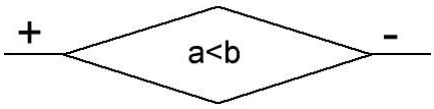


Рис. 2.5. Блок ветвления

Данный блок имеет один вход и несколько выходов. Внутри блока прописывается условие, по которому происходит переход по соответствующему выходу. В случае, если блок имеет 2 выхода на них подписывается результат сравнения – «да/нет» или «+/-». Если из блока выходит большее число линий (оператор выбора), внутри него записывается имя переменной, а на выходах значения этой переменной.

2.2.6. Блок вызова внешней процедуры/функции

Блок вызова процедуры или функции имеет вид:

Рис. 2.6. Блоки вызова функций/процедур.

Данный блок означает вызов функции, требующей перехода к подпрограмме. Если для решения поставленной задачи необходимо использовать дополнительные функции или процедуры, то необходимо приводить отдельный алгоритм данных процедур/функций.

2.2.7. Соединитель

Соединитель используется в случае, если блок-схема не умещается на лист.

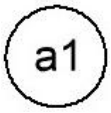


Рис. 2.7. Соединитель

Данный блок отражает переход управления между листами. Символ может использоваться и на одном листе, если по каким-либо причинам необходимо переместить часть алгоритма.

2.2.8. Блоки начала и конца цикла

Символы начала и конца цикла содержат имя и условие (рис. 2.8).

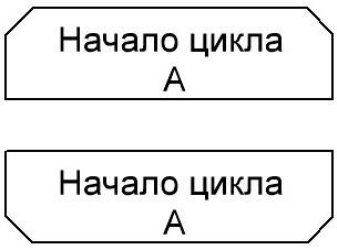


Рис. 2.8. Блоки начала и конца цикла

В данных блоках условие может отсутствовать, но только в одном из блоков. Наличие условия в блоке начала или блоке конца цикла будет определять тип условного оператора цикла, соответствующего символам на языке высокого уровня, например, ***while*** или ***do-while*** (см. ниже).

2.2.9. Блок комментариев

Комментарии при составлении алгоритмов имеют следующий вид:

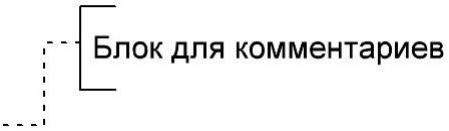


Рис. 2.9. Блок комментариев

Комментарий может быть соединен как с одним блоком, так и группой блоков. Во втором случае группа блоков выделяется на схеме пунктирной линией. При составлении алгоритмов необходимо следить за тем, чтобы комментариев было достаточно для правильного прочтения, но при этом они должны быть краткими и информативными.

2.3. Алгоритмическое описание действий, условий и циклов

2.3.1. Описание последовательности действий

Пример описания последовательности действий посредствам алгоритмических блоков представлен на рис. 2.10.

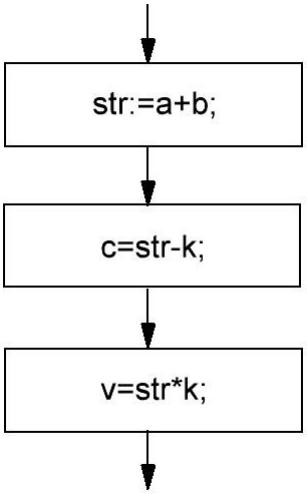


Рис. 2.10. Последовательность действий

Последовательность действий в алгоритме представляется соединением блоков с помощью стрелок. Переход от одного блока к другому определяется направлением стрелок.

2.3.2. Условие «*if*»

Блок «if» представляется с помощью блока ветвления (рис. 2.5). При переходе к данному блоку происходит проверка какого-либо

условия и если результат проверки «истина», то выполняются какие-либо действия. В ином случае ничего не происходит. Пример алгоритма условия «***if***» представлен на рис 2.11.

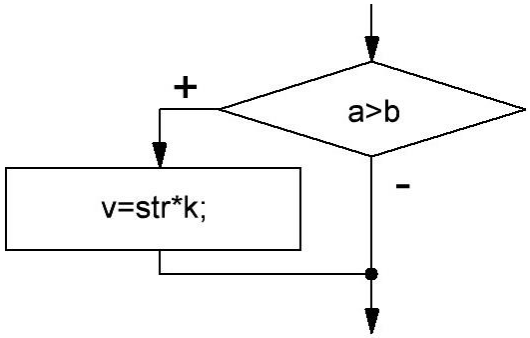


Рис. 2.11. Условие «if»

В данном примере происходит проверка условия ***a***>***b***. Если переменная ***a*** больше, чем переменная ***b***, то будет выполняться операция ***v***=***str*k***. Если же переменная ***a*** меньше либо равна переменной ***b***, то никакой операции выполняться не будет и алгоритм переходит к следующим блокам.

2.3.3. Условие «*if-else*»

Блок «***if-else***» также представляется с помощью блока ветвления (рис. 2.5). При переходе к данному блоку происходит проверка какого-либо условия и действие, которое будет выполняться в данном случае определяется тем является ли условие «истиной» или «ложью». Пример алгоритма условия «***if-else***» представлен на рис 2.12.

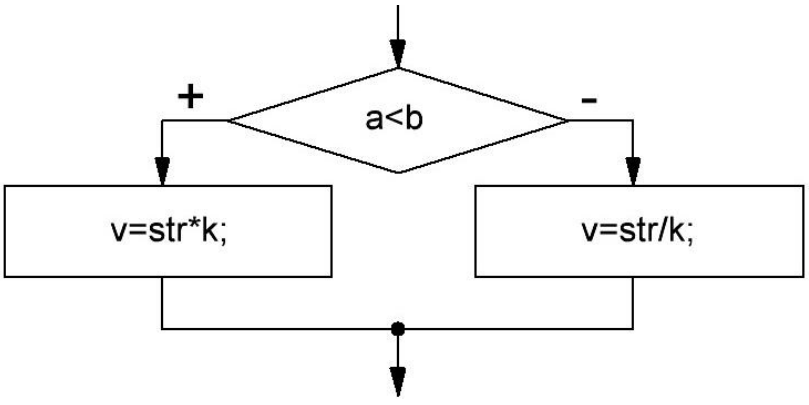


*Рис. 2.12. Условие «**if-else**»*

В данном примере происходит проверка условия ***a***<***b***. Если переменная ***a*** меньше, чем переменная ***b***, то будет выполняться операция ***v***=***str*k***. Если же переменная ***a*** больше либо равна переменной ***b***, то будет выполняться операция ***v***=***str/k***. Далее независимо от условия алгоритм переходит к следующим блокам.

2.3.4. Условие «*switch-case*»

Условие «***switch-case***» представляет собой операцию множественного выбора, при которой проверка условия может иметь более двух возможных вариантов. Действие, которое будет выполняться зависит от того какое условие имеет значение «истина». При выполнении какого-либо условия осуществляется переход к операции и далее условия больше не проверяется и алгоритм переходит к следующему блоку. Пример данного условия представлен на рис. 2.13.

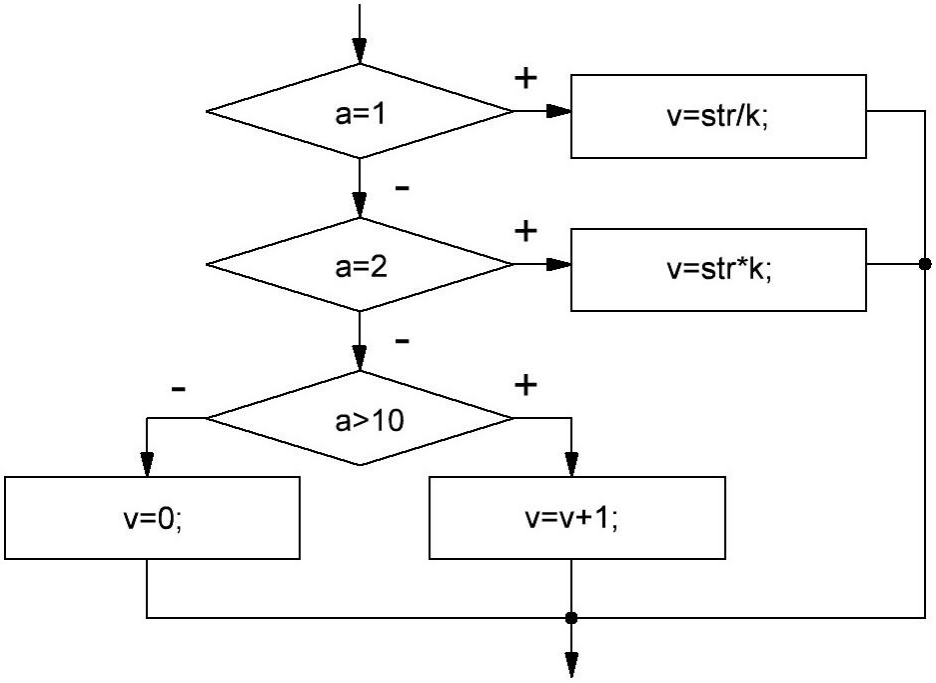


Рис. 2.13. Условие «switch…case»

При выполнении данного блока сначала осуществляется проверка ***a***=1, если условие «истина», то выполняется операция ***v***=***str/k*** и алгоритм переходит далее без проверки остальных условий. В случае если переменная ***a*** не равна 1, то проверяется условие ***a***=2. В случае выполнения данного условия выполняется действие ***v***=***str*k*** и алгоритм переходит далее без проверки остальных условий. В случае если переменная ***a*** не равна 2, то проверятся условие ***a***>10 и в зависимости от результата проверки будет выполняться либо операция ***v***=0, либо операция ***v***=***v***+1, и алгоритм переходит далее.

2.3.5. Цикл «*for*»

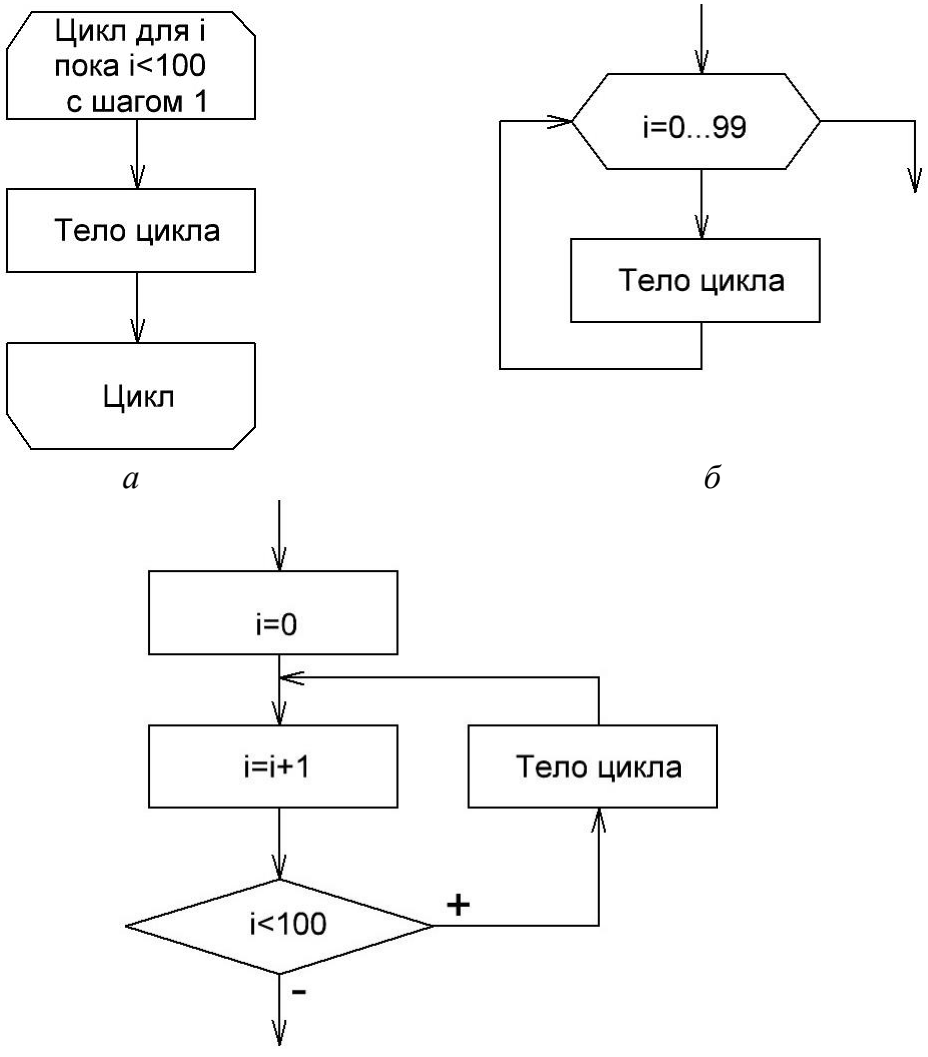
Цикл «***for***» - это цикл с заданным числом повторение. Синтаксис данного цикла в языках программирования подразумевает задание начального значения какой-либо переменной, действия которое

необходимо выполнять с данной переменной и условие для данной переменной при котором цикл заканчивается. Приведем пример:

***for*(*i*=0;*i*<100;*i*=*i*+1)**
{
тело цикла (операции)
}

В данном случае при переходе к данному циклу переменной ***i*** задается начальное значение равное 0. Тело цикла будет выполняться пока выполняется условие ***i***<100, при этом при каждом проходе цикла с переменной ***i*** будет осуществляться действие ***i***=***i***+1.

Алгоритмические представления цикла «***for***» могут быть различными. Приведем примеры алгоритмов для цикла «for», представленного выше (рис. 2.14).



а *б* *в*

Рис. 2.14. Цикл «for»

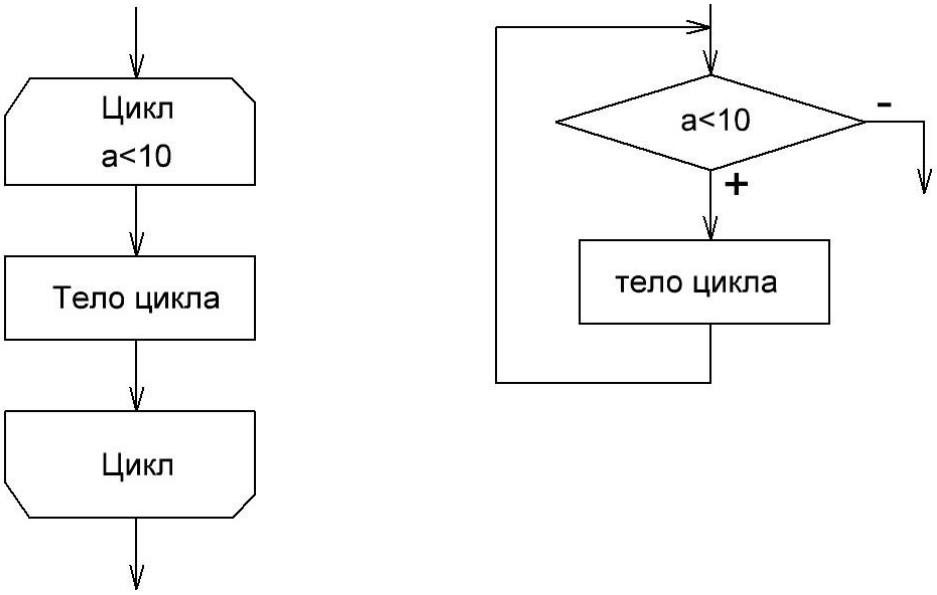
При разработке алгоритма можно использовать любой из блоков, представленных на рис. 2.14. При этом разработчик программы сам определяет каким оператором цикла он будет его реализовывать.

2.3.6. Цикл «*while*»

Цикл «*while*» – это цикл с предварительным условием. При переходе к данному циклу перед началом выполнения осуществляется проверка условия. В случае если условие не выполняется, происходит выход из цикла. Цикл с предусловием может не выполниться ни одного раза. Приведем пример цикла:

```
while(a<10)
{
тело цикла (операции)
}
```

В данном примере будет выполняться тело цикла до тех пор пока выполняется условие ***a***<10. Как и в предыдущем случаем могут быть разные варианты представления данного цикла в алгоритме (рис. 2.15).



а *б*

Рис. 2.15. Цикл «while»

При разработке алгоритма можно использовать любой из блоков, представленных на рис. 2.15.

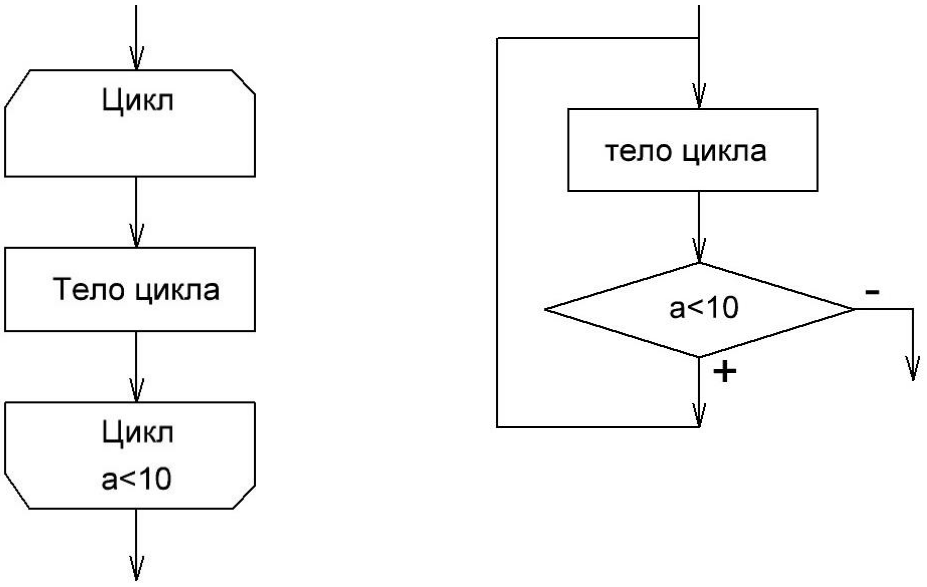
2.3.7. Цикл «*do-while*»

Цикл «***do-while***» – это цикл с постусловием. При переходе к данному циклу сначала выполняется операция (тело цикла) и только

потом проверяется условие. В случае если условие не выполняется, происходит выход из цикла. Таким образом, данный цикл выполняется как минимум один раз, в отличие от цикла «***while***». Приведем пример цикла:

```
do
{
тело цикла (операции)
}
while(a<10)
```

В данном примере будет выполняться тело цикла и после проверяется условие ***a***<10. На рис. 2.16 показаны варианты представления данного цикла в алгоритме.



а *б*

Рис. 2.16. Цикл «do-while»

При разработке алгоритма можно использовать любой из блоков, представленных на рис. 2.16.

2.4. Примеры алгоритмов

Пример 2.1. Алгоритм подсчета суммы двух чисел (***A*** и ***B***), введенных с клавиатуры, с выводом на экран результата (***C***). Алгоритм представлена на рис. 2.17.

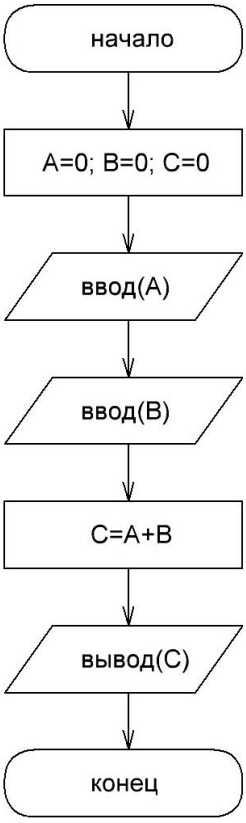


Рис. 2.17. Алгоритм подсчета суммы двух чисел

Пример 2.2. Алгоритм сравнения двух чисел (***A*** и ***B***), введенных с клавиатуры, с выводом на экран большего числа. В случае равенства выводить знак “=” (рис. 2.18).

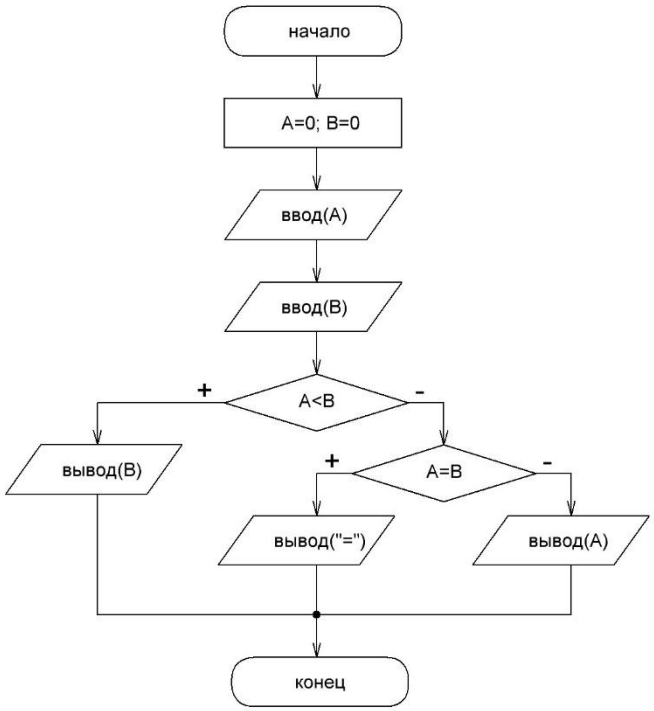


Рис. 2.18. Алгоритм сравнения двух чисел

Пример 2.3. Алгоритм подсчета суммы элементов массива ***A***[10] с размерностью 10 с выводом на экран результата ***B***. Три варианта алгоритма представлена на рис. 2.19-2.21.

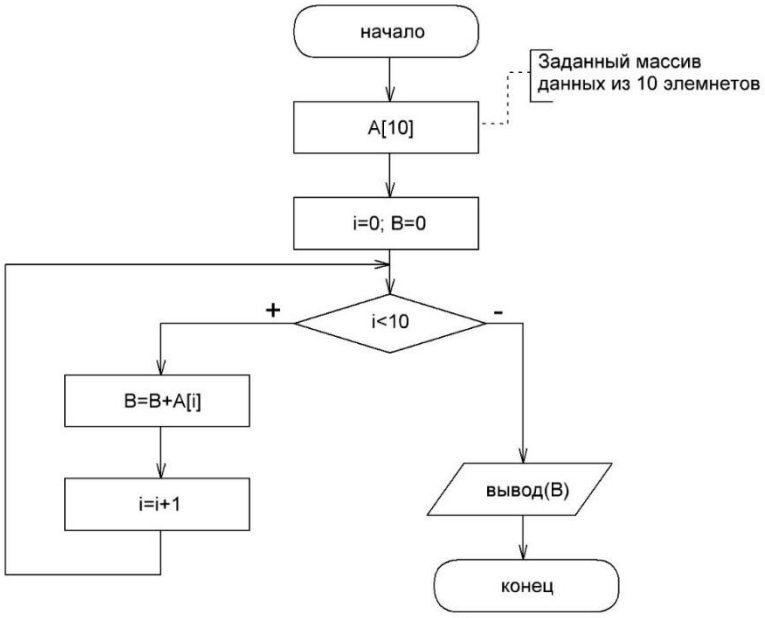


Рис. 2.19. Алгоритм подсчета суммы элементов массива с помощью функции «if-else»

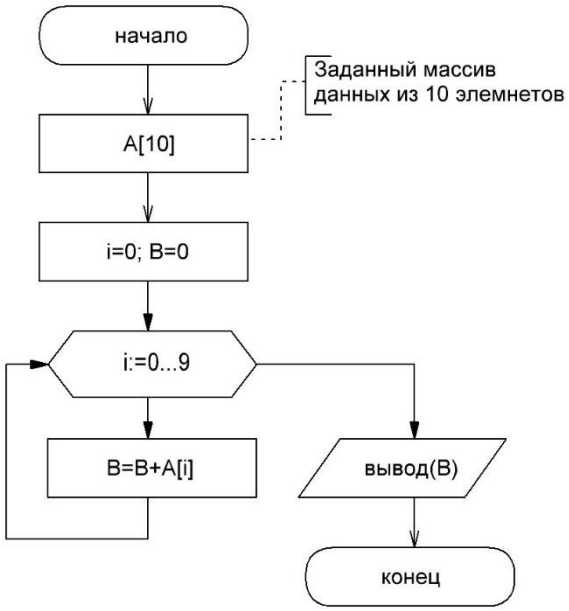


Рис. 2.20. Алгоритм подсчета суммы элементов массива с помощью функции «for»

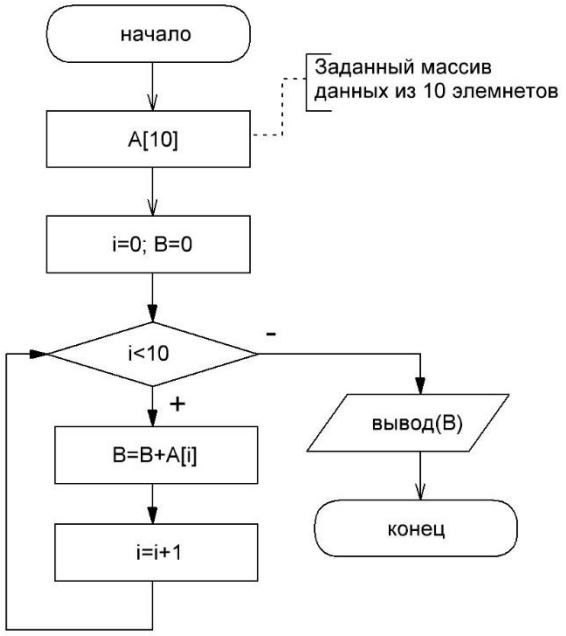


Рис. 2.21. Алгоритм подсчета суммы элементов массива с помощью функции «while»

Пример 2.4. Алгоритм вычисления площади квадрата (если нажата кнопка "***s***") или объема куба (если нажата кнопка "***v***") с использованием внешних функций и выводом результата на экран (рис.

2.22-2.24). Ввод величины грани квадрата/куба осуществляется с клавиатуры.

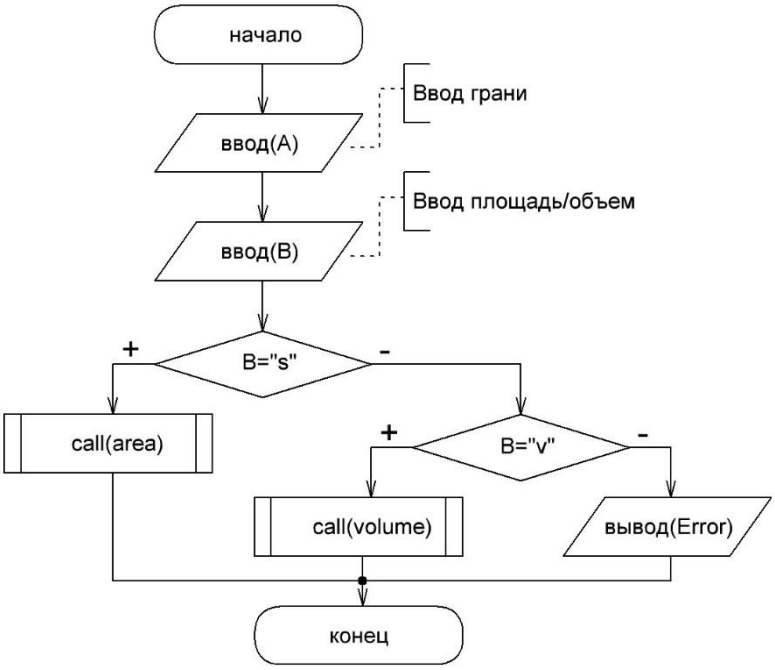


Рис. 2.22. Алгоритм вычисления площади квадрата

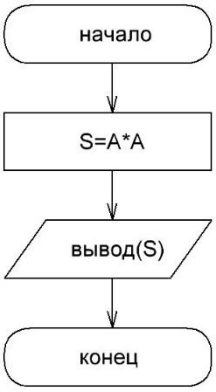


Рис. 2.23. Подпрограмма «area»

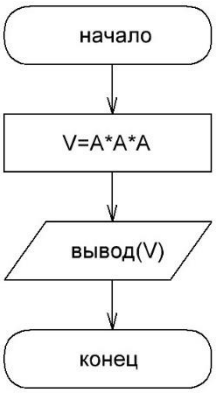


Рис. 2.24. Подпрограмма «volume»

2.5. Контрольные задания

1. Разработать алгоритм вычисления площади и периметра квадрата (если нажата кнопка "***s***" – площадь, если нажата кнопка "***p***" – периметр) с использованием функций и выводом результата на экран. Ввод величины грани квадрата осуществляется с клавиатуры.
2. Разработать алгоритм вывода на экран максимального числа (из чисел, введенных с клавиатуры). С клавиатуры вводятся десять чисел и далее выводится максимальное.
3. Разработать алгоритм вывода на экран всех чисел от 0 до 100, которые делятся на 9.
4. Разработать алгоритм вычисления суммы и произведения двух числе, введенных с клавиатуры.
5. Разработать алгоритм вычисления суммы всех чисел, введенных с клавиатуры. С клавиатуры вводится произвольное количество чисел, а результат их суммы выводится по кнопке «***S***».
6. Разработать алгоритм вывода на экран всех чисел от 100 до 1000, которые делятся на 9 и на 5.

Глава 3. Основы языка программирования Си

3.1. Элементы языка Си

Множество символов, используемых в Си можно разделить следующим образом [3-5]:

1. символы, используемые для образования ключевых слов (зарегистрированные операторы и команды) и идентификаторов (переменные). Входят прописные и строчные буквы латинского алфавита, символ подчеркивания;
2. группа прописных и строчных букв русского алфавита и арабские цифры (1,2,3…);
3. знаки нумерации и специальные символы. Используются для организации вычислений, а также для передачи компилятору инструкций;
4. управляющие и разделительные символы: пробел, табуляция, перевод строки, возврат каретки и т.д. Используется для разделения объектов;
5. Управляющие последовательности. Чаще используются в функциях ввода/вывода, например, ***\n*** – переход на новую строку, ***\t*** – горизонтальная табуляция и др.

3.2. Основные понятия языка Си

Программа, написанная на языке Си, состоит из операторов. Каждый оператор вызывает выполнение некоторых действий на соответствующем шаге выполнения программы.

При написании операторов применяются латинские прописные и строчные буквы, цифры и специальные знаки. К таким знакам, например, относятся: точка (.), запятая (,), двоеточие (:), точка с запятой (;) и др. Совокупность символов, используемых в языке, называется ***алфавитом языка***. В персональном компьютере символы хранятся в виде кодов. Соответствие между каждым символом и его кодом задается специальной кодовой таблицей. На нее разработан стандарт ASCII, поэтому коды символов называют ASCII-кодами.

Различают видимые и управляющие символы. Первые могут быть отображены на экране дисплея. Вторые вызывают определенные действия в машине, например, звуковой сигнал - код 710, возврат курсора на один шаг - код 810, горизонтальная табуляция - код 910, перевод курсора на новую строку - код 1010, перемещение курсора в

начало строки - код 1310 и т.д. Такие управляющие символы имеют десятичные номера 0 - 31, 127.

Для представления каждого символа в персональном компьютере используется один байт, поэтому общее число символов равно $2^8 = 256$. В таблице 3.1 представлены ASCII коды основных символов (***DEC***-десятичная система счисления, ***HEX*** – шестнадцатеричная система счисления, ***BIN*** – двоичная система счисления, ***CHAR*** – символьный вид) [3-5].

Таблица 3.1.

Таблица ASCII кодов

Dec	Hex	Char	Cmd	Dec	Hex	Char	Cmd	Dec	Hex	Char	Cmd	Dec	Hex	Char	Cmd
0	00		NUL	32	20		(sp)	64	40	@		96	60	`	
1	01	☺	SOH	33	21	!		65	41	A		97	61	a	
2	02	☻	STX	34	22	"		66	42	B		98	62	b	
3	03	♥	ETX	35	23	#		67	43	C		99	63	c	
4	04	♦	EOT	36	24	$		68	44	D		100	64	d	
5	05	♣	ENQ	37	25	%		69	45	E		101	65	e	
6	06	♠	ACK	38	26	&		70	46	F		102	66	f	
7	07	•	BEL	39	27	'		71	47	G		103	67	g	
8	08	◘	BS	40	28	(		72	48	H		104	68	h	
9	09	○	TAB	41	29	)		73	49	I		105	69	i	
10	0A	◙	LF	42	2A	*		74	4A	J		106	6A	j	
11	0B	♂	VT	43	2B	+		75	4B	K		107	6B	k	
12	0C	♀	FF	44	2C	,		76	4C	L		108	6C	l	
13	0D	♪	CR	45	2D	-		77	4D	M		109	6D	m	
14	0E	♫	SO	46	2E	.		78	4E	N		110	6E	n	
15	0F	☼	SI	47	2F	/		79	4F	O		111	6F	o	
16	10	►	DLE	48	30	0		80	50	P		112	70	p	
17	11	◄	DC1	49	31	1		81	51	Q		113	71	q	
18	12	↕	DC2	50	32	2		82	52	R		114	72	r	
19	13	‼	DC3	51	33	3		83	53	S		115	73	s	
20	14	¶	DC4	52	34	4		84	54	T		116	74	t	
21	15	§	NAK	53	35	5		85	55	U		117	75	u	
22	16	▬	SYN	54	36	6		86	56	V		118	76	v	
23	17	↨	ETB	55	37	7		87	57	W		119	77	w	
24	18	↑	CAN	56	38	8		88	58	X		120	78	x	
25	19	↓	EM	57	39	9		89	59	Y		121	79	y	
26	1A	→	SUB	58	3A	:		90	5A	Z		122	7A	z	
27	1B	←	ESC	59	3B	;		91	5B	[		123	7B	{	
28	1C	∟	FS	60	3C	<		92	5C	\		124	7C	|	
29	1D	↔	GS	61	3D	=		93	5D	]		125	7D	}	
30	1E	▲	RS	62	3E	>		94	5E	^		126	7E	~	
31	1F	▼	US	63	3F	?		95	5F	_		127	7F	⌂	DEL

Важным понятием языка является ***идентификатор***, который используется в качестве имени объекта (функции, переменной, константы и др.). Идентификаторы должны выбираться с учетом следующих правил:

- идентификаторы должны начинаться с буквы латинского алфавита (***a***,...,***z***, ***A***,...,***Z***) или с символа подчеркивания (_);

- в идентификаторах могут использоваться буквы латинского алфавита, символ подчеркивания и цифры (от 0 до 9). Использование других символов в идентификаторах запрещено;
- в языке Си буквы нижнего регистра (***a***,...,***z***), применяемые в идентификаторах, отличаются от букв верхнего регистра (***A***,...,***Z***). Это означает, что следующие идентификаторы считаются различными: ***function***, ***FUNCTION***, ***Function*** и т.д;
- идентификаторы объектов не должны совпадать с ключевыми словами языка и именами стандартных функций из применяемых библиотек.

Пробелы, символы табуляции и перехода на новую строку в программах на языке Си игнорируются. Это позволяет записывать различные выражения в удобном для прочтения и понимания виде. Кроме того, строки программы можно начинать с любой позиции, что дает возможность выделять в тексте группы операторов.

3.3. Структура программы в языке Си

Программа на языке Си всегда состоит из функций и переменных. Функции содержат набор инструкций, описывающих вычисления, которые необходимо выполнить. А переменные хранят значения, которые используются в процессе вычислений [3-5].

Обычно разработчик может определять любые имена для своих функций. Однако в языке Си есть особая функция с именем "***main***". Любая программа начинается с первой инструкции функции ***main***. Т.е. данная функция должна присутствовать в любой программе. Не допускается вызов функции ***main*** из других функций программы – она является управляющей.

Следующие за именем функции круглые скобки предназначены для указания параметров (или аргументов), которые передаются в функцию при обращении к ней, т.е. которые необходимы для выполнения инструкций внутри функции. В случае записи ***main*()** в функцию не передается никаких аргументов, поэтому список аргументов в круглых скобках пустой. Функцию ***main*()** можно записать двумя способами:

***int main*()**
***void main*()**

Перед именем функции указывается тип возвращаемого значения. При обращении к главной функции значение возвращается операционной системе. Последняя запись (***void main*()**) не будет возвращать никакого значения. В свою очередь первая запись ***int main*()**

сообщает компилятору о том, что после выполнения функции будет возвращено целочисленное значение, которое необходимо операционной системе и сообщает ей о том, что программа завершилась корректно. Если же это значение не возвращено, то операционная система понимает, что программа завершилась в аварийном режиме.

Для возврата целочисленного значения перед завершением функции добавляется строка:

***return* 0;**

В зависимости от программы, достаточно часто данная строка не требуется.

Язык Си является блочно-структурированным. Каждый блок заключается в фигурные скобки **{}**.

Каждое действие в языке Си заканчивается символом "точка с запятой" - **;**. В качестве действия может выступать вызов функции или осуществление некоторых операций.

В Си для написания комментариев используются следующие символы: **/*** - начало комментария, ***/** - конец комментария. При этом все что будет заключено между этими символами будет являться комментариями, т.е. не будет восприниматься компилятором как инструкция или набор инструкций. Такое ограничение области комментариев удобно использовать при комментировании блока программы, состоящего из нескольких строк. При необходимости комментария одной строки удобно использовать символ //, при этом комментарием будет являться все, что расположено между символом // и концом строки (т.е. все, что расположено справа от данного символа). Примеры вариантов комментариев представлены в табл. 3.2 [3-5].

Таблица 3.2.

Примеры написания комментариев в Си

Обозначение комментария двух строк.	int a; /* целая переменная */

Обозначение комментария блока программы При этом в ходе выполнения программы будут выполнены операции ***b*=*a*** и ***d*=*a*-*b***. А то что заключено между символами **/*** и ***/** выполняться не будет.	int a,b,c,d;
	float e;
	
	void main(void)
	{
	b=a;
	/*c=d
	e=a+b;*/
	d=a-b;

	}

Комментирование одной строки. При этом, то что находится правее символа // будет являться комментарием. Т.е. в данной программе фраза «определение переменных» является комментарием, а также в ходе выполнение программы операция ***e*=*a*+*b*** выполняться не будет, так как она закомментирована.	int a,b,c; //определение переменных
	float e;
	
	void main(void)
	{
	b=a;
	//e=a+b;
	}

Обычно ***main*** для выполнения своей работы пользуется услугами других функций; одни из них пишутся самим программистом, а другие берутся готовыми из имеющихся библиотек. Например, строка программы:

#include <stdio.h>

сообщает компилятору, что он должен включить стандартную библиотеку ввода-вывода. Эта строка встречается в начале многих файлов программ на языке Си. Существует большое количество различных библиотек, которые выполняют различные функции. В таблице 3.3 представлены примеры стандартных библиотек языка Си.

Таблица 3.3.

Примеры стандартных библиотек языка Си

<assert.h>	Макрос assert()
<ctype.h>	Обработка символов
<errno.h>	Сообщения об ошибках
<float.h>	Значения с плавающей точкой, зависящие от конкретной реализации компилятора
<iso646.h>	Макросы, соответствующие различным операторам, например && и ^. Добавлено в 1995 году (Поправка 1)
<limits.h>	Различные ограничения, зависящие от конкретной реализации компилятора
<locale.h>	Функция setlocale()
<math.h>	Различные определения, используемые библиотекой math
<setjmp.h>	Нелокальные переходы
<signal.h>	Значения сигналов
<stdarg.h>	Списки аргументов переменной длины

<stddef.h>	Распространенные константы
<stdio.h>	Ввод-вывод файлов
<stdlib.h>	Смешанные объявления
<string.h>	Функции обработки строк
<time.h>	Функции системного времени и даты

Описание и функции данных библиотек легко найти как в различных литературных источниках.

3.4. Типы данных языка Си

Программы в языке Си оперируют с различными данными, которые могут быть простыми и структурированными. Простые данные - это целые и вещественные числа, символы и указатели (адреса объектов в памяти). Целые числа не имеют, а вещественные имеют дробную часть. Структурированные данные - это массивы и структуры.

В языке различают понятия "***тип данных***" и "***модификатор типа***". Тип данных - это, например, целый, а модификатор - со знаком или без знака. Целое со знаком будет иметь как положительные, так и отрицательные значения, а целое без знака - только положительные значения. В языке Си можно выделить пять базовых типов, которые задаются следующими ключевыми словами:

- ***char*** **–** символьный;
- ***int*** **–** целый;
- ***float*** **–** вещественный;
- ***double*** **–** вещественный двойной точности;
- ***void*** **–** не имеющий значения.

В таблице 3.4 представлена краткая характеристика типов данных языка Си.

Таблица 3.4.

Краткая характеристика типов данных языка Си

Тип данных	Характеристика
char	Переменная типа ***char*** имеет размер 1 байт, ее значениями являются различные символы из кодовой таблицы, например, '**ф**', '**:**', '***j***' (при записи в программе они заключаются в одинарные кавычки).
int	Размер переменной типа ***int*** в стандарте языка Си не определен. В большинстве систем программирования размер переменной типа ***int*** соответствует размеру целого машинного слова. Например, в компиляторах

	для 16-разрядных процессоров переменная типа ***int*** имеет размер 2 байта. В этом случае знаковые значения этой переменной могут лежать в диапазоне от -32768 до 32767.
float	Ключевое слово ***float*** позволяет определить переменные вещественного типа, т.е. перменные имеющие дробную часть, например, -5.6, 31.28 и т.п. Вещественные числа могут быть записаны также в форме с плавающей точкой, например, -1.09e+4. Переменная типа ***float*** занимает в памяти 32 бита. Она может принимать значения в диапазоне от 3.4e-38 до 3.4e+38.
double	Ключевое слово ***double*** позволяет определить вещественную переменную двойной точности. Она занимает в памяти 64 бита. Переменная типа ***double*** может принимать значения в диапазоне от 1.7e-308 до 1.7e+308.
void	Ключевое слово ***void*** (не имеющий значения) используется для нейтрализации значения объекта, например, для объявления функции, не возвращающей никаких значений.

Объект некоторого типа может быть модифицирован с помощью специальных модификаторов. В языке Си имеются следующие модификаторы: ***unsigned***, ***signed***, ***short***, ***long***.

Модификаторы записываются перед спецификаторами типа**.** Если после модификатора опущен спецификатор, то по умолчанию, спецификатором является ***int.*** Таким образом, следующие строки языка Си являются идентичными:

***long a*;**
***long int a*;**

В таблице 3.5 представлены возможные сочетания модификаторов со спецификаторами, а также размер и диапазон значений объекта (для 16-разрядных компиляторов).

Таблица 3.5.

Возможные сочетания модификаторов со спецификаторами в Си

Тип	Размер в байтах (битах)	Интервал изменения
char	*1(8)*	от -128 до 127

unsigned char	*1(8)*	от 0 до 255
signed char	*1(8)*	от -128 до 127
int	*2(16)*	от -32768 до 32767
unsigned int	*2(16)*	от 0 до 65535
signed int	*2(16)*	от -32768 до 32767
short int	*2(16)*	от -32768 до 32767
unsigned short int	*2(16)*	от 0 до 65535
signed short int	*2(16)*	от -32768 до 32767
long int	*4(32)*	от -2147483648 до 2147483647
unsigned long int	*4(32)*	от 0 до 4294967295
signed long int	*4(32)*	от -2147483648 до 2147483647
float	*4(32)*	от 3.4E-38 до 3.4E+38
double	*8(64)*	от 1.7E-308 до 1.7E+308
long double	*10(80)*	от 3.4E-4932 до 3.4E+4932

3.5. Переменные и константы языка Си

Все ***переменные*** до их использования должны быть определены (объявлены). При этом задается тип, а затем идет список из одной или более переменных этого типа, разделенных запятыми.

```
int a, b, c;
char x, y;
```

В языке различают понятия объявления переменной и ее определения. Объявление устанавливает свойства объекта: его тип (например, целый), размер (например, 4 байта) и т.д. Определение наряду с этим вызывает выделение памяти (в приведенном примере дано определение переменных). Переменные можно разделять по строкам произвольным образом:

```
float a;float b;
int a, b;
int c;
char x, y;
```

Переменные в языке Си могут быть инициализированы при их определении:

```
int a = 25, h = 6;
char g = 'Q', k = 'm';
float r = 1.89;
long double n = r*123;
```

В языке возможны глобальные и локальные объекты. Первые определяются вне функций и, следовательно, доступны для любой из

них. Локальные объекты по отношению к функциям являются внутренними. Они начинают существовать, при входе в функцию и уничтожаются после выхода из нее. Ниже показана структура программы на Си и возможные места в программе, где определяются глобальные и локальные объекты:

```
int a;                                  //определение глобальной переменной

int function (int b, char c);           //объявление функции

void main (void)
{                                       //тело программы
        int d, e;                       //определение локальных переменных
        float f;                        //определение локальной переменной
}

int function (int b, char c)            /*определение функции и формальных
                                        параметров (по существу - локальных
                                        переменных) b и c*/
{                                       //тело функции
        char g;                         //определение локальной переменной
}
```

Отметим, что выполнение программы всегда начинается с вызова функции ***main*()**, которая содержит тело программы. Тело программы, как и тело любой другой функции, помещается между открывающей и закрывающей фигурными скобками – **{ }**. В языке Си все определения должны следовать перед операторами, составляющими тело функции. Если они сделаны в функции, то соответствующие объекты будут локальными, а если вне функций, то глобальными.

Наряду с переменными в языке существуют следующие виды ***констант***:

- вещественные, например, 123.456, 5.61e-4. Они могут снабжаться суффиксом ***F*** (или ***f***), например, 123.456F, 5.61e-4f;
- целые, например, 125;
- короткие целые, в конце записи которых добавляется буква (суффикс) ***H*** (или ***h***), например, 275h, 344H;
- длинные целые, в конце записи которых добавляется буква (суффикс) ***L*** (или ***l***), например, 361327L;
- беззнаковые, в конце записи которых добавляется буква ***U*** (или ***u***), например, 62125U;

- восьмеричные, в которых перед первой значащей цифрой записывается нуль (0), например, 071;
- шестнадцатеричные, в которых перед первой значащей цифрой записывается пара символов нуль-икс (0x), например, 0x5F;
- символьные - единственный символ, заключенный в одинарные кавычки, например, 'O', '2', '.' и т.п
- строковые - последовательность из нуля символов и более, заключенная в двойные кавычки, например, "Это строковая константа". Кавычки не входят в строку, а лишь ограничивают ее. Строка представляет собой массив из перечисленных элементов, в конце которого помещается байт с символом '\0'. Таким образом, число байтов, необходимых для хранения строки, на единицу превышает число символов между двойными кавычками;
- константное выражение, состоящее из одних констант, которое вычисляется во время трансляции (например, a=60+301);
- типа ***long double***, в конце записи которых добавляется буква ***L*** (или ***l***), например, 1234567.89L.

Объявить константу можно двумя способами:

1. Используется ключевое слово ***const*** **[*type*] <идентификатор>=<значение>;**

   const int a*=2;** ⇔ ***const a*=2;** //int*** задается по умолчанию.
2. ***#define*** **<идентификатор>пробел<значение>**

   ***#define min* 60;**

3.6. Преобразование типов данных

Если в выражении появляются операнды различных типов, то они преобразуются к некоторому общему типу, при этом к каждому арифметическому операнду применяется следующая последовательность правил:

1. Если один из операндов в выражении имеет тип ***long double***, то остальные тоже преобразуются к типу ***long double***.
2. В противном случае, если один из операндов в выражении имеет тип ***double***, то остальные тоже преобразуются к типу ***double***.
3. В противном случае, если один из операндов в выражении имеет тип ***float***, то остальные тоже преобразуются к типу ***float***.
4. В противном случае, если один из операндов в выражении имеет тип ***unsigned long***, то остальные тоже преобразуются к типу ***unsigned long***.
5. В противном случае, если один из операндов в выражении имеет тип ***long***, то остальные тоже преобразуются к типу ***long***.

6. В противном случае, если один из операндов в выражении имеет тип ***unsigned***, то остальные тоже преобразуются. к типу ***unsigned***.

7. В противном случае все операнды преобразуются к типу ***int***. При этом тип ***char*** преобразуется в ***int*** со знаком; тип ***unsigned char*** в ***int***, у которого старший байт всегда нулевой; тип ***signed char*** в ***int***, у которого в знаковый разряд передается знак из ***char***; тип ***short*** в ***int***.

Предположим, что вычислено значение некоторого выражения в правой части оператора присваивания. В левой части оператора присваивания записана некоторая переменная, причем ее тип отличается от типа результата в правой части. Здесь правила преобразования очень простые: значение справа от оператора присваивания преобразуется к типу переменной слева от оператора присваивания. Если размер результата в правой части больше размера операнда в левой части, то старшая часть этого результата будет потеряна.

3.7. Ввод и вывод информации в Си

В языке Си ввод и вывод информации возможно осуществлять с использованием консоли. Ввод и вывод информации осуществляется через функции стандартной библиотеки ***stdio.h***, посредствам следующих функций:

***printf*()** - для вывода информации
***scanf_s*(), *getchar*()** - для ввода информации
***gets_s*()** – ввод строки символов

3.7.1. Функция *getchar()*

Самый простой механизм ввода заключается в чтении по одному символу за раз, обычно с терминала пользователя, с помощью функции ***getchar*()**. Функция ***getchar*()** при каждом к ней обращении возвращает следующий вводимый символ. При этом принимаемый символ будет записан в ASCII коде. ASCII коды основных символов представлены в таблице 3.1.

Функция ***getchar*()** применяется не так часто, так как может считывать только по одному символу и только в ASCII коде. Т.е. если мы используем данную функцию для ввода с клавиатуры, например, цифры 5, переменная в которую мы считаем вводимый символ будет содержать десятичное число 53 (табл. 3.1).

Пример 3.1. Код программы считывания символа с клавиатуры.

```
#include <stdio.h>                    //подключение заголовочного файла
```

```
#include <stdlib.h>                    //для перехода на русский язык

char a;                                //объявление переменных типа char
int b;                                 //объявление переменных типа integer

void main()                            //основной цикл
{                                      //начало основного цикла
        a = getchar();                 //считывание виденного символа в a
        b = getchar();                 //считывание виденного символа в b

}                                      //конец основного цикла
```

В ходе выполнения кода данной программы будут считаны два символа введенные с клавиатуры и записаны в соответствующие переменные «***a***» и «***b***». Причем первая переменная является типа ***char***, а вторая типа ***integer***.

3.7.2. Функция *printf()*

Для вывода в консоль информации используется функция ***printf*()**. Она переводит данные в символьное представление и выводит полученные изображения символов на экран. При этом у пользователя имеется возможность форматировать данные, то есть влиять на их представление на экране.

Функция ***printf*()** имеет следующую структуру:

***printf*("Строка Форматов", объект$_1$, объект$_2$, ..., объект$_n$)**

При этом на экран будет выводиться информация, заключенная в кавычках. Строка форматов состоит из следующих элементов:

1. Управляющие символы – это символы, которые не выводятся на экран, а определяют расположение выводимых символов (расположение курсора). Отличительной чертой управляющего символа является наличие обратного слэша ***«\»*** перед ним.
2. Текста, т.е. информации представленной для непосредственного вывода на экран.
3. форматов, предназначенных для вывода значений переменных различных типов. Отличительной чертой формата является наличие символа процент ***«%»*** перед ним.

В таблице 3.6 представлены основные управляющие символы, применяемые в языке Си при выводе в консоль:

Таблица 3.6.

Управляющие символы в языке Си

Символ	**Значение символа**

\n	перевод строки
\t	горизонтальная табуляция
\v	вертикальная табуляция
\b	возврат на символ
\r	возврат на начало строки
\a	звуковой сигнал

В таблице 3.7 представлены основные форматы, применяемые при выводе информации в языке Си.

Таблица 3.7.

Форматы вывода информации в языке Си

Формат	**Значение формата**
%d	целое число типа ***int*** со знаком в десятичной системе счисления
%u	целое число типа ***unsigned int***
%x	целое число типа ***int*** со знаком в шестнадцатеричной системе счисления
%o	целое число типа ***int*** со знаком в восьмеричной системе счисления
%hd	целое число типа ***short*** со знаком в десятичной системе счисления;
%hu	целое число типа ***unsigned short***
%hx	целое число типа ***short*** со знаком в шестнадцатеричной системе счисления
%ld	целое число типа ***long int*** со знаком в десятичной системе счисления
%lu	целое число типа ***unsigned long int***
%lx	целое число типа ***long int*** со знаком в шестнадцатеричной системе счисления
%f	вещественный формат (числа с плавающей точкой типа ***float***)
%lf	вещественный формат двойной точности (числа с плавающей точкой типа ***double***)
%e	вещественный формат в экспоненциальной форме (числа с плавающей точкой типа ***float*** в экспоненциальной форме)
%c	символьный формат
%s	строковый формат

Отметим, что форма данной функции позволяет выводить на экран текст и значения переменных различного типа. Причем вывод нескольких переменных может быть выполнен в одной функции. При этом их форматы объектов просто разделяются запятыми.

После использования функции ***printf*** окно консоли автоматически закрывается. Для того, чтобы это не происходило, можно после данной функции использовать функцию ***getchar***. Причем если в программе применяется ни одна функция вывода, то необходимо в конце программы прописывать столько же функций ***getchar***, т.е. команда закрытия консоли генерируется в конце каждой функции вывода в консоль. При этом если используется одна функция ***printf*** для вывода нескольких переменных, то количество функций ***getchar*** должно соответствовать количеству выводимых переменных.

Изначально в библиотеки ***stdio.h*** имеются только латинские символы. Для вывода информации в кириллице необходимо подключить еще одну библиотеку: ***stdlib.h***. В данной библиотеке функция перехода в консоли на русский язык записывается как ***system*("*chcp* 1251")**. Также часто используемой является функция отчистки экрана консоли - ***system*("*cls*")**.

Приведем ряд примеров по вводу информации с помощью функции ***getchar*()** и вывода ее на экран.

Пример 3.2. Пример программы вывода на экран фразы «Информационные технологии».

```
#include <stdio.h>                          //подключение заголовочного файла
#include <stdlib.h>                         //для перехода на русский язык

void main()                                 //основной цикл
{                                           //начало основного цикла
system("chcp 1251");                        //переходим на русский язык
system("cls");                              //очищаем окно консоли
printf("Информационные технологии\n");//выводим в консоль фразу
getchar();
}                                           //конец основного цикла
```

В результаты выполнения программы в консоль будет выведена соответствующая фраза, причем за счет использования управляющего символа ***\n*** курсор будет находиться на следующей строке:

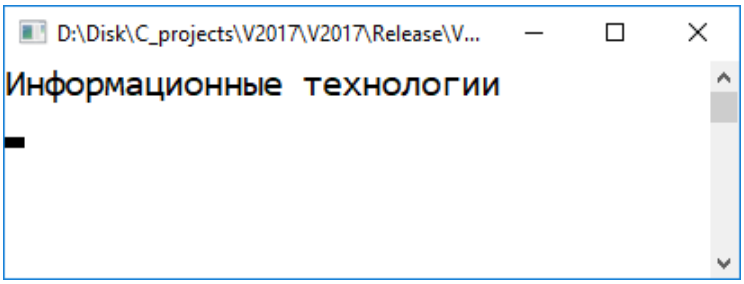


Рис. 3.1. Окно вывода в консоль строки «Информационные технологии»

Пример 3.3. Пример программы ввода трех символов и вывода на экран их ASCII кодов в десятичном виде и самих символов.

```
#include <stdio.h>                          //подключение заголовочного файла
#include <stdlib.h>                         //для перехода на русский язык

int a, b, c;                                //объявление переменных типа integer

void main()                                 //основной цикл
{                                           //начало основного цикла
system("chcp 1251");                        //переходим в консоли на русский язык
system("cls");                              //очищаем окно консоли
printf("Нажмите 3 клавиши и Enter: ");      //выводим в консоль фразу
a = getchar();                              //ожидание ввода символа с клавиатуры
b = getchar();                              //ожидание ввода символа с клавиатуры
c = getchar();                              //ожидание ввода символа с клавиатуры
printf("ASCII код клавиши 1 - %d\n",a);     //ASCII кода клавиши 1 в дес-ой форме
printf("ASCII код клавиши 2 - %d\n",b);     //ASCII кода клавиши 2 в дес-ой форме
printf("ASCII код клавиши 3 - %d\n",c);     //ASCII кода клавиши 3 в дес-ой форме
printf("Клавиша 1 - %c\n", a);              //вывод символа клавиши 1 в формате char
printf("Клавиша 2 - %c\n", b);              //вывод символа клавиши 2 в формате char
printf("Клавиша 3 - %c\n", c);              //вывод символа клавиши 3 в формате char
getchar(); getchar(); getchar();
getchar(); getchar(); getchar();
}                                           //конец основного цикла
```

Выполнение данной программы будет осуществляться по следующему алгоритму:

1. Будет открыто окно консоли (рис. 3.2а).
2. Далее необходимо ввести с клавиатуры три любых символа, например, «1», «R» и «%» (рис. 3.2б).
3. После нажатия клавиши Enter на экран будут выведены десятичные значения ASCII кодов данных символов и сами символы (рис. 3.2в).

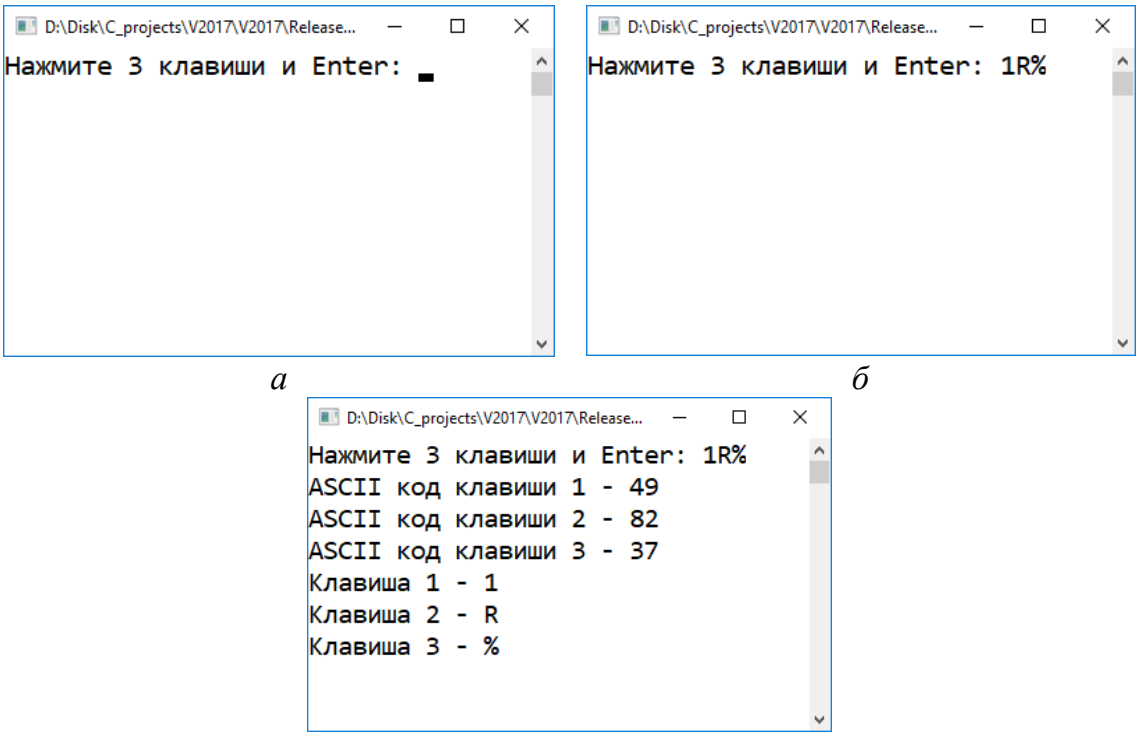


Рис. 3.2. Программа ввода трех символов и вывода на экран их ASCII кодов в десятичном виде и самих символов

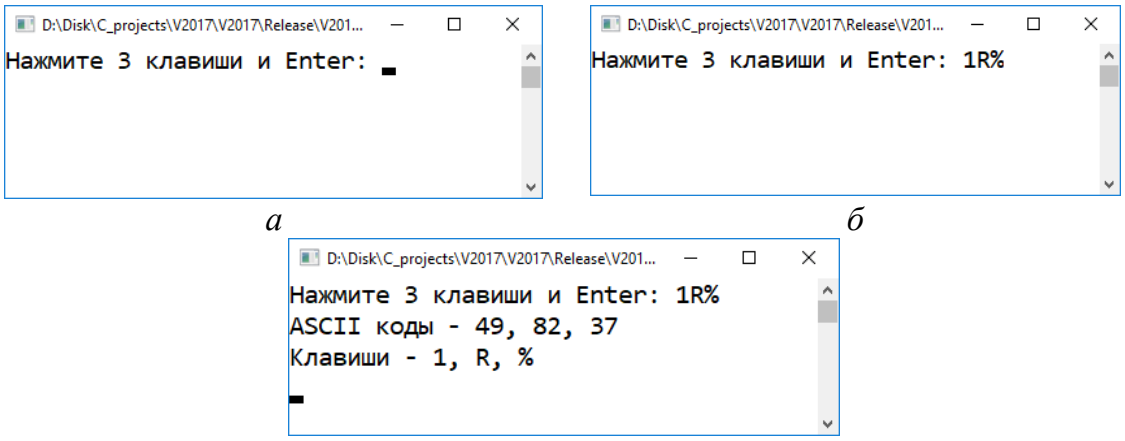
Пример 3.4. Пример программы ввода трех символов и вывода на экран их ASCII кодов в десятичном виде и самих символов. В отличие от примера 3.3, в данном случае вывод будет осуществляться с использованием двух функций ***printf***. Таким образом будет показана возможность вывода нескольких объектов в рамках одной функции вывода.

```
#include <stdio.h>                          //подключение заголовочного файла
#include <stdlib.h>                         //для перехода на русский язык

int a, b, c;                                //объявление переменных integer

void main()                                 //основной цикл
{                                           //начало основного цикла
system("chcp 1251");                        //переходим на русский язык
system("cls");                              //очищаем окно консоли
printf("Нажмите 3 клавиши и Enter: ");      //выводим в консоль фразу
a = getchar();                              //ожидание ввода символа
b = getchar();                              //ожидание ввода символа
c = getchar();                              //ожидание ввода символа
printf("ASCII коды - %d, %d, %d\n", a,b,c); //ASCII коды клавиш в дес-ой форме
printf("Клавиши - %c, %c, %c\n", a,b,c);    //вывод символов клавиш в char
```

```
getchar(); getchar();
}                                                    //конец основного цикла
```

Выполнение данной программы будет осуществляться по следующему алгоритму:

1. Будет открыто окно консоли (рис. 3.3а).
2. Далее необходимо ввести с клавиатуры три любых символа, например, «1», «R» и «%» (рис. 3.3б).
3. После нажатия клавиши Enter на экран будут выведены десятичные значения ASCII кодов данных символов и сами символы (рис. 3.3в).



а *б* *в*

Рис. 3.3. Программа ввода трех символов и вывода на экран их ASCII кодов в десятичном виде и самих символов

Пример 3.5. Пример программы ввода двух символов и вывода на экран первого символа и ASCII кода в десятичном виде второго символа.

```
#include <stdio.h>                                   //подключение заголовочного файла
#include <stdlib.h>                                  //для перехода на русский язык

char a;                                              //объявление переменных типа char
int b;                                               //объявление переменных типа
integer

void main()                                          //основной цикл
{                                                    //начало основного цикла
        system("chcp 1251");                         //переходим на русский язык
        system("cls");                               //очищаем окно консоли
        a = getchar();                               //считывание виденного символа в a
        b = getchar();                               //считывание виденного символа в b
        printf("Вы ввели символы: %c,%d", a,b);      //выводим фразу и символы
```

```
        getchar(); getchar();
}                                                    //конец основного цикла
```

После запуска данной программы необходимо будет ввести два символа. Символы могут быть как одинаковыми, так и различными. Для примера введем одинаковые символы – цифру «1» (рис. 3.4а). Далее нажав клавишу Enter на экран будет выведена фраза «Вы ввели символы: » и первым будет выведен сам символ, а через запятую его ASCI код в десятичном виде (рис. 3.4б).

а *б*

Рис. 3.4. Программа ввода трех символов и вывода на экран их ASCII кодов в десятичном виде и самих символов

3.7.3. Функция *scanf_s()*

Наиболее универсальной функцией для ввола информации является функция ***scanf_s*()**. Функция форматированного ввода данных с клавиатуры ***scanf_s*()** выполняет чтение данных, вводимых с клавиатуры, преобразует их во внутренний формат и передает вызывающей функции. Общая форма записи функции ***scanf_s*()**:

***scanf_s* ("СтрокаФорматов", адрес1, адрес2,...);**

Строка форматов аналогична функции ***printf()***. Для формирования адреса переменной используется символ амперсанд '**&**': **адрес = &объект**. Строка форматов и список аргументов для функции обязательны. Окончанием считывания с консоли будет символ нажатия клавиши ***Enter***.

Функция ***scanf_s()*** может работать сразу с несколькими переменными. Предположим, что необходимо ввести два целых числа с клавиатуры. Формально для этого можно дважды вызвать функцию ***scanf_s*(),** однако лучше воспользоваться такой конструкцией:

***scanf_s*(“ *%d, %d* ”, &*n*, &*m*);**

Функция ***scanf_s*()** интерпретирует это так, как будто ожидает, что пользователь введет число, затем – запятую, а затем – второе число. Все происходит так, как будто требуется ввести два целых числа следующим образом:

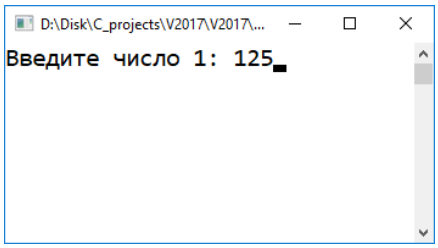
88,221 или 88, 221

Функция ***scanf_s()*** возвращает число успешно считанных элементов. Если операции считывания не происходило, что бывает в том случае, когда вместо ожидаемого цифрового значения вводится какая-либо буква, то возвращаемое значение равно 0.

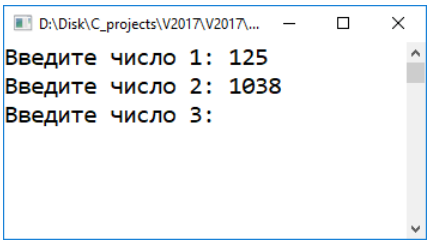
Пример 3.5. Пример программы ввода трех десятичных чисел и вывода их на экран. Для ввода чисел, а не символов в ASCII коде необходимо использовать функцию ***scanf_s()***.

```
#include <stdio.h>                      //подключение заголовочного файла
#include <stdlib.h>                     //для перехода на русский язык

int a, b, c;                            //объявление переменных типа integer

void main()                             //основной цикл
{                                       //начало основного цикла
        system("chcp 1251");            //переходим в консоли на русский язык
        system("cls");                  //очищаем окно консоли
        printf("Введите число 1: ");    //выводим в консоль фразу
        scanf_s("%d", &a);              //считывание числа в переменную a
        printf("Введите число 2: ");    //выводим в консоль фразу
        scanf_s("%d", &b);              //считывание числа в переменную b
        printf("Введите число 3: ");    //выводим в консоль фразу
        scanf_s("%d", &c);              //считывание числа в переменную c
        printf("Число 1 - %d\n", a);    //вывод числа 1 в десятичной форме
        printf("Число 2 - %d\n", b);    //вывод числа 2 в десятичной форме
        printf("Число 3 - %d\n", c);    //вывод числа 3 в десятичной форме
        getchar(); getchar(); getchar();
}                                       //конец основного цикла
```

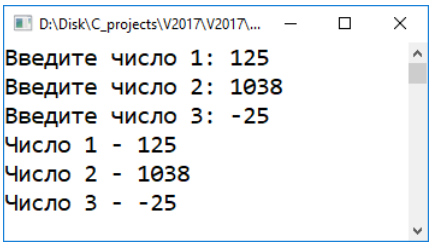
В ходе выполнения данной программы будет выведено стартовое окно для ввода первого десятичного числа, далее вводим первое число (например, 125) и нажимаем ***Enter***. Далее, вводим второе число (например, 1038) и вновь нажимаем ***Enter*** (рис. 3.5б). Вводим третье число (например, -25) и вновь нажимаем ***Enter*** и все три введенных числа будут выведены на экран (рис. 3.5в).



а



б



в

Рис. 3.5. Программа ввода трех десятичных чисел и вывода их на экран

При этом все три числа будут сохранены в соответствующих переменных «***a***», «***b***» и «***c***», соответственно. В данной программе все три переменные определены как ***integer***, и в функциях считывания и вывода использовались десятичные форматы. Поэтому при попытке ввода или вывода числа с точкой программа либо будет закрыта, либо будет выполнен некорректный вывод.

Пример 3.6. Пример программы ввода трех чисел с точкой и вывода их на экран. Например, осуществим ввод и вывод следующих чисел: 1.25, -30.54 и 250.1.

Для ввода и вывода чисел с плавающей точкой необходимо в функциях использовать формат ***%f***, а не ***%d***. А также определить переменные, в которые будет осуществляться ввод, типа ***float***.

```
#include <stdio.h>                      //подключение заголовочного файла
#include <stdlib.h>                     //для перехода на русский язык

float a, b, c;                          //объявление переменных типа float

void main()                             //основной цикл
{                                       //начало основного цикла
        system("chcp 1251");            //переходим в консоли на русский язык
        system("cls");                  //очищаем окно консоли
        printf("Введите число 1: ");    //выводим в консоль фразу
        scanf_s("%f", &a);              //считывание числа в переменную a
        printf("Введите число 2: ");    //выводим в консоль фразу
        scanf_s("%f", &b);              //считывание числа в переменную b
        printf("Введите число 3: ");    //выводим в консоль фразу
        scanf_s("%f", &c);              //считывание числа в переменную c
        printf("Число 1 - %f\n", a);    //вывод числа 1 с точкой
        printf("Число 2 - %f\n", b);    //вывод числа 2 с точкой
        printf("Число 3 - %f\n", c);    //вывод числа 3 с точкой
        getchar(); getchar(); getchar();
}                                       //конец основного цикла
```

В ходе выполнения данной программы будет выведено стартовое окно для ввода первого числа с точкой, далее вводим первое число (1.25) и нажимаем ***Enter*** (рис. 3.6а). Далее, вводим второе число (-30.54) и вновь нажимаем ***Enter*** (рис. 3.6б). Далее, вводим третье число (-250.1) и вновь нажимаем ***Enter*** и все три введенных числа будут выведены на экран (рис. 3.6в).

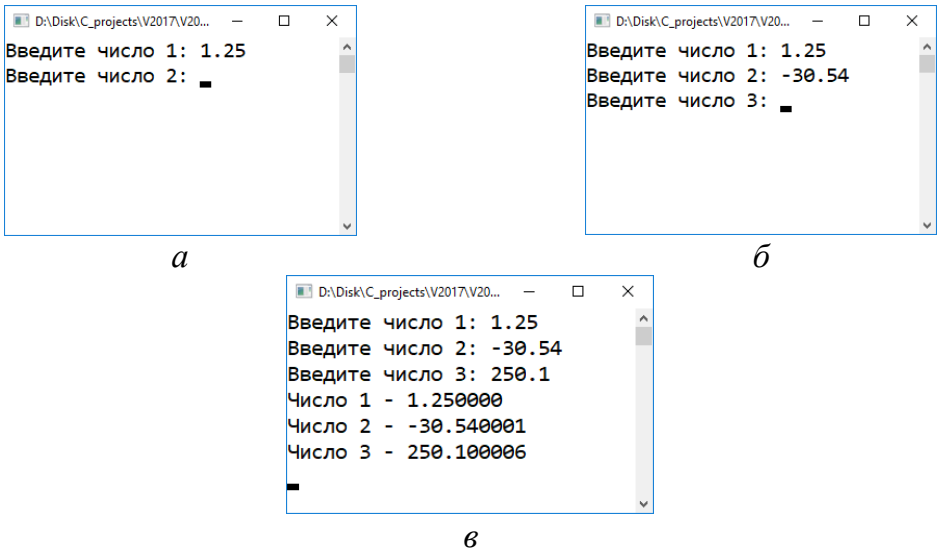


Рис. 3.6. Программа ввода трех чисел с точкой и вывода их на экран

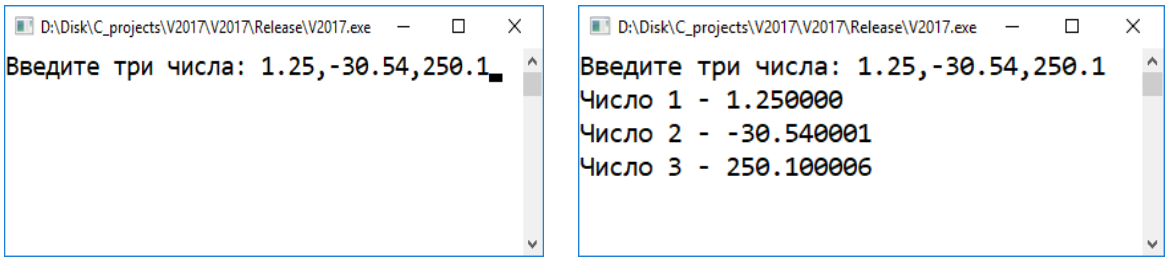
При этом все три числа будут сохранены в соответствующих переменных «***a***», «***b***» и «***c***», соответственно.

Пример 3.7. Пример программы ввода трех чисел с точкой и вывода их на экран. Например, осуществим ввод и вывод следующих чисел: 1.25, -30.54 и 250.1. при этом для ввода будем использовать одну функцию ***scanf_s*()**.

```
#include <stdio.h>                      //подключение заголовочного файла
#include <stdlib.h>                     //для перехода на русский язык

float a, b, c;                          //объявление переменных типа float

void main()                             //основной цикл
{                                       //основного цикла
        system("chcp 1251");            //переходим в консоли на русский язык
        system("cls");                  //очищаем окно консоли
        printf("Введите три числа: ");  //выводим в консоль фразу
        scanf_s("%f,%f,%f", &a,&b,&c);  //считывание трех числа в a, b, c
        printf("Число 1 - %f\n", a);    //вывод числа 1 с точкой
        printf("Число 2 - %f\n", b);    //вывод числа 2 с точкой
        printf("Число 3 - %f\n", c);    //вывод числа 3 с точкой
        getchar(); getchar(); getchar();
```

```
}                                   //конец основного цикла
```

После запуска программы на экран будет выведена фраза «Введите три числа:». Далее необходимо ввести три числа разделяя их запятыми. При этом введенные числа будут сохранены в формате плавающей точки в соответствующие переменные (рис. 3.7а). При нажатии клавиши ***Enter*** на экран будут выведены все три числа (рис. 3.7б).



а *б*

Рис. 3.7. Программа ввода трех чисел с точкой и вывода их на экран

3.7.4. Функция *get_s()*

Для считывания строки символов, например, какого-либо предложения или слова, необходимо задать следующую переменную:

***char* имя_переменной[*N*];**

где ***N*** – количество символов в строке. При этом количество символов можно задавать больше, в этом случае просто будет зарезервировано больше ячеек памяти. В ином случае можно потерять часть вводимой информации. Считывание строки будет осуществляться с помощью функции:

***get_s* (имя_переменной);**

Пример 3.8. Пример программы ввода фамилии и дальнейшего ее вывода на экран. Ввод осуществляется в символьную переменную «***a***». Для вывода строки в функции ***printf_s*** используется формат ***%s***.

```
#include <stdio.h>                  //подключение заголовочного файла
#include <stdlib.h>                 //для перехода на русский язык

char a[20];                         //объявление переменных типа char

void main()                         //основной цикл
{                                   //начало основного цикла
        system("chcp 1251");        //переходим в консоли на русский язык
        system("cls");              //очищаем окно консоли
        printf("Введите фамилию: "); //выводим в консоль фразу
```

```
	gets_s(a);                          //функция ввода строки символов в a
	printf("Ваша фамилия - %s\n", a);   //вывод введённой строки на экран
	getchar();
}                                       //конец основного цикла
```

В ходе выполнения кода программы будет выведена фраза «Введите фамилию:», вводим фамилию (рис. 3.8а). После ввода фамилии нажимаем клавишу ***Enter***. Далее будет выведена фраза «Ваша фамилия - » и фамилия (рис. 3.8б).

а *б*

Рис. 3.8. Программа ввода фамилии и дальнейшего ее вывода на экран

Пример 3.9. Пример программы ввода фамилии, имени и отчества, и вывода их в одной строке. Для вывода строки ФИО без запятых в функции ***printf_s*** используется форматы **%s** также не разделенные запятыми. В функции автоматически происходит првязка формата к соответствующей переменной (по порядку).

```
#include <stdio.h>                  //подключение заголовочного файла
#include <stdlib.h>                 //для перехода на русский язык

char fam[20];                       //объявление переменных типа char
char name[20];                      //объявление переменных типа char
char otch[20];                      //объявление переменных типа char

void main()                         //основной цикл
{                                   //начало основного цикла
	system("chcp 1251");            //переходим на русский язык
	system("cls");                  //очищаем окно консоли
	printf("Введите фамилию: ");    //выводим в консоль фразу
	gets_s(fam);                    //функция ввода строки в fam
	printf("Введите имя: ");        //выводим в консоль фразу
	gets_s(name);                   //функция ввода строки в name
	printf("Введите имя: ");        //выводим в консоль фразу
	gets_s(otch);                   //функция ввода строки в otch
	printf("Ваше ФИО - %s %s %s\n", fam,name,otch);     //вывод на экран
	getchar();
}                                   //конец основного цикла
```

При выполнении кода программы сначала необходимо будет ввести фамилию и нажать ***Enter*** (рис. 3.9а). Далее вводим имя и отчество (рис. 3.9б-в). После нажатия клавиши ***Enter*** на экран будет выведена фраза «Ваше ФИО – Фамилия Имя Отчество» (рис. 3.9г).

Рис. 3.9. Программа ввода фамилии, имени и отчества, и вывода их в одной строке

3.8. Операции языка Си

Любое выражение языка состоит из операндов (переменных, констант и др.), соединенных знаками операций. Знак операции - это символ или группа символов, которые сообщают компилятору о необходимости выполнения определенных арифметических, логических или других действий.

Операции выполняются в строгой последовательности. Величина, определяющая преимущественное право на выполнение той или иной операции, называется приоритетом. В табл. 3.8 перечислены различные операции языка Си. Их приоритеты для каждой группы одинаковы (группы выделены цветом). Чем большим преимуществом пользуется соответствующая группа операций, тем выше она расположена в таблице. Порядок выполнения операций может регулироваться с помощью круглых скобок.

Таблица 3.8.

Операции языка Си

Знак операции	Назначение операции
()	Вызов функции

[ ]	Выделение элемента массива
.	Выделение элемента записи
->	Выделение элемента записи
!	Логическое отрицание
~	Поразрядное отрицание
-	Изменение знака
++	Увеличение на единицу
--	Уменьшение на единицу
&	Взятие адреса
*****	Обращение по адресу
(тип)	Преобразование типа
sizeof( )	Определение размера в байтах
*****	Умножение
/	Деление
%	Определение остатка от деления
+	Сложение
-	Вычитание
<<	Сдвиг влево
>>	Сдвиг вправо
<	Меньше, чем
<=	Меньше или равно
>	Больше, чем
>=	Больше или равно
= =	Равно
!=	Не равно
&	Поразрядное логическое "И"
^	Поразрядное исключающее "ИЛИ"
|	Поразрядное логическое "ИЛИ"
&&	Логическое "И"
||	Логическое "ИЛИ"
?:	Условная (тернарная) операция
=	Присваивание
+=, - =, *=, /=, %=, <<=, >>=, &=, |=, ^=	Составные операции присваивания (например, a*= b, т.е. a = a*b)
,	Операция запятая

Для исключения путаницы в понятиях "операция" и "оператор", отметим, что оператор - это наименьшая исполняемая единица

программы. Различают операторы выражения, действие которых состоит в вычислении заданных выражений (например, ***a*=*sin*(*b*)+*c*** или ***j***++), операторы объявления, составные операторы, пустые операторы, операторы метки, цикла и т.д. Для обозначения конца оператора в языке Си используется точка с запятой. Что касается составного оператора (или блока), представляющего собой набор логически связанных операторов, помещенных между открывающей (**{**) и закрывающей (**}**) фигурными скобками ("операторными скобками"), то за ним точка с запятой не ставится. Отметим, что блок отличается от составного оператора наличием определений в теле блока.

Охарактеризуем основные операции языка Си

Сначала рассмотрим одну из них - ***операцию присваивания*** (=). Выражение вида

x=y;

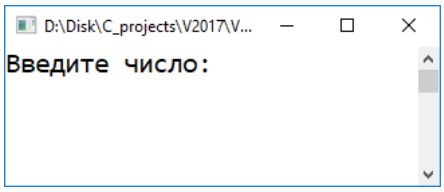
присваивает переменной ***x*** значение переменной ***y***. Операцию "=" разрешается использовать многократно в одном выражении, например,

***x=y=z*=100;**

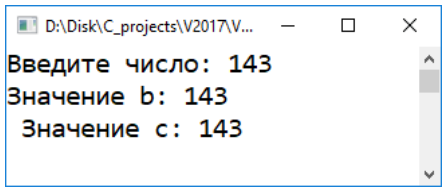
Пример 3.10. Пример программы ввода числа и присвоения его другим переменным.

```
#include <stdio.h>                  //подключение заголовочного файла
#include <stdlib.h>                 //для перехода на русский язык

int a,b,c;                          //объявление переменных типа integer

void main()                         //основной цикл
{                                   //начало основного цикла
        system("chcp 1251");        //переходим на русский язык
        system("cls");              //очищаем окно консоли
        printf("Введите число: ");  //выводим в консоль фразу
        scanf_s("%d", &a);          //считывание числа в a
        b=a;                        //операция присвоения
        c=b;
        printf("Значение b: %d\n ", b);  //выводим в консоль
        printf("Значение c: %d\n ", c);  //выводим в консоль
        getchar(); getchar();
}                                   //конец основного цикла
```

При выполнении кода программы сначала необходимо ввести число и нажать ***Enter*** (рис. 3.10а). После ввода числа оно будет присвоена переменным ***a***, ***b*** и ***c*** (рис. 3.10б).



а



б

Рис. 3.10. Программа ввода числа и присвоения его другим переменным

Различают ***унарные*** и ***бинарные*** операции. У первых из них один операнд, а у вторых - два. Начнем их рассмотрение с операций, отнесенных к первой из следующих традиционных групп:

- арифметические операции;
- логические операции и операции отношения;
- операции с битами.

Арифметические операции задаются следующими символами: **+, -, *, /, %.** Последнюю из них нельзя применять к переменным вещественного типа. Например,

a=b+c;
x=y-z;
r=t*v;
s=k/l;
p=q%w.

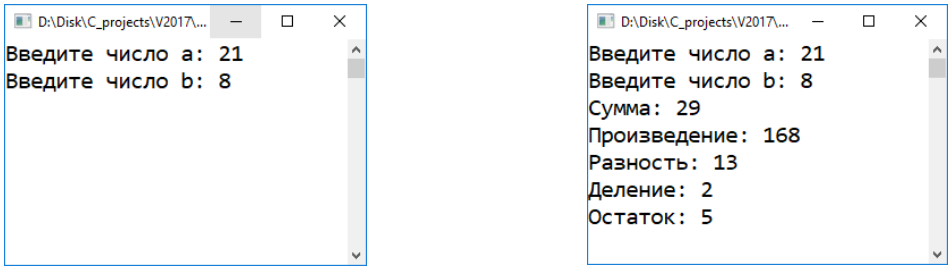
При этом в случае если операция деления присваивается в переменную типа ***integer***, то она будет хранить целую часть от деления.

Пример 3.11. Пример программы ввода двух чисел и вывода их суммы, произведения, разности, результата деления и остатка от деления.

```
#include <stdio.h>                  //подключение заголовочного файла
#include <stdlib.h>                 //для перехода на русский язык

int a,b,c,d,e,f,g;                  //объявление переменных типа integer

void main()                         //основной цикл
{                                   //начало основного цикла
        system("chcp 1251");        //переходим на русский язык
        system("cls");              //очищаем окно консоли
        printf("Введите число a: "); //выводим в консоль фразу
        scanf_s("%d", &a);          //считывание числа в a
        printf("Введите число b: "); //выводим в консоль фразу
        scanf_s("%d", &b);          //считывание числа в b
        c=a+b;                      //расчет суммы
        d=a*b;                      //расчет произведения
        e=a-b;                      //расчет разности
```

```
        f=a/b;                          //деление
        g=a%b;                          //остаток от деления
        printf("Сумма: %d\n", c);       //выводим в консоль
        printf("Произведение: %d\n", d); //выводим в консоль
        printf("Разность: %d\n", e);    //выводим в консоль
        printf("Деление: %d\n", f);     //выводим в консоль
        printf("Остаток: %d\n", g);     //выводим в консоль
        getchar(); getchar(); getchar(); getchar(); getchar();
}                                       //конец основного цикла
```

При выполнении кода программы сначала необходимо ввести два числа и нажать ***Enter*** (рис. 3.11а). После ввода чисел будут выведены результаты арифметических операций с этими числами (рис. 3.11б).



а *б*

Рис. 3.11. Программа ввода двух чисел и вывода их суммы, произведения, разности, результата деления и остатка от деления

Логические операции отношения задаются следующими символами: && ("И"), || ("ИЛИ"), ! ("НЕ"), >, >=, <, <= , = = (равно), != (не равно). Традиционно эти операции должны давать одно из двух значений: истину или ложь. В языке Си принято следующее правило: истина - это любое ненулевое значение; ложь - это нулевое значение. Выражения, использующие логические операции и операции отношения, возвращают 0 для ложного значения и 1 для истинного. Ниже приводится таблица истинности для логических операций.

Таблица 3.9.

Таблица истинности логических операций

x	**y**	**x&&y**	**x||y**	**!x**
0	0	0	0	1
0	1	0	1	1
1	0	0	1	0
1	1	1	1	0

Битовые операции можно применять к переменным, имеющим типы ***int***, ***char***. Их нельзя применять к переменным типов ***float***, ***double***, ***void*** (или более сложных типов). Эти операции задаются следующими

символами: ~ (поразрядное отрицание), << (сдвиг влево), >> (сдвиг вправо), & (поразрядное "И"), ^ (поразрядное исключающее "ИЛИ"), | (поразрядное "ИЛИ").

Пример 3.12: Программа вычисления ряда логических операций с числами ***a* = 0000 1111** и ***b* = 1000 1000**:

~*a*=1111 0000 → 240_{10} или -16_{10},
***a*<<1=0001 1110** → 30_{10},
***a*>>1=0000 0111** → 7_{10},
***a*&*b*=0000 1000** → 8_{10},
***a*^*b*=1000 0111** → 135_{10},
***a*|*b*=1000 1111** → 143_{10}.

```
#include <stdio.h>                      //подключение заголовочного файла
#include <stdlib.h>                     //для перехода на русский язык

unsigned int a, b, c, d, e, f, g, h;    //объявление переменных типа integer

void main()                             //основной цикл
{                                       //начало основного цикла
        system("chcp 1251");            //переходим на русский язык
        system("cls");                  //очищаем окно консоли
        a = 15;
        b = 136;
        c = ~a;
        d = a<<1;
        e = a>>1;
        f = a&b;
        g = a^b;
        h = a | b;
        printf("Операция ~: %d\n", c);      //выводим в консоль
        printf("Операция <<: %d\n", d);     //выводим в консоль
        printf("Операция >>: %d\n", e);     //выводим в консоль
        printf("Операция &: %d\n", f);      //выводим в консоль
        printf("Операция ^: %d\n", g);      //выводим в консоль
        printf("Операция |: %d\n", h);      //выводим в консоль
        getchar(); getchar(); getchar();
        getchar(); getchar();
}                                       //конец основного цикла
```

После выполнения данной программы окно вывода будет иметь вид, представленный на рис. 3.12:

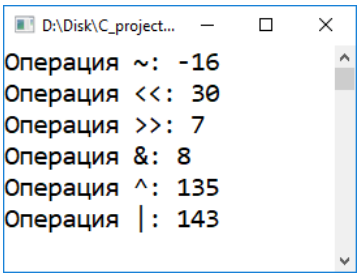


Рис. 3.12. Программа вычисления ряда логических операций

В языке предусмотрены две нетрадиционные операции инкремента (++) и декремента (--). Они предназначены для увеличения и уменьшения на единицу значения операнда. Операции ++ и -- можно записывать как перед операндом, так и после него. В первом случае (***++n*** или ***--n***) значение операнда (***n***) изменяется перед его использованием в соответствующем выражении, а во втором (***n++*** или ***n--***) - после его использования.

Пример 3.13: Рассмотрим программу, реализующую операции:

a=b+c++;

a1=b1+ ++c1;

```
#include <stdio.h>                          //подключение заголовочного файла
#include <stdlib.h>                         //для перехода на русский язык

unsigned int a, a1, b, b1, c, c1;           //объявление переменных типа integer

void main()                                 //основной цикл
{                                           //начало основного цикла
        system("chcp 1251");                //переходим на русский язык
        system("cls");                      //очищаем окно консоли
        b = 2;
        b1 = b;
        c = 4;
        c1 = c;
        a = b + c++;
        a1 = b1 + ++c1;
        printf("a=%d, b=%d, c=%d\n", a,b,c);            //выводим в консоль
        printf("a1=%d, b1=%d, c1=%d\n", a1, b1, c1);    //выводим в консоль

        getchar(); getchar();
}                                           //конец основного цикла
```

Предположим, что ***b=b1*=2**, ***c=c1*=4**. Тогда после выполнения операций: ***a*=6**, ***b*=2**, ***c*=5**, ***a1*=7**, ***b1*=2**, ***c1*=5**.

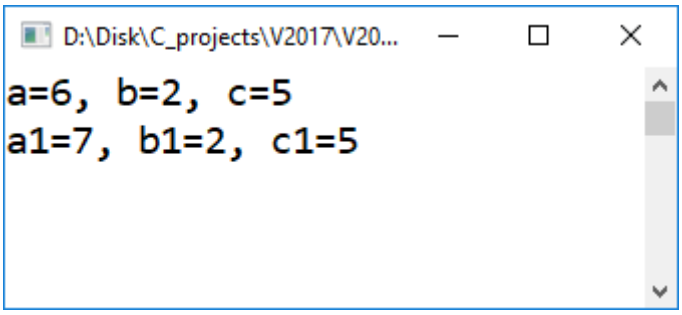


Рис. 3.13. Программа, реализующая операции a=b+c++; a1=b1+ ++c1

Широкое распространение находят также выражения с еще одной нетрадиционной тернарной или условной операцией **?**:. В выражении

y=x?a:b;

y=a, если ***x*** не равен нулю (т.е. истинно), и ***y=b***, если ***x*** равен нулю (ложно). Следующее выражение

y=(a>b)?a:b;

позволяет присвоить переменной ***y*** значение большей переменной (***a*** или ***b***), т.е. ***y=max(a,b)***.

Еще одним отличием языка является то, что выражение вида ***a=a+5***; можно записать в другой форме: ***a+=5***;. Вместо знака + можно использовать и символы других бинарных операций (табл. 3.8).

Пример 3.14: Рассмотрим программу, реализующую операции:

a=a+5;
b+=5;
c=c-10;
d-=10;

```
#include <stdio.h>                          //подключение заголовочного файла
#include <stdlib.h>                         //для перехода на русский язык

int a, b, c, d;                             //объявление переменных типа integer

void main()                                 //основной цикл
{                                           //начало основного цикла
        system("chcp 1251");                //переходим на русский язык
        system("cls");                      //очищаем окно консоли
        a=b=c=d=20;
        a=a+5;
        b+=5;
        c=c-10;
        d-=10;
        printf("a=%d, b=%d, c=%d, d=%d\n", a, b, c,d);     //выводим в консоль
        getchar();
}                                                          //конец основного цикла
```

Предположим, что ***a=b=c=d=20***, после выполнения операций результат будет выведен в консоль (рис. 3.14).

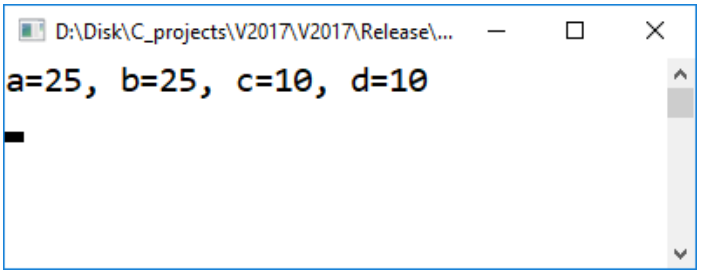


Рис. 3.14. Программа, реализующая операции a=a+5; b+=5; c=c-10; d-=10

3.9. Операторы цикла языка Си

3.9.1. Оператор цикла *for*

Циклы организуются, чтобы выполнить некоторый оператор или группу операторов определенное число раз. В языке Си три оператора цикла: ***for***, ***while*** и ***do-while***. Первый из них формально записывается, в следующем виде:

***for* (выражение_1; выражение_2; выражение_3)**
{
Тело цикла
}

Тело цикла составляет либо один оператор, либо несколько операторов, заключенных в фигурные скобки **{...}** (после блока точка с запятой не ставится). В выражениях 1, 2, 3 фигурирует специальная переменная, называемая управляющей. По ее значению устанавливается необходимость повторения цикла или выхода из него.

Выражение_1 присваивает начальное значение управляющей переменной, **выражение_3** изменяет его на каждом шаге, а **выражение_2** проверяет, не достигло ли оно граничного значения, устанавливающего необходимость выхода из цикла. Пример:

***for* (*i* = 1; *i* < 10; *i*++)**
{
Тело цикла
}

В данном примере изначальное значение переменной ***i*** равно1. При каждом проходе тела цикла будет выполняться инкремент переменной **i**. И тело цикло было выполняться пока выполняется условие ***i*<10**.

Любое из трех выражений в цикле ***for*** может отсутствовать, однако точка с запятой должна оставаться. Таким образом, ***for* (; ;) {...}**

- это бесконечный цикл, из которого можно выйти лишь другими способами.

Допускаются вложенные конструкции, т.е. в теле некоторого цикла могут встречаться другие операторы ***for***.

Пример 3.15. Пример программы расчета суммы чисел от 1 до 100 с выводом результата расчета на экран (***способ 1***).

```
#include <stdio.h>                    //подключение заголовочного файла
#include <stdlib.h>                   //для перехода на русский язык

int i,sum;                            //объявление переменных типа int

void main()                           //основной цикл
{                                     //начало основного цикла
        system("chcp 1251");          //переходим на русский язык
        system("cls");                //очищаем окно консоли
        sum = 0;
        for (i = 1; i < 101; i++)     //расчет суммы
        {
        sum = sum + i;
        }
        printf("Сумма чисел от 1 до 100 = %d", sum);//вывод суммы
        getchar();
}                                     //конец основного цикла
```

Пример 3.16. Пример программы расчета суммы чисел от 1 до 100 с выводом результата расчета на экран (***способ 2***).

```
#include <stdio.h>                    //подключение заголовочного файла
#include <stdlib.h>                   //для перехода на русский язык

int i,sum;                            //объявление переменных типа int

void main()                           //основной цикл
{                                     //начало основного цикла
        system("chcp 1251");          //переходим в консоли на русский язык
        system("cls");                //очищаем окно консоли
        sum = 0;
        for (i = 1; i < 101; i++)     //расчет суммы
        {
        sum += i;
        }
        printf("Сумма чисел от 1 до 100 = %d", sum);      //вывод суммы
        getchar();
}                                     //конец основного цикла
```

Пример 3.17. Пример программы расчета суммы чисел от 1 до 100 с выводом результата расчета на экран (способ 3).

```
#include <stdio.h>                          //подключение заголовочного файла
#include <stdlib.h>                         //для перехода на русский язык

int i,sum;                                  //объявление переменных типа int

void main()                                 //основной цикл
{                                           //начало основного цикла
        system("chcp 1251");                //переходим на русский язык
        system("cls");                      //очищаем окно консоли
        sum = 0;
        for (i = 1; i != 101; i++)          //расчет суммы
        {
                sum += i;
        }
        printf("Сумма чисел от 1 до 100 = %d", sum);      //вывод суммы
        getchar();
}                                           //конец основного цикла
```

В примере 1 используется суммирование с помощью операции ***sum=sum+i***. Также суммирование можно выполнять как во втором и третьем примерах с помощью операции ***sum+=i***.

В примерах 1 и 2 оператор цикла ***for*** выполняет тело цикла пока переменная ***i*<101**, в примере 3 тело цикла выполняется пока переменная ***i*** не равна 101 (**!=**). Отличие будет заключаться в том, что если в примерах 1 и 2 в теле цикла переменная ***i*** станет больше 101, то цикл закончиться. В третьем же примере проверяется условия равенства переменной числу 101, следовательно, если она станет в теле цикла больше 101, то цикл продолжиться дальше.

Во всех трех примерах вывод на экран будет следующий:

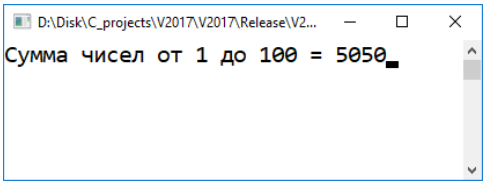


Рис. 3.15. Программа расчета суммы чисел от 1 до 100 с выводом результата расчета на экран

Пример 3.18. Пример программы вывода на экран всех четных чисел от 0 до 20 с использованием оператора цикла ***for***.

```
#include <stdio.h>                          //подключение заголовочного файла
#include <stdlib.h>                         //для перехода на русский язык
```

```
int i;                                  //объявление переменных типа int

void main()                             //основной цикл
{                                       //начало основного цикла
        system("chcp 1251");            //переходим в консоли на русский язык
        system("cls");                  //очищаем окно консоли

        for (i = 0; i < 21;i=i+2)       //расчет суммы
        {
                printf("%d\n", i);      //вывод четных чисел
        }
        getchar();
}
```

Результат выполнения программы представлен на рис. 3.16.

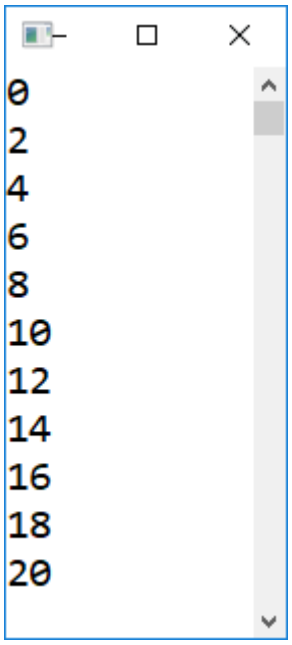


Рис. 3.16. Программа вывода на экран всех четных чисел от 0 до 20 с использованием оператора цикла ***for***

3.9.2. Оператор цикла *while*

Оператор ***while*** формально записывается в таком виде:

while (выражение)
{
Тело цикла
}

Выражение в скобках может принимать ненулевое (истинное) или нулевое (ложное) значение. Если оно истинно, то выполняется тело

цикла и выражение вычисляется снова. Если выражение ложно, то цикл ***while*** заканчивается.

Условие в цикле ***while*** может отсутствовать, при этом выражение ***while*() {...}** – это бесконечный цикл, из которого можно выйти лишь другими способами.

Пример 3.19. Пример программы расчета суммы чисел от 1 до 100 с выводом результата расчета на экран.

```
#include <stdio.h>                    //подключение заголовочного файла
#include <stdlib.h>                   //для перехода на русский язык

int i,sum;                            //объявление переменных типа int

void main()                           //основной цикл
{                                     //начало основного цикла
        system("chcp 1251");          //переходим в консоли на русский язык
        system("cls");                //очищаем окно консоли
        sum = 0;
        i = 1;
        while (i<101)                 //расчет суммы
        {
        sum += i;
        i++;
        }
        printf("Сумма чисел от 1 до 100 = %d", sum);//вывод суммы
        getchar();
}                                     //конец основного цикла
```

После выполнения программы будет выведен следующий результат (рис. 3.17):

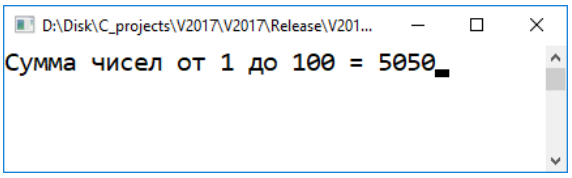


Рис. 3.17. Программа расчета суммы чисел от 1 до 100 с выводом результата расчета на экран

Пример 3.20. Пример программы вывода на экран всех нечетных чисел от 0 до 20 с использованием оператора цикла ***while***.

```
#include <stdio.h>                    //подключение заголовочного файла
#include <stdlib.h>                   //для перехода на русский язык

int i;                                //объявление переменных типа int
```

```
void main()                          //основной цикл
{                                    //начало основного цикла
        system("chcp 1251");         //переходим в консоли на русский язык
        system("cls");               //очищаем окно консоли
        i = 1;
        while (i<20)                 //расчет чисел
        {
        printf("%d\n", i);           //вывод нечетных чисел
        i = i + 2;
        }
        getchar();
}                                    //конец основного цикла
```

Результат выполнения программы представлен на рис. 3.18.

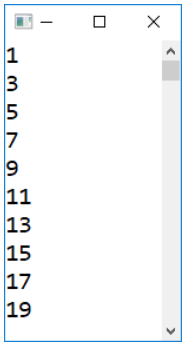


Рис. 3.18. Программа вывода на экран всех нечетных чисел от 0 до 20 с использованием оператора цикла ***while***

3.9.3. Оператор цикла *do-while*

Оператор ***do-while*** формально записывается следующим образом:

do
{
тело цикла
}
***while* (выражение);**

Основным отличием между циклами ***while*** и ***do-while*** является то, что тело в цикле ***do-while*** выполняется по крайней мере один раз. Тело цикла будет выполняться до тех пор, пока выражение в скобках не примет ложное значение. Если оно ложно при входе в цикл, то его тело выполняется ровно один раз. Необходимо обратить внимание, что в цикле ***do-while*** после ***while*()** необходимо ставить «***;***»

Допускается вложенность одних циклов в другие, т.е. в теле любого цикла могут появляться операторы ***for***, ***while*** и ***do-while*.**

В теле цикла могут использоваться новые операторы ***break*** и ***continue***. Оператор ***break*** обеспечивает немедленный выход из цикла, оператор ***continue*** вызывает прекращение очередной и начало следующей итерации.

Пример 3.21. Пример программы расчета суммы чисел от 1 до 100 с выводом результата расчета на экран.

```
#include <stdio.h>                    //подключение заголовочного файла
#include <stdlib.h>                   //для перехода на русский язык

int i,sum;                            //объявление переменных типа int

void main()                           //основной цикл
{                                     //начало основного цикла
        system("chcp 1251");          //переходим в консоли на русский язык
        system("cls");                //очищаем окно консоли
        sum = 0;
        i = 1;
        do                            //расчет суммы
        {
                sum += i;
                i++;
        }

        while (i<101)

        printf("Сумма чисел от 1 до 100 = %d", sum);//вывод суммы
        getchar();
}                                     //конец основного цикла
```

Результат выполнения программы представлен на рис. 3.19.

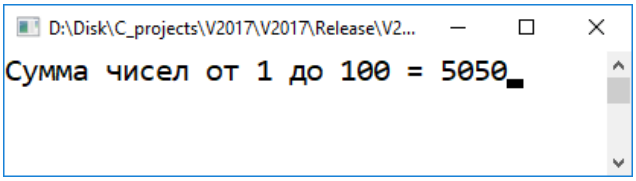


Рис. 3.19. Программа расчета суммы чисел от 1 до 100 с выводом результата расчета на экран

Пример 3.22. Пример программы вывода на экран всех нечетных чисел от 0 до 20 с использованием оператора цикла ***do-while***.

```
#include <stdio.h>                    //подключение заголовочного файла
#include <stdlib.h>                   //для перехода на русский язык
```

```
int i;                                  //объявление переменных типа int

void main()                             //основной цикл
{                                       //начало основного цикла
        system("chcp 1251");            //переходим в консоли на русский язык
        system("cls");                  //очищаем окно консоли
        i = 1;
        do                              //расчет чисел
        {
        printf("%d\n", i);              //вывод нечетных чисел
        i = i + 2;
        }
        while (i<20)
        getchar();
}                                       //конец основного цикла
```

Результат выполнения программы представлен ан рис. 3.20.

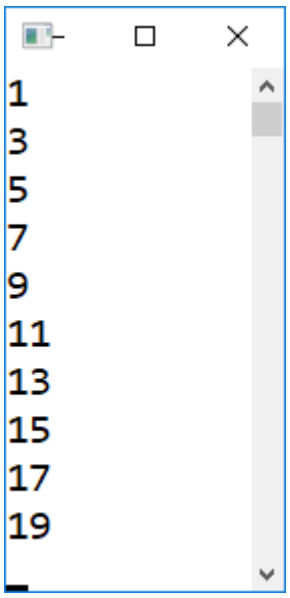


Рис. 3.20. Программа вывода на экран всех нечетных чисел от 0 до 20

3.10. Операторы условных и безусловных переходов языка Си

3.10.1.Оператор *if*

Для организации условных и безусловных переходов в программе на языке Си используются операторы: ***if - else***, ***switch*** и ***goto***.

Оператор ***if-else*** записывается следующим образом:

_if_ (проверка_условия)
{
оператор_1;

}
else
{
оператор_2;
}

Если условие в скобках принимает истинное значение, выполняется ***оператор_1***, если ложное - ***оператор_2***. Если необходимо выполнить только один оператор, то их можно не заключать в фигурные скобки. В операторе ***if*** слово ***else*** может отсутствовать.

В операторе ***if - else*** непосредственно после ключевых слов ***if*** и ***else*** должны следовать другие операторы. Если хотя бы один из них является оператором ***if***, его называют вложенным. Согласно принятому в языке Си соглашению слово ***else*** всегда относится к ближайшему предшествующему ему ***if***.

Пример 3.23. Пример программы сравнения двух чисел, введенных с клавиатуры, с выводом результата на экран.

```
#include <stdio.h>                  //подключение заголовочного файла
#include <stdlib.h>                 //для перехода на русский язык

int a, b;                           //объявление переменных типа integer

void main()                         //основной цикл
{                                   //начало основного цикла
        system("chcp 1251");        //переходим в консоли на русский язык
        system("cls");              //очищаем окно консоли
        printf("Введите число 1: "); //выводим в консоль фразу
        scanf_s("%d", &a);          //считывание числа в переменную a
        printf("Введите число 2: "); //выводим в консоль фразу
        scanf_s("%d", &b);          //считывание числа в переменную b

        if (a > b)
        {
                printf("Первое число больше\n");
        }
        else
        {
                if (a == b)
                {
                        printf("Числа равны\n");
                }
                else
                {
                        printf("Второе число больше\n");
```

```
        }
    }

    getchar();getchar();getchar();
    }                                   //конец основного цикла
```

Ниже представлены результаты выполнения программы для разных вариантов введенных чисел (рис. 3.21).

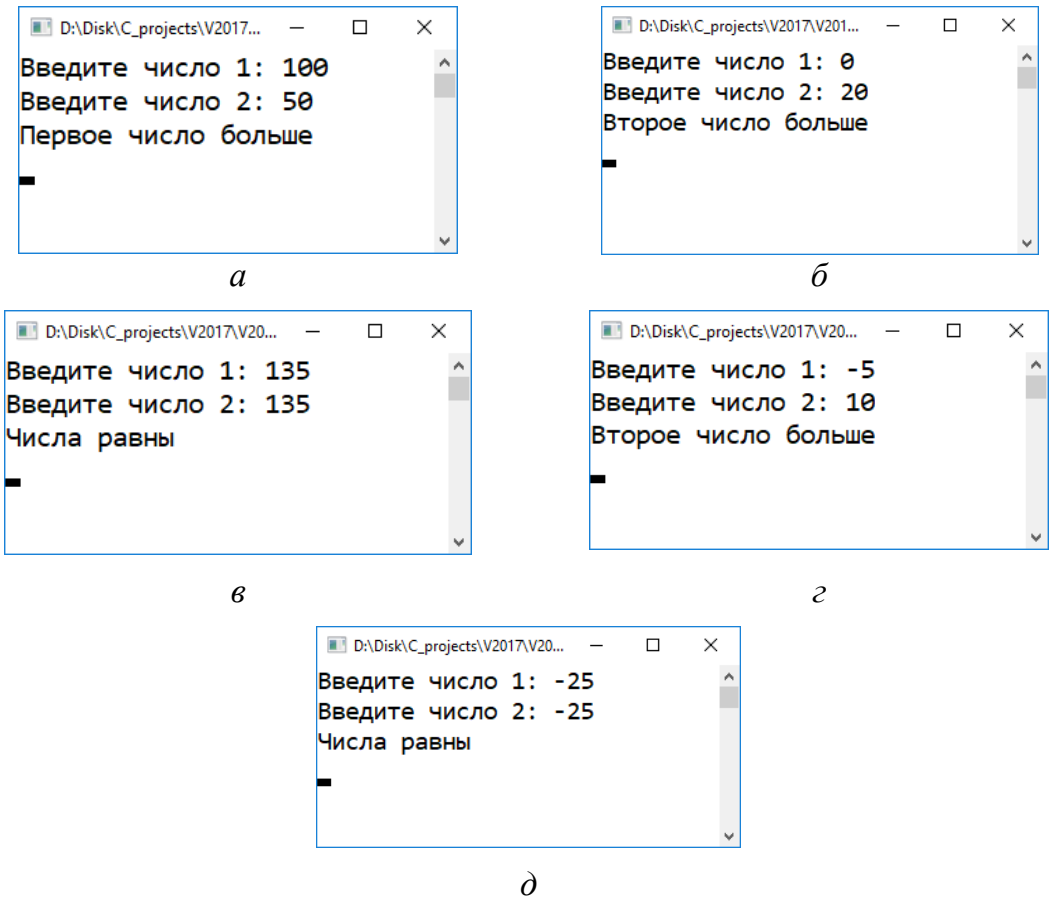


Рис. 3.21. Программа сравнения двух чисел, введенных с клавиатуры, с выводом результата на экран: а) первое число – 100, второе число – 50; б) первое число – 0, второе число – 20; в) первое число – 135, второе число – 135; г) первое число – -5, второе число – 10; д) первое число – -25, второе число – -25

3.10.2.Оператор switch

Оператор ***switch*** позволяет выбрать одну из нескольких альтернатив. Он записывается в следующем формальном виде:

```
switch (выражение)
{
        case константа_1: операторы_1;
```

***break*;**
***case* константа_2: операторы_2;**
***break*;**
***default*: операторы_*default*;**
}

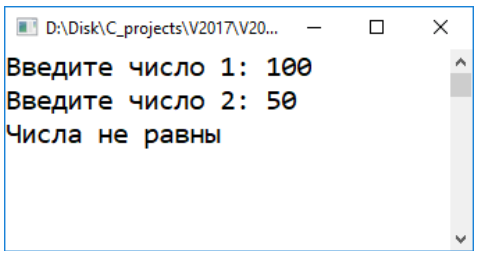
Здесь вычисляется значение целого выражения в скобках (его иногда называют селектором) и оно сравнивается со всеми константами (константными выражениями). Все константы должны быть различными. При совпадении выполнится соответствующий вариант операторов (один или несколько операторов). Вариант с ключевым словом ***default*** реализуется, если ни один другой не подошел (слово ***default*** может и отсутствовать). Если ***default*** отсутствует, а все результаты сравнения отрицательны, то ни один вариант не выполняется.

Для прекращения последующих проверок после успешного выбора некоторого варианта используется оператор ***break***, обеспечивающий немедленный выход из переключателя ***switch***.

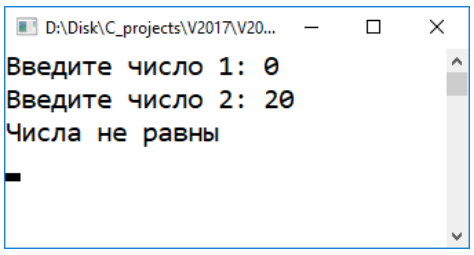
Пример 3.24. Пример программы определения равенства двух чисел, введенных с клавиатуры, с выводом результата на экран.

```
#include <stdio.h>                  //подключение заголовочного файла
#include <stdlib.h>                 //для перехода на русский язык

int a, b,c;                         //объявление переменных типа integer

void main()                         //основной цикл
{                                   //начало основного цикла
        system("chcp 1251");        //переходим в консоли на русский язык
        system("cls");              //очищаем окно консоли
        printf("Введите число 1: "); //выводим в консоль фразу
        scanf_s("%d", &a);          //считывание числа в переменную a
        printf("Введите число 2: "); //выводим в консоль фразу
        scanf_s("%d", &b);          //считывание числа в переменную b
        c = a - b;                  //вычисляем разницу
        switch (c)
        {
        case 0: printf("Числа равны\n");
        break;
        default: printf("Числа не равны\n");
        }

        getchar(); getchar();
        }                           //конец основного цикла
```

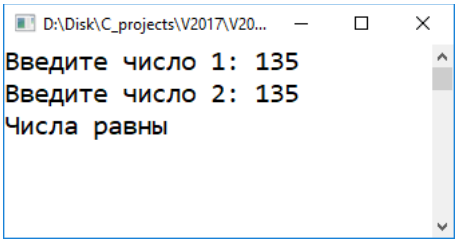
На рис. 3.22 представлены результаты выполнения программы для разных вариантов введенных чисел.



а



б



в

Рис. 3.22. Программа сравнения двух чисел, введенных с клавиатуры, с выводом результата на экран: а) первое число – 100, второе число – 50; б) первое число – 0, второе число – 20; в) первое число – 135, второе число – 135

3.10.3.Оператор *goto*

Рассмотрим правила выполнения безусловного перехода, который можно представить в следующей форме:

***goto* метка;**

Метка - это любой идентификатор, после которого поставлено двоеточие. Оператор ***goto*** указывает на то, что выполнение программы необходимо продолжить, начиная с оператора, перед которым записана метка. Метку можно поставить перед любым оператором в той функции, где находится соответствующий ей оператор ***goto***. Метку не надо объявлять.

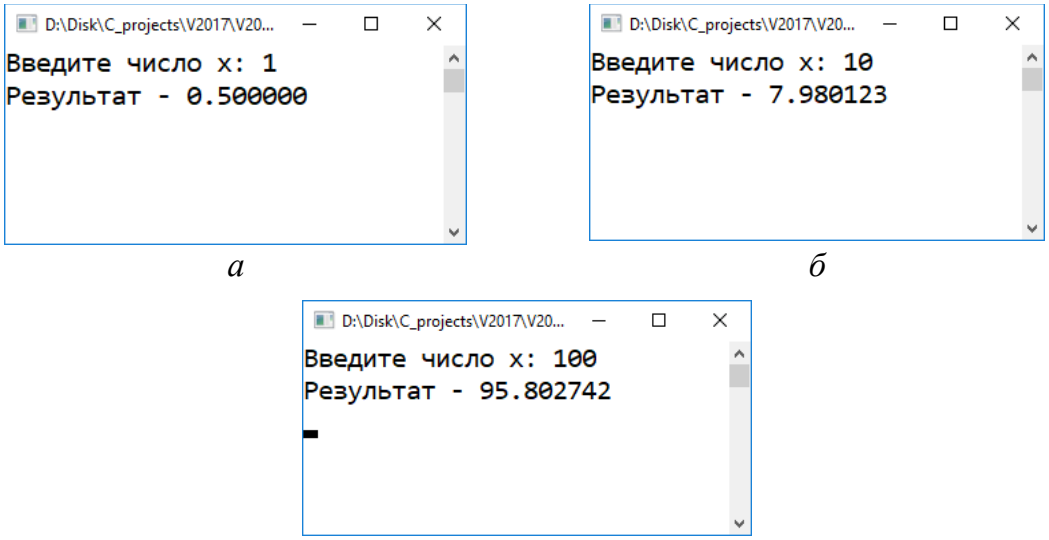
Пример 3.25. Пример программы расчета значения $s = \sum_{n=1}^{x} \frac{n}{n+1}$.

Значение ***x*** вводится с клавиатуры.

```
#include <stdio.h>                  //подключение заголовочного файла
#include <stdlib.h>                 //для перехода на русский язык

float n,x;                          //объявление переменных типа float
float s;                            //объявление переменных типа float
```

```
void main()                          //основной цикл
{                                    //начало основного цикла
        system("chcp 1251");         //переходим в консоли на русский язык
        system("cls");               //очищаем окно консоли
        printf("Введите число x: "); //выводим в консоль фразу
        scanf_s("%f", &x);           //считывание числа в переменную x

m1:  ++n;
        if (n <= x)
        {
                s += n/(n+1);
                goto m1;
        }

        printf("Результат - %f\n",s);
        getchar(); getchar();
        }                            //конец основного цикла
```

На рис. 3.23 представлены результаты выполнения данной программы для различных значений переменной ***x*** (1, 10 и 100).



а *б*

в

Рис. 3.23. Программа расчета значения $s = \sum_{n=1}^{x} \frac{n}{n+1}$ *.: а)* ***x****=1; б)* ***x****=10; в)* ***x****=100*

3.11. Структурированные типы данных

3.11.1.Массивы

Массив состоит из элементов одного и того же типа. Ко всему массиву целиком можно обращаться по имени. Кроме того, можно выбирать любой элемент массива. Для этого необходимо задать индекс, который указывает на его относительную позицию. Число элементов массива назначается при его определении и в дальнейшем не изменяется. Если массив объявлен, то к любому его элементу можно обратиться следующим образом: указать имя массива и индекс элемента в квадратных скобках. Массивы определяются так же, как и переменные:

***int a*[100]**
***char b*[20]**
***float d*[50]**

В первой строке объявлен массив ***a*** из 100 элементов целого типа: ***a*[0], *a*[1], ..., *a*[99]** (**индексация всегда начинается с нуля**). Во второй строке элементы массива ***b*** имеют тип ***char***, а в третьей - ***float***.

Имя массива – это константа, которая содержит адрес его первого элемента (в первом примере, ***a*** содержит адрес элемента ***a*[0]**). Предположим, что ***a*=**1000. Тогда, адрес элемента ***a*[1]** будет равен 1002 (элемент типа ***int*** занимает в памяти 2 байта), адрес следующего элемента ***a*[2]** - 1004 и т.д.

Язык Си позволяет инициализировать массив при его определении. Для этого используется следующая форма:

тип имя_массива[...]={список значений}

Пример 3.26. Пример программы объявления различных массивов.

```
#include <stdio.h>                  //подключение заголовочного файла
#include <stdlib.h>                 //для перехода на русский язык

int a[4] = {10,21,32,43};           //объявление массива типа integer
float s[5] = {0.1,0.2,0.3,0.4,0.5}; //объявление массива типа float
char c[3] = { 'a','b','c' };        //объявление массива типа char


void main()                         //основной цикл
{                                   //начало основного цикла
}                                   //конец основного цикла
```

Пример 3.27. Пример программы объявления различных массивов и вывода этих массивов на экран.

```
#include <stdio.h>                          //подключение заголовочного файла
#include <stdlib.h>                         //для перехода на русский язык

int a[5] = {10,21,32,43,55};                //объявление массива типа integer
float s[5] = {0.1,0.2,0.3,0.4,0.5};         //объявление массива типа float
char c[5] = { 'a','b','c','d','e' };        //объявление массива типа char
int i = 0;

void main()                                 //основной цикл
{                                           //начало основного цикла
        system("chcp 1251");                //переходим в консоли на русский язык
        system("cls");                      //очищаем окно консоли
        i = 0;
        while (i < 5)
        {
                printf("Элемент %d - %d - %f - %c\n", i, a[i], s[i], c[i]);
                i++;
        }
        getchar(); getchar();
}                                           //конец основного цикла
```

После выполнения программы на экран будут выведены элементы массивов (рис. 3.24).

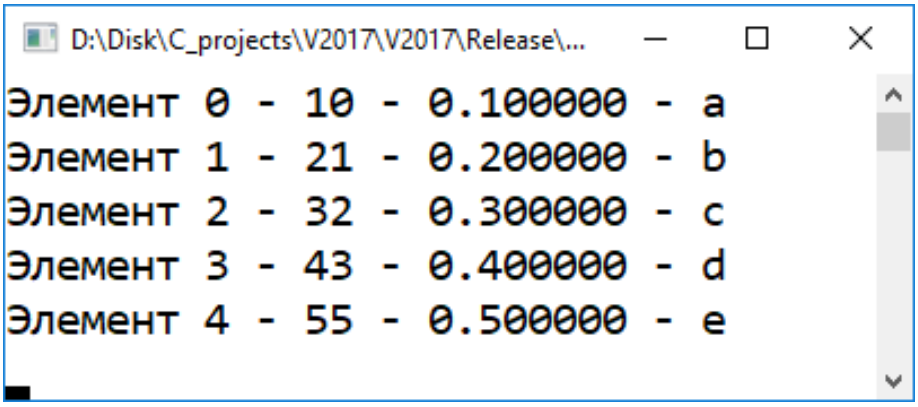


Рис. 3.24. Программа объявления различных массивов и вывода этих массивов на экран

Пример 3.28. Пример программы выбора из массива максимального числа и вывода его на экран.

```
#include <stdio.h>                          //подключение заголовочного файла
#include <stdlib.h>                         //для перехода на русский язык
```

```
int a[15] = {10,21,32,43,55,-10,5,3,28,91,92,1,2,65,-7};
int i = 0,j;

void main()                                    //основной цикл
{                                              //начало основного цикла
        system("chcp 1251");                   //переходим в консоли на русский язык
        system("cls");                         //очищаем окно консоли
        i = 0;
        j = 0;
        while (i < 15)
        {
                if (j < a[i])
                {
                        j = a[i];
                }
                i++;
        }
        printf("Максимальное число - %d\n", j);
        getchar(); getchar();
}                                              //конец основного цикла
```

После выполнения программы на экран будут выведено максимальное число массива (рис. 3.25).

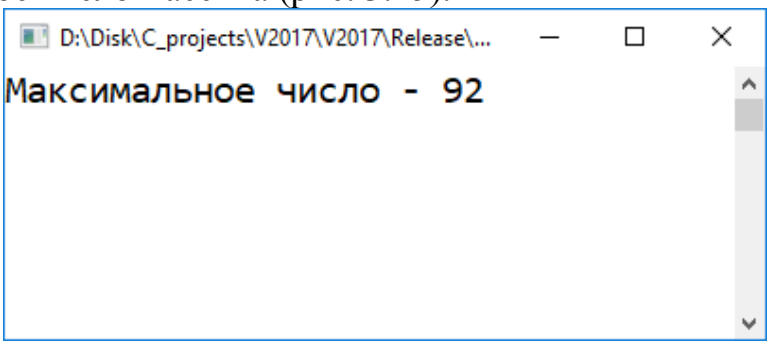


Рис. 3.25. Программа выбора из массива максимального числа и вывода его на экран

3.11.2.Структуры

Структура – это объединение одного или нескольких объектов (переменных, массивов, указателей, других структур и т.д.). Как и массив, она представляет собой совокупность данных. Отличием является то, что к ее элементам необходимо обращаться по имени и что различные элементы структуры не обязательно должны принадлежать одному типу.

Объявление структуры осуществляется с помощью ключевого слова ***struct***, за которым идет ее тип и далее список элементов, заключенных в фигурные скобки:

***struct* тип { тип элемента_1 имя элемента_1;**
…
тип элемента_*n* имя элемента_*n*;
};

Именем элемента может быть любой идентификатор. Как и выше, в одной строке можно записывать через запятую несколько идентификаторов одного типа.

***sruct date* { *int day*;**
***int month*;**
***int year*;**
};

Следом за фигурной скобкой, заканчивающей список элементов, могут записываться переменные данного типа, например:

***struct date* {...} *a, b, c*;**

(при этом выделяется соответствующая память). Описание без последующего списка не выделяет никакой памяти; оно просто задает форму структуры. Введенное имя типа позже можно использовать для объявления структуры, например:

***struct date days*;**

Теперь переменная ***days*** имеет тип ***date***.

При необходимости структуры можно инициализировать, помещая вслед за описанием список начальных значений элементов.

Разрешается вкладывать структуры друг в друга, например:

***struct man* { *char name*[20], *fam*[20];**
***struct date bd*;**
***int age*;**
};

Определенный выше тип ***data*** включает три элемента: ***day***, ***month***, ***year***, содержащий целые значения (***int***). Структура man включает элементы ***name***, ***fam***, ***bd*** и ***age***. Первые два - ***name***[20] и ***fam***[20] - это символьные массивы из 20 элементов каждый. Переменная ***bd*** представлена составным элементом (вложенной структурой) типа ***data***. Элемент ***age*** содержит значения целого типа ***int***. Теперь можно определить переменные, значения которых принадлежат введенному типу:

***struct man man*_[100];**

Здесь определен массив ***man***, состоящий из 100 структур типа ***man***.

Чтобы обратиться к отдельному элементу структуры, необходимо указать имя элемента массива, поставить точку и сразу же за ней записать имя нужного элемента структуры, например:

***man*_[j].*age* = 19;**
***man*_[j].*bd.day* = 24;**
***man*_[j].*bd.month* = 2**
***man*_[j].*bd.year* = 1987;**

3.12. Функции

Программы на языке Си обычно состоят из большого числа отдельных функций (подпрограмм). Как правило, эти функции имеют небольшие размеры и могут находиться как в одном, так и в нескольких файлах. Все функции являются глобальными. В языке запрещено определять одну функцию внутри другой. Связь между функциями осуществляется через аргументы, возвращаемые значения и внешние переменные.

В общем случае функции в языке Си необходимо объявлять. Объявление функции (т.е. описание заголовка) должно предшествовать ее использованию, а определение функции (т.е. полное описание) может быть помещено как после тела программы (т.е. функции ***main***), так и до него. Если функция определена до тела программы, а также до ее вызовов из определений других функций, то объявление может отсутствовать. Как уже отмечалось, описание заголовка функции обычно называют прототипом функции.

Функция объявляется следующим образом:

тип имя_функции(тип имя_параметра_1, тип имя_параметра_2, ...);

Тип функции определяет тип значения, которое возвращает функция. Если тип не указан, то предполагается, что функция возвращает целое значение (***int***).

При объявлении функции для каждого ее параметра можно указать только его тип (например: ***тип функции (int, float, ...)***), а можно дать и его имя (например: ***тип функции (int a, float b, ...)***).

В языке Си разрешается создавать функции с переменным числом параметров. Тогда при задании прототипа вместо последнего из них указывается многоточие.

Определение функции имеет следующий вид:

тип имя_функции(тип имя_параметра_1, тип имя_параметра_2,...)
{

тело функции
}

Передача значения из вызванной функции в вызвавшую происходит с помощью оператора возврата ***return***, который записывается следующим образом:

***return* выражение;**

Таких операторов в подпрограмме может быть несколько, и тогда они фиксируют соответствующие точки выхода. **Например**:

```
int f(int a, int b)
{
if (a > b)
{
a=2*a; return a;
}
b=2*b; return b;
}
```

Вызвать эту функцию можно следующим образом:

```
c = f(15, 5);
c = f(d, g);
f(d, g);
```

Вызвавшая функция может, при необходимости, игнорировать возвращаемое значение. После слова ***return*** можно ничего не записывать; в этом случае вызвавшей функции никакого значения не передается. Управление передается вызвавшей функции и в случае выхода "по концу" (последняя закрывающая фигурная скобка).

В языке Си аргументы функции передаются по значению, т.е. вызванная функция получает свою временную копию каждого аргумента. Вызванная функция не может изменить значение переменной вызвавшей ее программы.

Если некоторые переменные, константы, массивы, структуры объявлены как глобальные, то их не надо включать в список параметров вызванной функции.

Пример 3.29. Пример программы расчета периметра квадрата, площади квадрата или объема куба. Сторона куба и требуемый расчет вводится с клавиатуры.

```
#include <stdio.h>                  //подключение заголовочного файла
#include <stdlib.h>                 //для перехода на русский язык

int a,R;                            //объявление переменных типа integer
char b;                             //объявление переменных типа char
```

```
void vol(void);                         //объявление функции vol
void area(void);                        //объявление функции area
void perim(void);                       //объявление функции perim

void main()                             //основной цикл
{                                       //начало основного цикла
        system("chcp 1251");            //переходим в консоли на русский язык
        system("cls");                  //очищаем окно консоли
        printf("Введите сторону квадрата и параметр расчета: ");
        scanf_s("%d,%c", &a,&b);        //считывание

        if (b == 'S')                   //проверка условий
        {
                area();                 //вызов функции area
        }
        else
        {
                if (b == 'V')
                {
                        vol();          //вызов функции vol
                }
                else
                {
                        perim();        //вызов функции perim
                }
        }

        getchar(); getchar(); getchar();
}                                       //конец основного цикла


void vol(void)                          //функция vol
{
        R = a*a*a;
        printf("Объем - %d\n", R);
}

void area(void)                         //функция area
{
        R = a*a;
        printf("Площадь - %d\n", R);
}

void perim(void)                        //функция perim
{
        R = a+a+a+a;
        printf("Периметр - %d\n", R);
```

}

На рис. 3.26 представлен результаты выполнения данной программы.

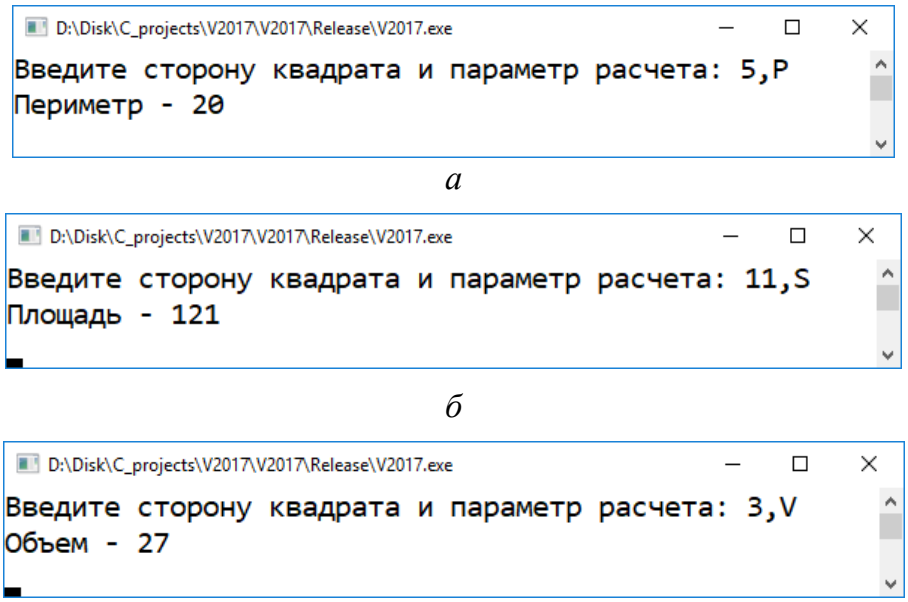


Рис. 3.26. Программа расчета периметра квадрата, площади квадрата или объема куба: а) сторона – 5, расчет периметра; б) сторона – 11, расчет площади; в)) сторона – 3, расчет объема

Глава 4. Лабораторные работы

Лабораторная работа №1

Создание проекта в программе Microsoft Visual Studio 2017. Структура программы Си. Типы данных в Си.

1. Цель работы

Целью работы является знакомство с программой Microsoft Visual Studio. Компиляция и сборка программы на языке Си. Выявление ошибок.

2. Создание и отладка проекта в программе Microsoft Visual Studio 2017

В ходе лабораторной работы используется программа Microsoft Visual Studio 2017. После открытия программы будет открыто следующее окно (рис. 4.1), в котором необходимо нажать «Войти»:

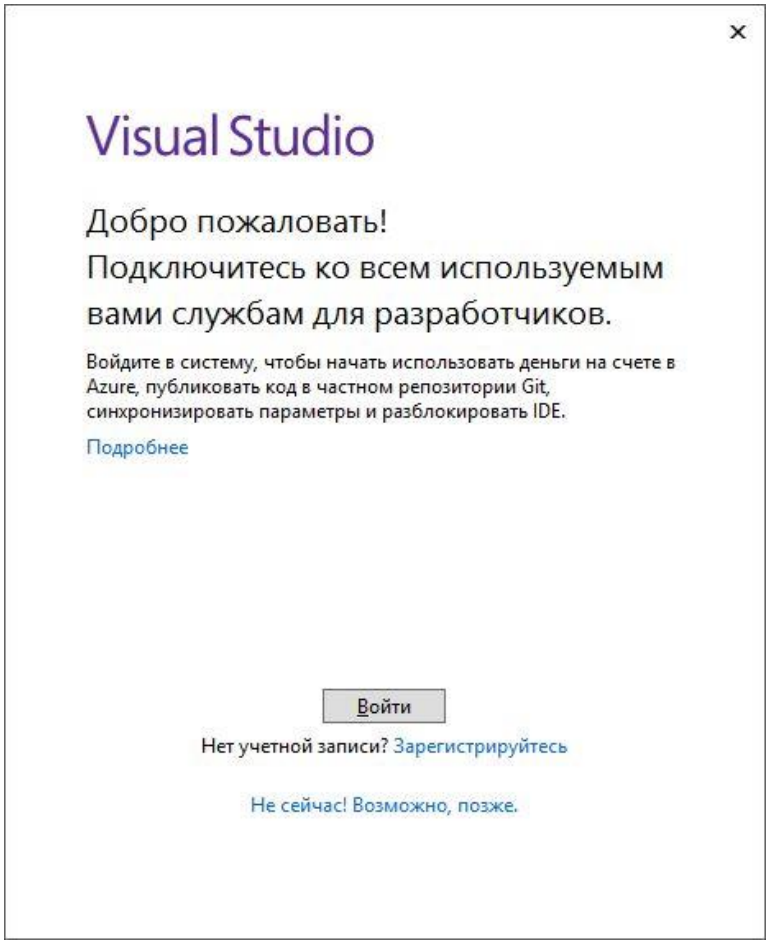


Рис. 4.1. Окно запуска Visual Studio 2017

Далее, если потребуется, то необходимо будет войти в учетную запись Microsoft. В случае если у вас нет учетной записи, то ее необходимо создать. После входа будет открыто окно программы.

Стартовое окно программы имеет следующий вид (рис. 4.2):

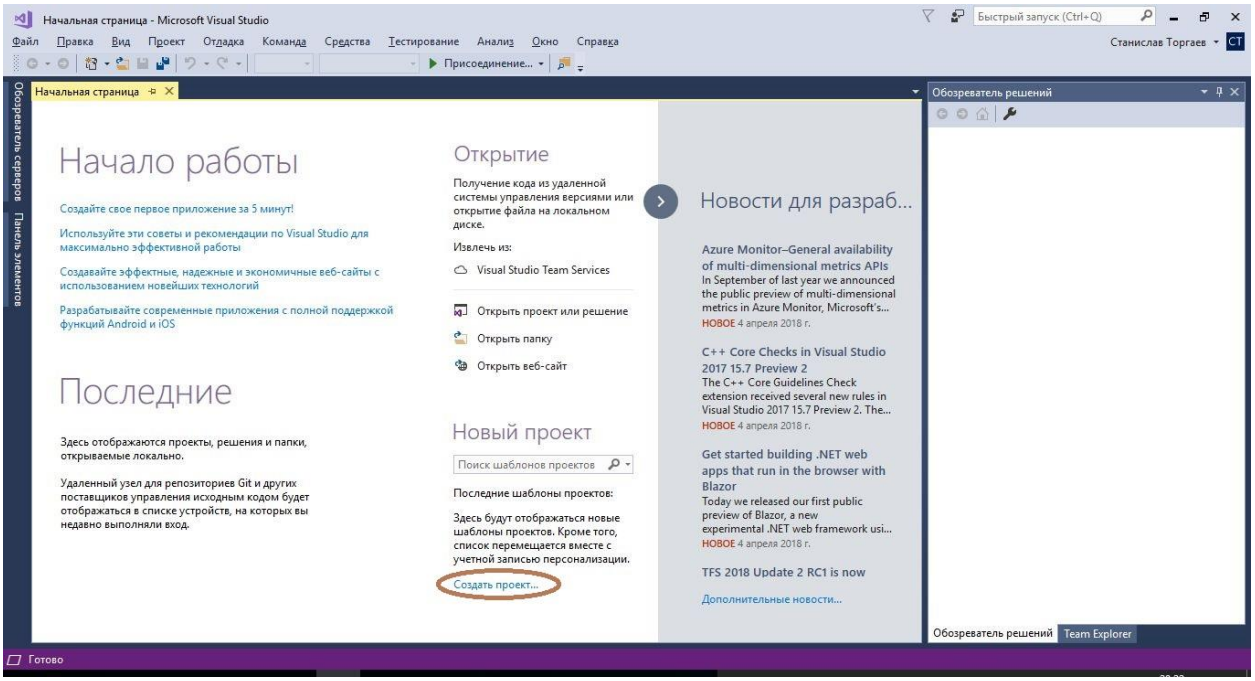


Рис. 4.2. Стартовое окно программы

Для создания нового проекта необходимо выбрать пункт "*Создать проект*". В открывшемся окне необходимо выполнить следующее:

1. Выбираем "*Шаблоны → Visual C++ → Общие*" (рис. 4.3).
2. В окне справа необходимо выбрать "*Пустой проект*" (рис. 4.3).
3. Далее необходимо определить имя проекта и место его расположения на компьютере. Желательно чтобы имя проекта состояло из латинских букв. Также допускается использование цифр (рис. 4.3). **Расположение проекта определяется преподавателем!!!**
4. Завершаем процесс создания проекта посредствам нажатия "*OK*".

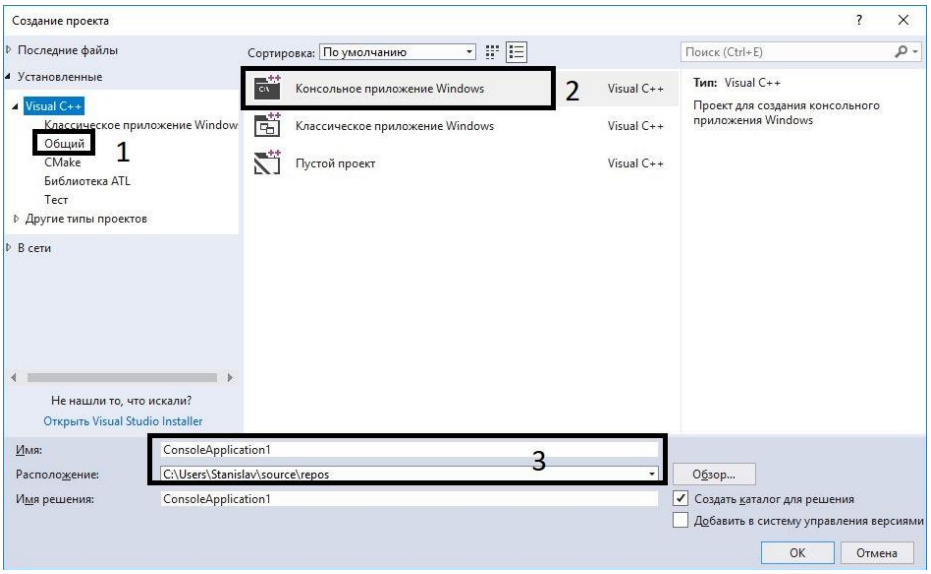


Рис. 4.3. Окно создания проекта

После создания проекта окно программы будет иметь следующий вид (рис. 4.4):

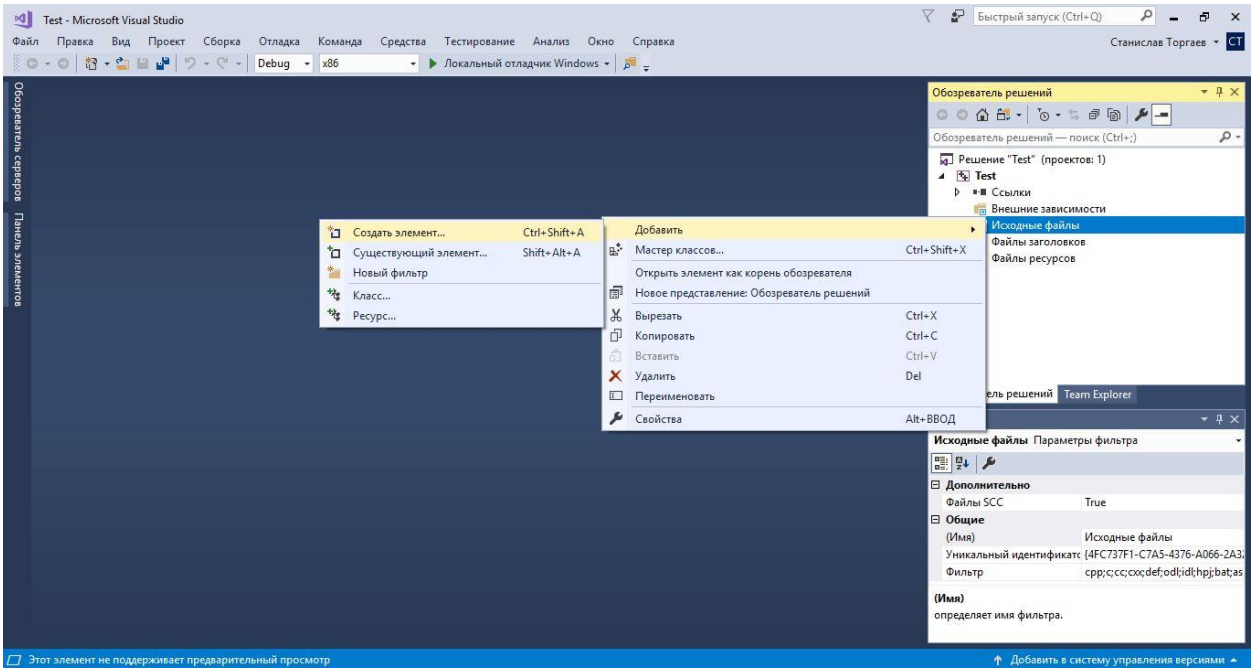


Рис. 4.4. Окно проекта

Для создания source-файла (файла в котором будет содержаться код программы) необходимо выполнить действия, представленные на рис. 4.4. В открывшемся окне необходимо:

1. Выбрать пункт “*Файл C++(.cpp)*” (рис. 4.5).
2. Определить имя файла (рис. 4.5).
3. Далее “*Добавить*”

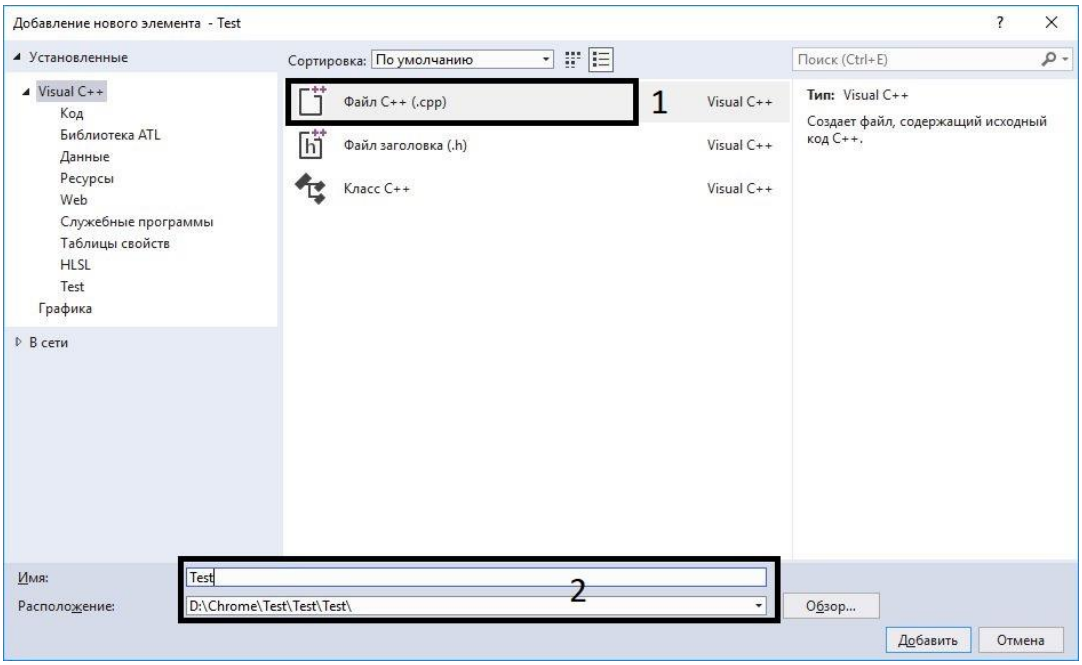


Рис. 4.5. Окно создания исходного файла

После добавление файла исходного кода окно программы будет иметь вид, представленный на рис. 4.6.

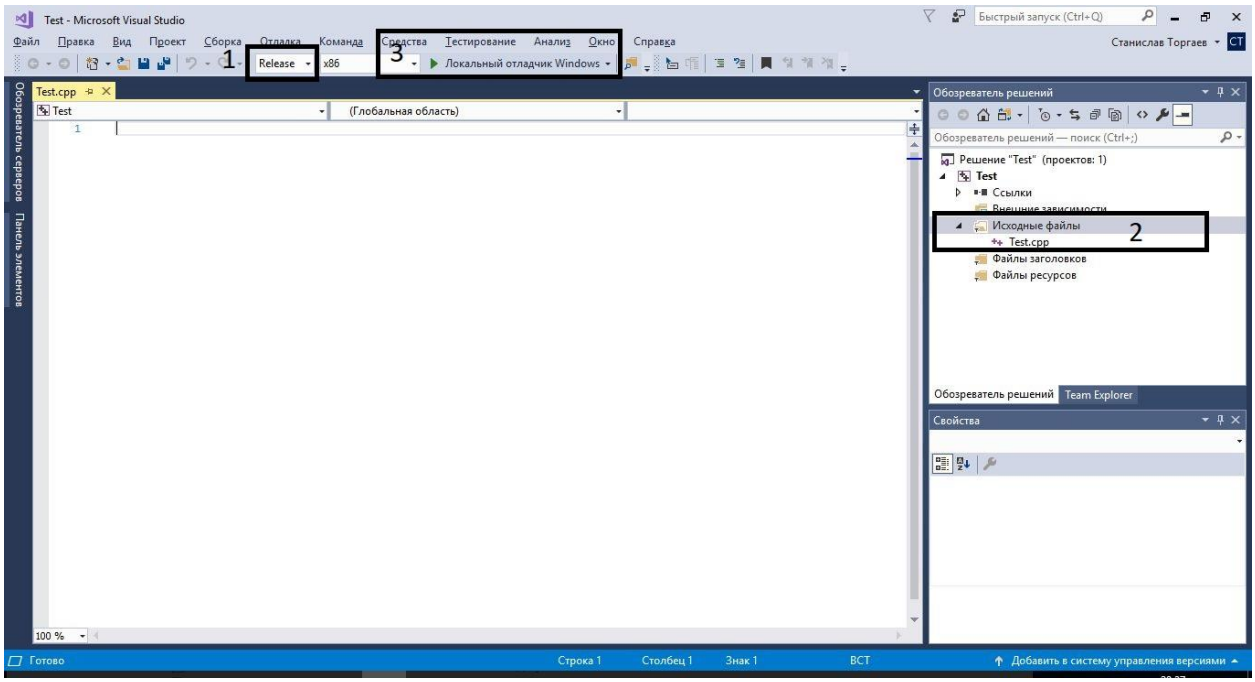


Рис. 4.6. Окно проекта

При этом в окне “*Обозреватель решений*” появится файл исходного кода (цифра 2 на рис. 4.6). В области отмеченной цифрой 1 (рис. 4.6) необходимо вместо “*Debug*” выбрать “*Release*”. Код программы записывается в основном окне проекта. После написания

программы необходимо выполнить компиляцию проекта (цифра 3 на рис. 4.6). После завершения компиляции, в случае отсутствия ошибок в программном коде, в исходной папке проекта появится папка “*Release*”, в которой будет сформирован файл имя которого соответствует имени файла исходного кода, но с расширением “*.exe*”. Этот файл и является соответствующей программой.

На рис. 4.7 представлен внешний вид окна программы с примером кода на Си. В данном случае программный код имеет следующий вид:

```
#include <stdio.h>                    //подключение заголовочного файла

int a;                                //объявление переменной

void main()                           //основной цикл
{                                     //начало основного цикла

}                                     //конец основного цикла
```

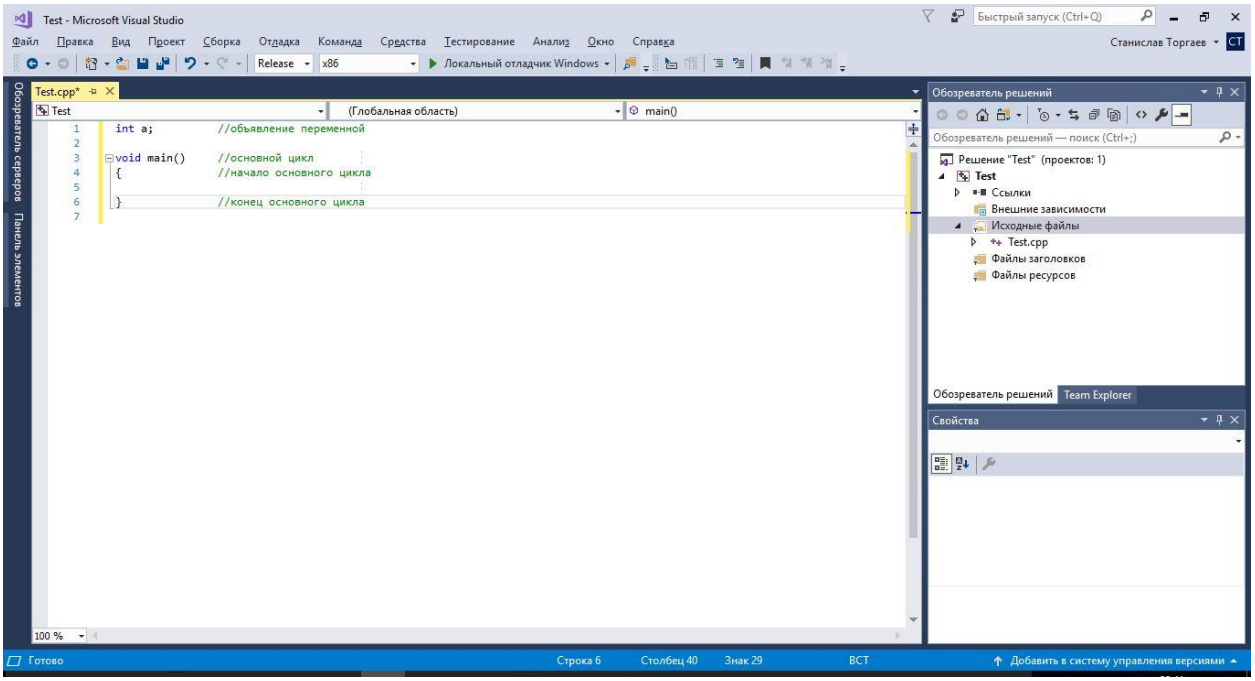


Рис. 4.7. Окно программы с кодом на Си

После написания программного кода в проекте необходимо выполнить компиляцию программы и сборку проекта. Компиляция программы позволить выявить ошибки в программном коде. Для того чтобы выполнить компиляцию и сборку необходимо на панели сверху нажать кнопку «Локальный отладчик Windows». В случае если в программе не было нижнее окно программы будет иметь следующий

вид в окне «Показать выходные данные от» должна быть выбрана «Сборка» (рис. 4.8):

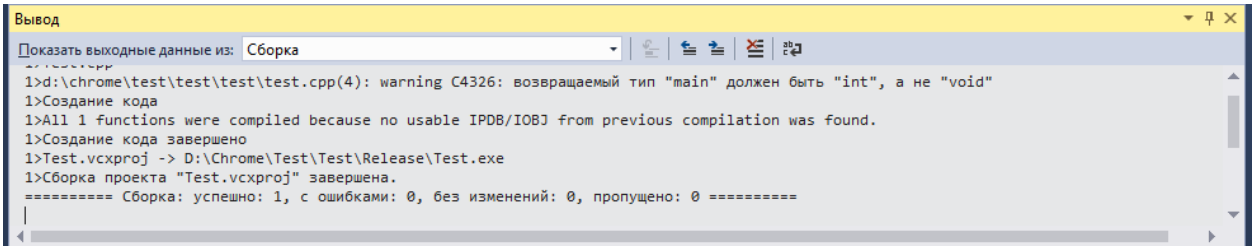


Рис. 4.8. Внешний вид нижнего окна программы после успешной сборки программы

В случае если в программном коде будет ошибка, то вид нижнего окна программы будет иметь следующий вид (рис. 4.9):

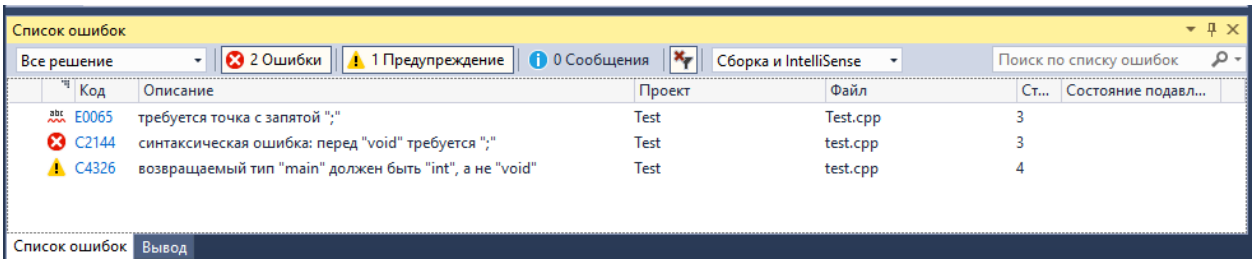


Рис. 4.9. Внешний вид нижнего окна программы после сборки программы с ошибками

Видно, что в программе были выявлены ошибки, список которых представлен в нижнем окне (рис. 4.9).

3. Программа работы

Выполните следующие пункты работы:

1. Откройте программу Visual Studio 2017 и создайте проект. Имя проекта должно иметь следующую структуру: LW1_Фамилия, при этом Фамилия должна быть написана латинскими буквами. Расположение папки проекта должно быть указано преподавателем.

2. После создания проекта и добавления файла исходного кода напишите следующую программу (включая комментарии) и откомпилируйте ее, убедившись в отсутствии ошибок:

```
#include <stdio.h>                    //подключение заголовочного файла

int a;                                //объявление переменной

void main()                           //основной цикл
{                                     //начало основного цикла
```

```
}                                   //конец основного цикла
```

3. Добавьте в данную программу глобальное объявление переменных в табл. 4.1 и выполните компиляцию на наличие ошибок.

Таблица 4.1.

Типы переменных

Имя переменных	Тип переменных
a,b	***integer***
c	***double***
d,e	***float***
letter, digits	***char***
a1,b1,c1,d1	***signed integer***
a2, b2,c2,d2	***long integer***

В случае наличия ошибок попробуйте исправить их и вновь выполнить компиляцию. При этом необходимо внести комментарии в строки объявления переменных с указанием их типов.

4. Добавить в программу глобальное определение констант из табл. 4.2, посредствам, соответствующим им идентификаторов.

Таблица 4.2.

Константы

Имя констант	Значения	Тип идентификатора
min	60	***define***
max	100	***define***
avr	80	***define***
con1	10	***const***
con2	1.53	***const***
con3	-20	***const***

В случае наличия ошибок попробуйте исправить их и вновь выполнить компиляцию. При этом необходимо внести комментарии в строки объявления констант.

5. Запишите следующую программу в том виде в котором Вы ее видите ниже (не внося исправлений):

```
#include <stdio.h>                  //подключение заголовочного файла

int a;                              //объявление переменной

void main()                         //основной цикл
{                                   //начало основного цикла
```

```
a=5                    //присвоение переменной a значения 5
b=10                   //присвоение переменной b значения 10
c=a+b                  //присвоение переменной c значения a+b
}                      //конец основного цикла
```

Откомпилируйте данную программу. После компиляции будут обнаружены следующие ошибки:

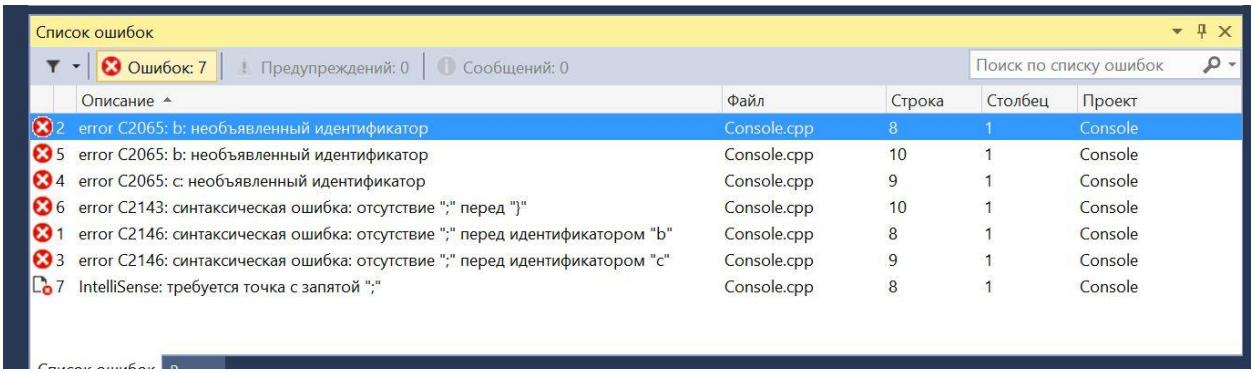


Рис. 4.10. Внешний вид нижнего окна программы после сборки программы с ошибками

Исправьте ошибки и добейтесь корректной компиляции проекта!!! Покажите результат исправленной программы преподавателю.

Лабораторная работа №2

Ввод-вывод данных в языке Си с использованием консоли.

1. Цель работы

Целью работы является изучение ввода и вывода данных с использованием консоли.

2. Краткое описание функций ввода-вывода данных

Функция ввода *getchar()*.

Функция ***getchar()*** при каждом к ней обращении возвращает следующий вводимый символ. При этом принимаемый символ будет записан в ASCII коде.

Функция вывода *printf()*.

Для вывода в консоль информации используется функция ***printf ()*.** Она переводит данные в символьное представление и выводит полученные изображения символов на экран. При этом у пользователя имеется возможность форматировать данные, то есть влиять на их представление на экране.

Общая форма записи функции ***printf()*** выглядит следующим образом:

***printf*("Строка Форматов", объект1, объект2, ..., объектn)**

Строка Форматов состоит из следующих элементов:

- управляющих символов;
- текста, представленного для непосредственного вывода;
- форматов, предназначенных для вывода значений переменных различных типов.

Управляющие символы не выводятся на экран, а управляют расположением выводимых символов. Отличительной чертой управляющего символа является наличие обратного слэша '\' перед ним.

При выводе в консоль существуют следующие основные управляющие символы:

'\n' – перевод строки;

'\t' – горизонтальная табуляция;

'\v' – вертикальная табуляция;

'\b' – возврат на символ;

'\r' – возврат на начало строки;

'\a' – звуковой сигнал.

Форматы нужны для того, чтобы указывать вид, в котором информация будет выведена на экран. Отличительной чертой формата является наличие символа процент *'%'* перед ним:

%d – целое число типа ***int*** со знаком в десятичной системе счисления;

%u – целое число типа ***unsigned int***;

%x – целое число типа ***int*** со знаком в шестнадцатеричной системе счисления;

%o – целое число типа ***int*** со знаком в восьмеричной системе счисления;

%hd – целое число типа ***short*** со знаком в десятичной системе счисления;

%hu – целое число типа ***unsigned short***;

%hx – целое число типа ***short*** со знаком в шестнадцатеричной системе счисления;

%ld – целое число типа ***long int*** со знаком в десятичной системе счисления;

%lu – целое число типа ***unsigned long int***;

%lx – целое число типа ***long int*** со знаком в шестнадцатеричной системе счисления;

%f – вещественный формат (числа с плавающей точкой типа ***float***);

%lf – вещественный формат двойной точности (числа с плавающей точкой типа ***double***);

%e – вещественный формат в экспоненциальной форме (числа с плавающей точкой типа ***float*** в экспоненциальной форме);

%c – символьный формат;

%s – строковый формат.

После использования функции ***printf*** окно консоли автоматически закрывается. Для того, чтобы это не происходило, можно после данной функции использовать функцию ***getchar***. Причем если в программе применяется ни одна функция вывода, то необходимо в конце программы прописывать столько же функций ***getchar***, т.е. команда закрытия консоли генерируется в конце каждой функции вывода в консоль. При этом если используется одна функция ***printf*** для вывода нескольких переменных, то количество функций ***getchar*** должно соответствовать количеству выводимых переменных.

Функция ввода *scanf_s*.

Функция форматированного ввода данных с клавиатуры ***scanf_s()*** выполняет чтение данных, вводимых с клавиатуры, преобразует их во

внутренний формат и передает вызывающей функции. Общая форма записи функции ***scanf_s()***:

***scanf_s* ("СтрокаФорматов", адрес1, адрес2,...);**

Строка форматов аналогична функции ***printf()***. Для формирования адреса переменной используется символ амперсанд '**&**': **адрес = &объект**. Строка форматов и список аргументов для функции обязательны. Окончанием считывания с консоли будет символ нажатия клавиши ***Enter*.**

Функция ***scanf_s()*** может работать сразу с несколькими переменными. Предположим, что необходимо ввести два целых числа с клавиатуры. Формально для этого можно дважды вызвать функцию ***scanf_s(),*** однако лучше воспользоваться такой конструкцией:

***scanf_s*(" *%d, %d* ", *&n, &m*);**

Функция ***scanf_s()*** интерпретирует это так, как будто ожидает, что пользователь введет число, затем – запятую, а затем – второе число. Все происходит так, как будто требуется ввести два целых числа следующим образом:

88,221 или 88, 221

Функция ***scanf_s()*** возвращает число успешно считанных элементов.

Функция *get_s*.

Для считывания строки символов, например, какого-либо предложения или слова, необходимо задать следующую переменную:

char имя_переменной[***N***];

где ***N*** – количество символов в строке. При этом количество символов можно задавать больше, в этом случае просто будет зарезервировано больше ячеек памяти. В ином случае можно потерять часть вводимой информации. Считывание строки будет осуществляться с помощью функции:

get_s (имя_переменной);

В табл. 4.3 приведены примеры команд ввода и вывода различных данных.

Таблица 4.3.

Примеры команд ввода и вывода

Синтаксис	Комментарии
getchar();	Функция ожидания ввода символа с клавиатуры
y=getchar();	Ожидание ввода в консоль одного символа с клавиатуры и сохранение ASCI кода этого символа в переменную

	***y*.**
printf("Информационные технологии***n**");	Вывод в консоль фразы «Информационные технологии».
printf("a=%d\n", a);	Вывод в консоль текста «***a*=**» и далее выводится значение переменной **a**. За счет использования формата ***%d*** значение переменной будет выводиться в десятичной форме. За счет использования управляющего символа ***n*** курсор будет находиться на следующей строке.
printf("b=%f\n", b);	Вывод в консоль текста «***b*=**» и далее выводится значение переменной ***a***. За счет использования формата ***%f*** значение переменной будет выводиться в форме числа с плавающей точкой. За счет использования управляющего символа ***n*** курсор будет находиться на следующей строке.
printf("c=%10.5f\n", c);	Вывод в консоль текста «**c=**» и далее выводится значение переменной **c**. За счет использования формата ***%f*** значение переменной будет выводиться в форме числа с плавающей точкой. За счет использования управляющего символа ***n*** курсор будет находиться на следующей строке. В приведенном примере 10 — общее количество знакомест под значение переменной; 5 — количество позиций после десятичной точки. В указанном примере количество знакомест в выводимом числе меньше 10, поэтому свободные знакоместа слева от числа заполняются пробелами. Такой способ форматирования часто используется для построения таблиц.
scanf_s("%d", &y);	Чтение символов, введенных с клавиатуры и преобразование их в десятичный формат. Далее передача полученного числа в переменную ***y*.**
scanf_s("%f", &z);	Чтение символов, введенных с клавиатуры и преобразование их в формат числа с плавающей точкой. Далее передача полученного числа в переменную **z.**
scanf_s("%d, %c,%f", &a,&b,&c);	Чтение нескольких символов,

	введенных с клавиатуры с помощью одной функции и преобразование их в соответствующий формат.

get_s(a)	Считывание нескольких символов (строки) в переменную «***a***». При этом переменная должна быть объявлена как ***char*** и иметь формат строки, например, ***char a*[20]**. В данном случае переменная представляет собой набор из двадцати символов – ***a*[0], *a*[1], *a*[2]** и т.д. Однако при выводе строки, которая сохранена в данную переменную к ней можно обращаться без указания размера – **a.** Пример:
	char a[20];
	gets_s(a);
	printf("%s\n", a);

3. Программа работы

Выполните следующие пункты работы:

1. Написать код программы считывания двух символов с помощью функции ***getchar()*** и вывода на экран ASCII кодов символов в десятичной и символьной формах. Проверить правильность ее работы (табл. 4.4).

Формат ввода-вывода данных:
«Введите два символа:»
«Первый символ:»
«Второй символ:»
«ASCII код 1 символа:»
«ASCII код 2 символа:»

Таблица 4.4.

Варианты заданий

Вариант	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
Символы	«A» «1»	«2» «B»	«3» «4»	«F» «5»	«S» «6»	«T» «R»	«r» «9»	«s» «m»	«P» «1»
Вариант	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**
Символы	«1» «5»	«2» «b»	«y» «4»	«F» «n»	«f» «u»	«t» «e»	«r» «0»	«s» «z»	«d» «7»

На рис. 4.11 представлен пример ввода-вывода.

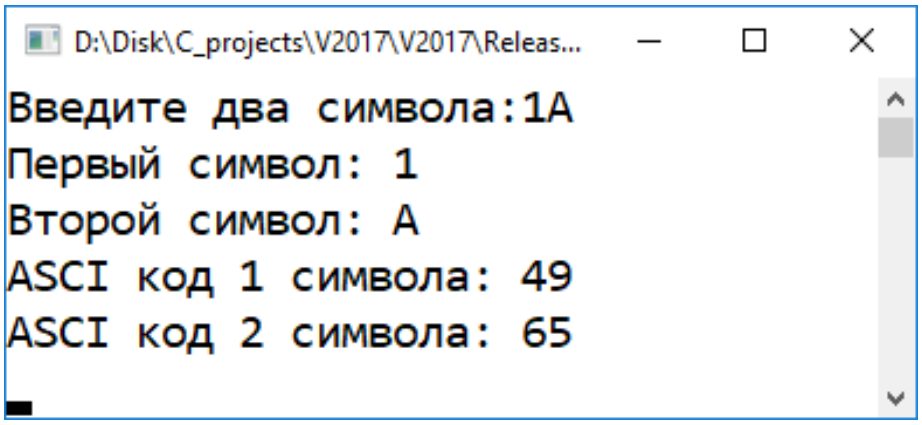


Рис. 4.11. Внешний вид окна ввода-вывода

2. Написать код программы считывания двух символов с помощью функции ***getchar()*** и вывода на экран ASCII кодов этих символов в десятичной форме в строку через запятую (табл. 4.5). Проверить правильность ее работы.

Формат ввода-вывода данных:
«Введите два символа:»
«ASCII коды символов:»

Таблица 4.5.

Варианты заданий

Вариант	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
Символы	«A» «1»	«2» «B»	«3» «4»	«F» «5»	«S» «6»	«T» «R»	«r» «9»	«s» «m»	«P» «1»
Вариант	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**
Символы	«1» «5»	«2» «b»	«y» «4»	«F» «n»	«f» «u»	«t» «e»	«r» «0»	«s» «z»	«d» «7»

На рис. 4.12 представлен пример ввода-вывода.

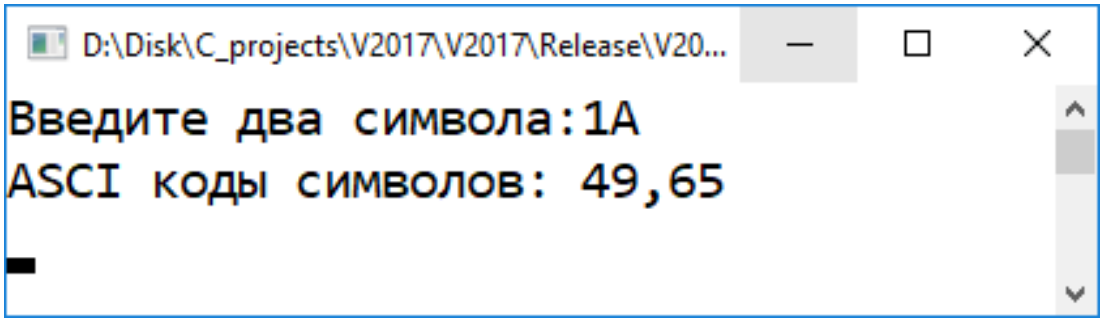


Рис. 4.12. Внешний вид окна ввода-вывода

3. Написать код программы вывода Вашего года рождения и возраста с помощью функции ***scanf_s()***. Проверить правильность ее работы.

Формат ввода-вывода данных:
Ввод:
«Введите г.р. и возраст:»
Вывод:
«Иванов Иван Иванович»
«Год рождения:»
«Возраст:»
Пример выполненного кода (рис. 4.13):

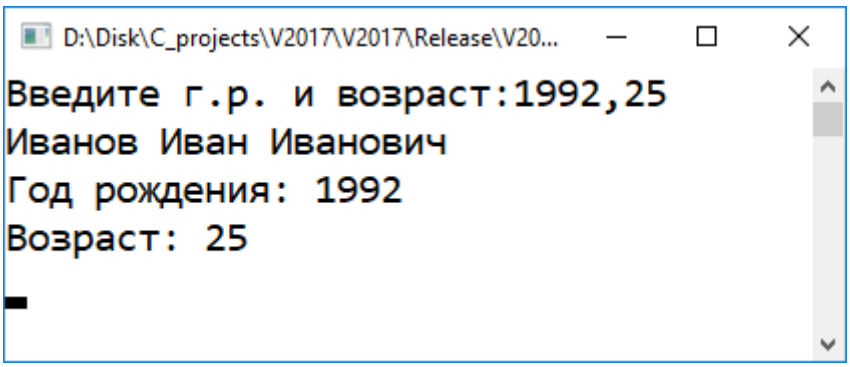


Рис. 4.13. Внешний вид окна ввода-вывода

4. Написать код программы ввода одного числа с плавающей точкой, одного целого числа, одного отрицательного числа с точкой, одного отрицательного целого числа и вывода их на экран с помощью функции ***scanf_s()***. Проверить правильность ее работы.

Формат ввода-вывода данных:
Ввод:
«Введите целое число:»
«Введите число с точкой:»
«Введите отрицательное число с точкой:»
«Введите целое отрицательное число:»
Вывод:
«Вы ввели следующие числа:»
«Целое число»
«Число с точкой:»
«Отрицательное число с точкой:»
«Отрицательное целое число:»
Пример выполненного кода (рис. 4.14):

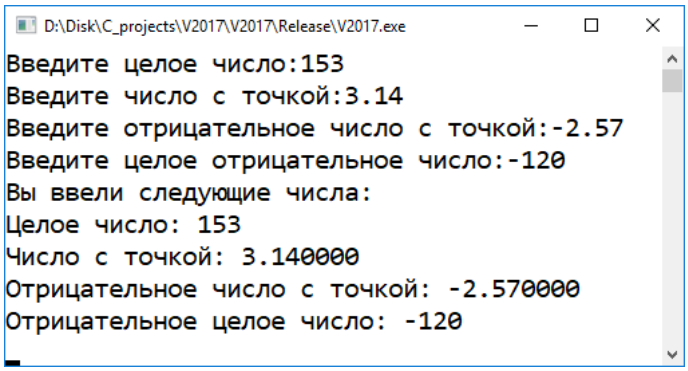


Рис. 4.14. Внешний вид окна ввода-вывода

Вводимые числа могут быть любые!!!

5. Написать код программы ввода одного числа с плавающей точкой, одного целого числа, одного отрицательного числа с точкой, одного отрицательного целого числа и вывода их на экран с помощью функции ***scanf_s()***. В отличие от предыдущего задания на экране не должно быть строк ввода, т.е. необходимо производить отчистку консоли после каждого ввода. Проверить правильность ее работы.

Формат ввода-вывода данных:

Ввод:

«Введите целое число:»

«Введите число с точкой:»

«Введите отрицательное число с точкой:»

«Введите целое отрицательное число:»

Вывод:

«Вы ввели следующие числа:»

«Целое число»

«Число с точкой:»

«Отрицательное число с точкой:»

«Отрицательное целое число:»

Пример выполненного кода (рис. 4.15):

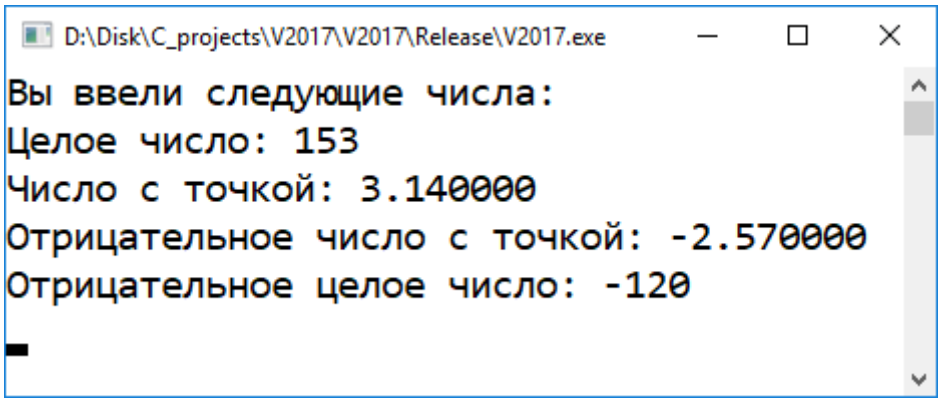


Рис. 4.15. Внешний вид окна ввода-вывода

Вводимые числа могут быть любые!!!

6. Написать код программы ввода информации о студенте с функцией отчистки экрана после каждого ввода. Далее необходимо вывести все данные на экран. Для ввода текстовых данных необходимо использовать функцию ***gets_s()***. Проверить правильность ее работы.

Формат ввода-вывода данных:

Ввод:
«Введите фамилию:»
«Введите имя:»
«Введите отчество:»
«Введите год рождения:»
«Введите Ваш возраст:»
«Введите Ваш родной город:»
«Введите номер школы:»

Пример выполненного кода (рис. 4.16):

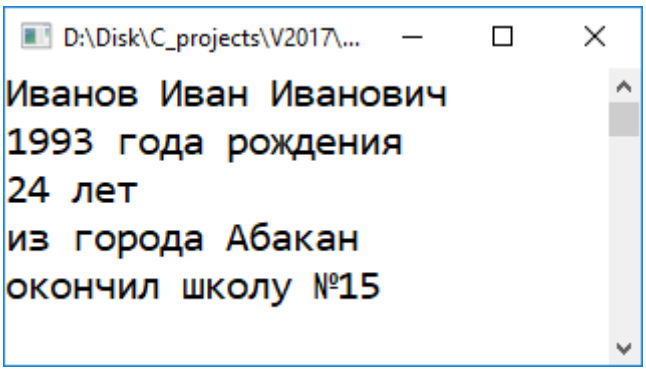


Рис. 4.16. Внешний вид окна ввода-вывода

Лабораторная работа №3

Основные операции языка Си.

1. Цель работы

Целью работы является изучение основных операций языка Си.

2. Программа работы

Выполните следующие пункты работы:

1. Написать код программы считывания двух целых чисел, введенных с клавиатуры, и вывода их суммы и разницы на экран. Экран консоли необходимо очищать после ввода каждого числа. Проверить правильность ее работы с различными вариантами чисел.

Формат ввода-вывода данных:

Ввод:

«Введите первое число:»

«Введите второе число:»

Вывод:

«Сумма чисел:»

«Разница чисел:»

На рис. 4.17 представлен пример ввода-вывода (в данном случае были введены числа 153 и 21).

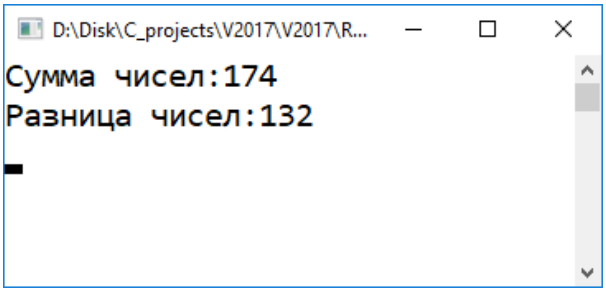


Рис. 4.17. Внешний вид окна ввода-вывода для чисел 153 и 21

На рис. 4.18 представлен пример ввода-вывода (в данном случае были введены числа 100 и -50).

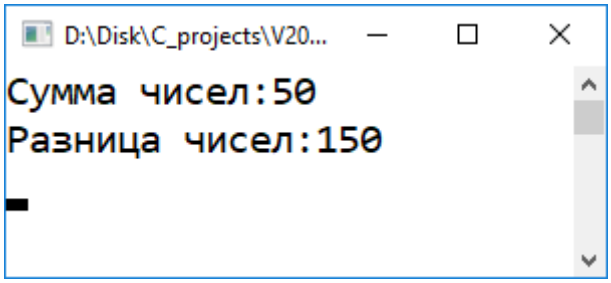


Рис. 4.18. Внешний вид окна ввода-вывода для чисел 100 и -50

2. Написать код программы считывания двух целых чисел, введенных с клавиатуры, и вывода их произведения, целой части от деления и остатка на экран. Экран консоли необходимо очищать после ввода каждого числа. Проверить правильность ее работы с различными вариантами чисел.

Формат ввода-вывода данных:

Ввод:

«Введите первое число:»

«Введите второе число:»

Вывод:

«Произведение чисел:»

«Целая часть от деления:»

«Остаток от деления:»

На рис. 4.19 представлен пример ввода-вывода (в данном случае были введены числа 53 и 9).

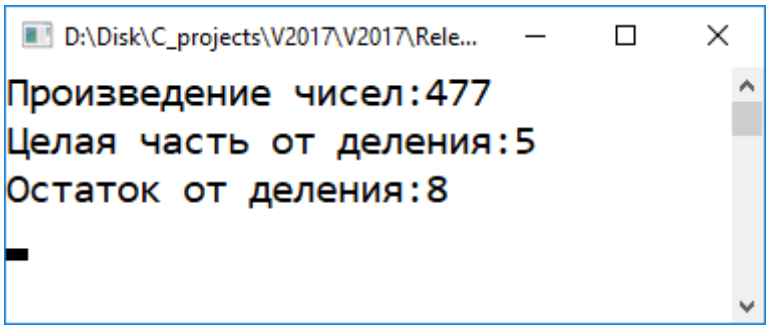


Рис. 4.19. Внешний вид окна ввода-вывода для чисел 53 и 9

3. Написать код программы считывания трех целых чисел, введенных с клавиатуры, и вывода их на экран (табл. 4.6). А также реализующее математическое выражение с выводом результата. Экран консоли необходимо очищать после ввода каждого числа. Проверить правильность ее работы.

Формат ввода-вывода данных:

Ввод:

«Введите первое число:»

«Введите второе число:»

«Введите третье число:»

Вывод:

«выражение = »

Таблица 4.6.

Варианты заданий

Вариант	Имя переменных	Значения	Выражение
1	***a, b, c***	1.25, 3.3, 10	(a+b*c)(a-b/c)
2	***x, y, z***	5.5, 3.4, 7	x*y*z-x*y-x/z
3	***n, m, r***	275, 0.37, 26.7	10*m+(m*r-n*r)
4	***a1, a2, a3***	-9.4, 15, 9.4	5*a1+5*a1*a2+(a1+a3)
5	***b1, b2, b3***	890, -1.45, 4	b1/100+(b2+b3)*6-b3*5
6	***p, q, w***	-3, 0.12, 0.13	(p+3)*(q+7)*(w-8)*5
7	***s, v, t***	68.75, 247,-9	(s+v)/5.65-(s/11.3+t)*1.51+21
8	***c1, c2, c3***	34, -2345.23, -2	c2/(c1*c3+c1+4.64*c3)*c3
9	***z, v, r***	934, -23, 0.1	(z*r+v)*82.6-v/r*v
10	***x, o, o1***	0.234, -4.7, -56	x*x/(o+6.12*x*o1)+2*o1
11	***n, k, r***	6.97, -67, 0.76	(1.56*n+15.6*k+156*r)*(n+k)*8
12	***p1, q1, w1***	90, 0.002, -8	p1+w1*748.675*q1-(w1+p1\10)

На рис. 4.20 представлен пример для переменных ***a***3=1.75, ***b***1=5.86, ***c***1=-10, реализующее выражение ***rez***=***a***3*(***a***1+***b***1-***c***1/5)*c1.

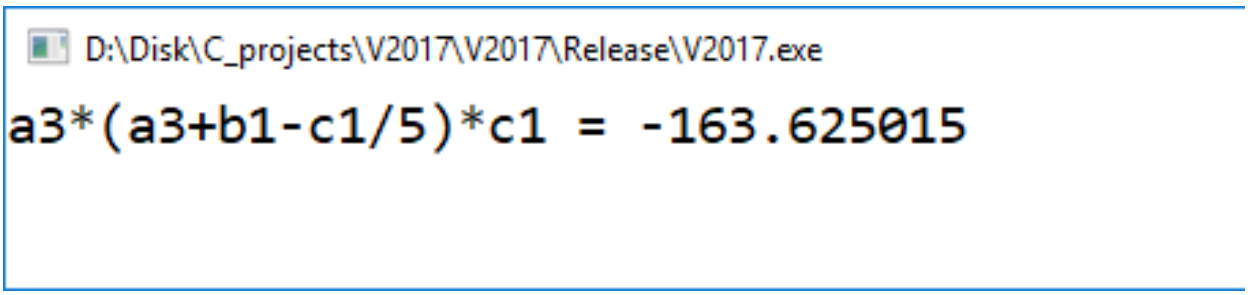


Рис. 4.20. Внешний вид окна ввода-вывода для выражения rez=a3(a1+b1-c1/5)*c1*

4. Написать код программы вывода на экран периметра и площади фигур в соответствии с вариантом (табл. 4.7). Экран консоли

необходимо очищать после ввода каждого числа. Проверить правильность ее работы с различными вариантами чисел.

Таблица 4.7.

Варианты заданий

Вариант	**Фигура**	**Формат ввода-вывода**
1	Квадрат	Ввод: «Введите сторону квадрата:» Вывод: «Периметр квадрата = » «Площадь квадрата = »
2	Прямоугольник	Ввод: «Введите первую сторону:» «Введите вторую сторону:» Вывод: «Периметр прямоугольника = » «Площадь прямоугольника = »
3	Прямоугольный треугольник	Ввод: «Введите первый катет:» «Введите второй катет:» «Введите гипотенузу:» Вывод: «Периметр треугольника = » «Площадь треугольника = »
4	Окружность	Ввод: «Введите радиус:» Вывод: «Периметр окружности = » «Площадь окружности = »

На рис. 4.21 представлен пример выходного окна для окружности радиусом 5.

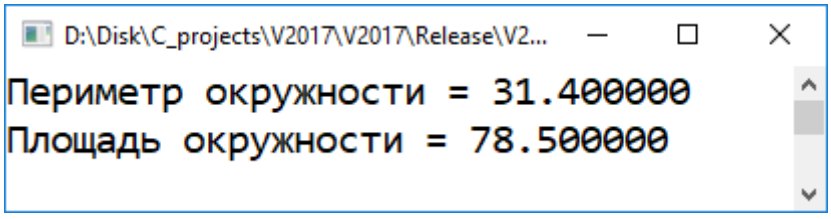


Рис. 4.21. Внешний вид окна ввода-вывода для окружности радиусом 5

5. Написать код программы ввода пяти чисел и вывода на экран среднего значения. Проверить правильность ее работы с различными вариантами чисел.

Пример ввода-вывода показан на рис. 4.22.

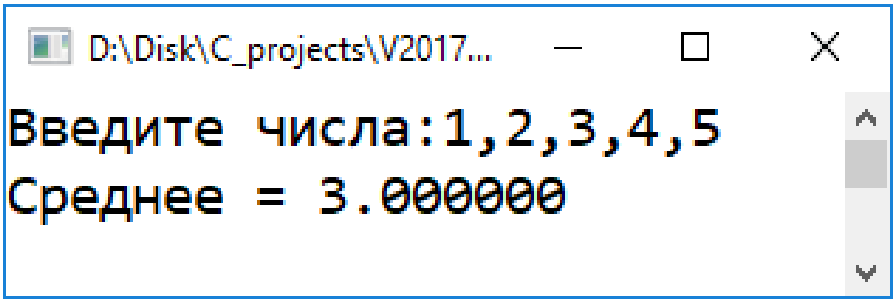


Рис. 4.22. Внешний вид окна ввода

6. Написать код программы, которая выводит на экран результаты логических и арифметических операций двух переменных, заданных программно (табл. 4.8). Проверить правильность работы.

Таблица 4.8.

Варианты заданий

Вариант	**Имя переменных и значения**
1	***a***=0xAA ***b***=0xFF
2	***a***=0xF1 ***b***=0x77
3	***a***=0xAA ***b***=0x11
4	***a***=0x33 ***b***=0xBB
5	***a***=0x55 ***b***=0x33
6	***a***=0x88 ***b***=0xFF

Операции, которые необходимо выполнить:
Логическое умножение ***a*** и ***b***;
Логическое сложение ***a*** и ***b***;
Логическое отрицание переменных ***a*** и ***b***;
Исключающее или ***a*** и ***b***;
Сдвиг влево переменной a и сдвиг вправо переменной ***b***;
Операцию: ***c***=***a***+ ++***b*** – при этом вывести на экран ***c***, ***a***, ***b***

Операцию: ***d***=***a***+ ***b***++ – при этом вывести на экран ***d***, ***a***, ***b***

Пример вывода для переменных ***a***=0xAA и ***b***=0x77 показан на рис. 4.23.

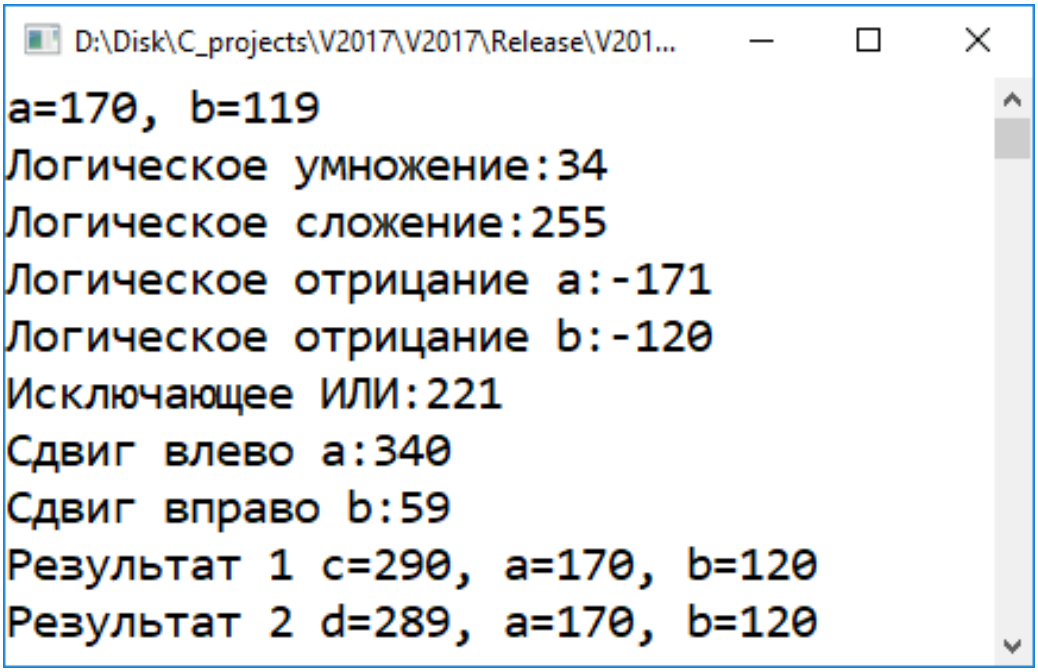


Рис. 4.23. Внешний вид окна ввода

7. Написать код программы, которая выводит на экран результаты логических и арифметических операций двух переменных, заданных программно (табл. 4.9). Проверить правильность работы.

Таблица 4.9.

Варианты заданий

Вариант	**Имя переменных и значения**
1	***a***=0xAA ***b***=0xFF
2	***a***=0xF1 ***b***=0x77
3	***a***=0xAA ***b***=0x11
4	***a***=0x33 ***b***=0xBB
5	***a***=0x55 ***b***=0x33
6	***a***=0x88

	b=0xFF

Операции, которые необходимо выполнить:

1. ***a=a&b;***
2. ***a&=b;***
3. ***a=a|b;***
4. ***a|=b;***
5. ***a=a+b;***
6. ***a+=b;***
7. ***a=a*b;***
8. ***a*=b;***

На экран необходимо вывести значение переменной a после выполнения каждой операции. Пример окна представлен для переменных ***a***=0xAA и ***b***=0x77 показан на рис. 4.24.

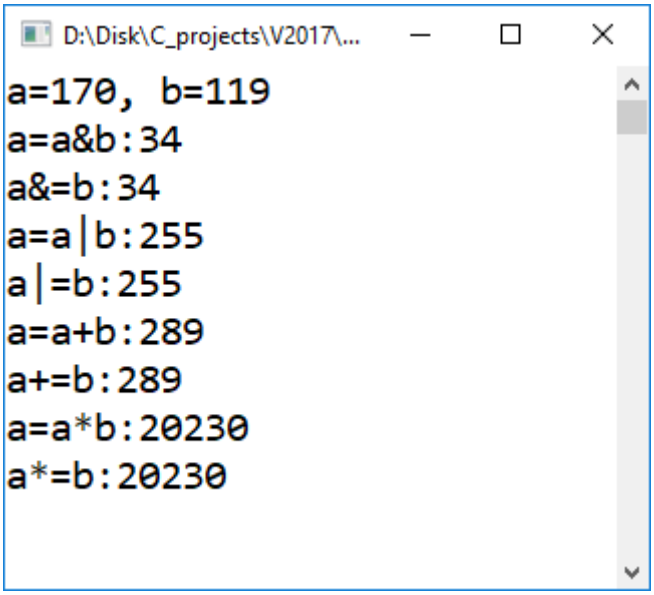


Рис. 4.24. Внешний вид окна ввода при a=0xAA и b=0x77

Лабораторная работа №4

Операторы цикла

1. Цель работы

Целью работы является изучение операторов цикла языка Си.

2. Программа работы

Выполните следующие пункты работы:

1. Написать код расчета суммы чисел от ***n*** до ***N*** (в соответствии с вариантом задания, табл. 4.10). Результат вывести на экран. Проверить правильность ее работы. Использовать оператор цикла ***for***.

Таблица 4.10.

Варианты заданий

	1	2	3	4	5	6	7	8	9	10	11	12
n	5	3	10	20	11	15	100	101	31	45	67	90
N	50	13	100	200	110	150	1000	150	70	56	93	1090

На рис. 4.25 представлен пример вывода (в данном случае были числа от 40 до 50).

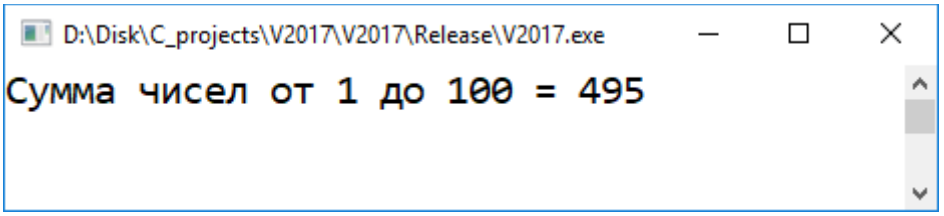


Рис. 4.25. Внешний вид окна вывода для чисел от 40 до 50

2. Написать код расчета суммы чисел от ***n*** до ***N***. Числа ***n*** и ***N*** должны вводиться с клавиатуры. Результат вывести на экран. Проверить правильность ее работы. Использовать оператор цикла ***for***.

На рис. 4.26 представлен пример ввода-вывода.

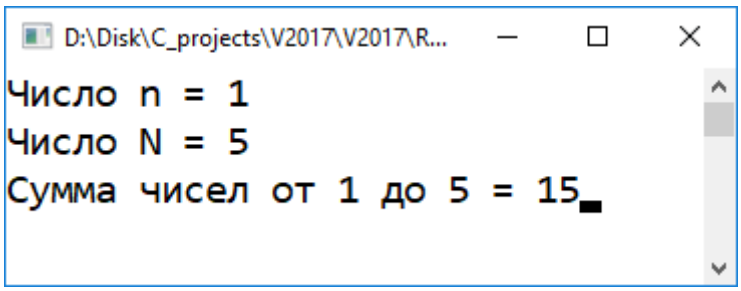


Рис. 4.26. Внешний вид окна ввода-вывода

3. Написать код расчета суммы чисел от ***n*** до ***N*** (в соответствии с вариантом задания, табл. 4.11). Результат вывести на экран. Проверить правильность ее работы. Использовать оператор цикла ***while***.

Таблица 4.11.

Варианты заданий

	1	2	3	4	5	6	7	8	9	10	11	12
n	5	3	10	20	11	15	100	101	31	45	67	90
N	50	13	100	200	110	150	1000	150	70	56	93	1090

На рис. 4.27 представлен пример вывода (в данном случае были числа от 40 до 50).

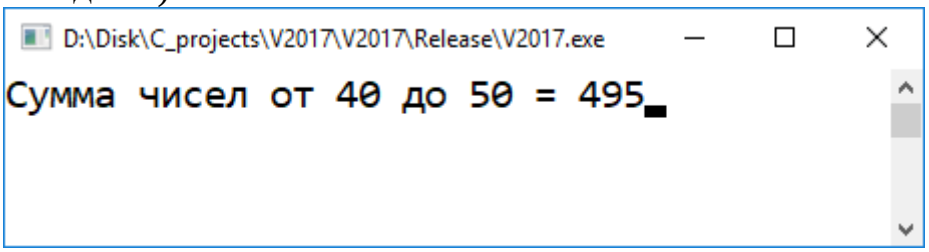


Рис. 4.27. Внешний вид окна вывода для чисел от 40 до 50

4. Написать код расчета суммы чисел от ***n*** до ***N***. Числа ***n*** и ***N*** должны вводиться с клавиатуры. Результат вывести на экран. Проверить правильность ее работы. Использовать оператор цикла ***while***.

На рис. 4.28 представлен пример ввода-вывода.

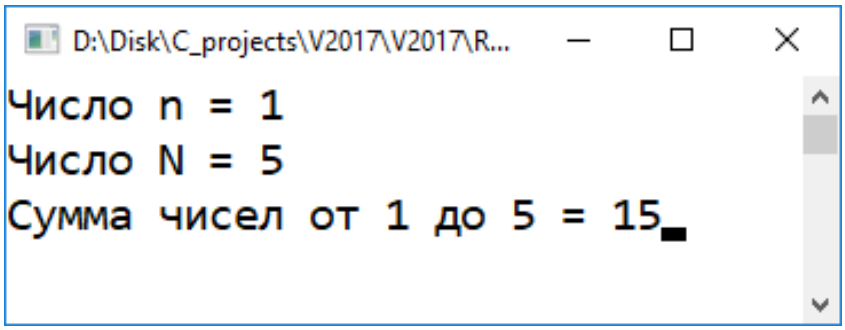


Рис. 4.28. Внешний вид окна ввода-вывода

5. Написать программу вывода на экран всех числе от 0 до 100 удовлетворяющих условию в соответствии с вариантом (табл. 4.12). Проверить правильность ее работы. Использовать любой операторы цикла ***for*** или ***while***.

Таблица 4.12.

Варианты заданий

Вариант	**Условие**
1	Все числа кратные 3

2	Все числа кратные 4
3	Все числа кратные 5
4	Все числа кратные 6
5	Все числа кратные 7
6	Все числа кратные 8

На рис. 4.29 представлен пример вывода всех чисел кратных 9:

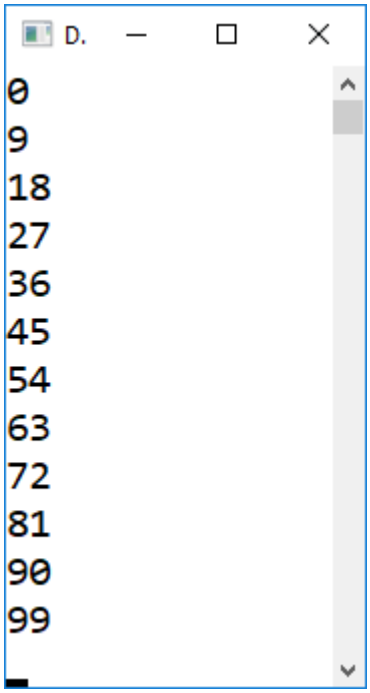


Рис. 4.29. Внешний вид окна вывода

Лабораторная работа №5

Операторы условных и безусловных переходов

1. Цель работы

Целью работы является изучение операторов условных и безусловных переходов языка Си.

2. Программа работы

Выполните следующие пункты работы:

1. Написать код программы сравнения трех целых чисел, введенных с клавиатуры, и вывода на экран максимального числа.

На рис. 4.30 представлен пример ввода-вывода.

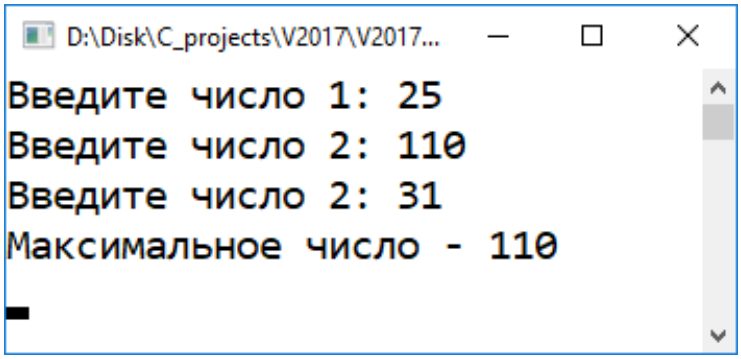


Рис. 4.30. Внешний вид окна ввода-вывода

2. Написать код программы сравнения трех целых чисел, введенных с клавиатуры, и вывода на экран минимального числа.

На рис. 4.31 представлен пример ввода-вывода.

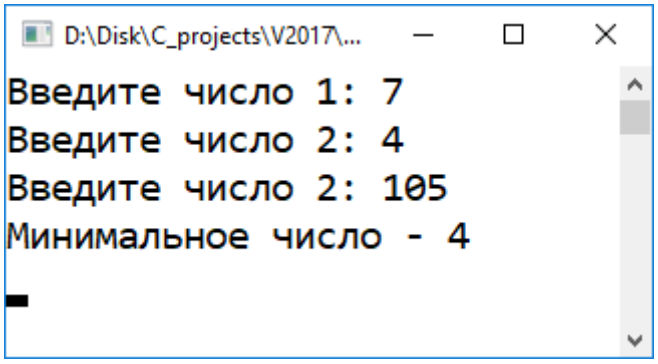


Рис. 4.31. Внешний вид окна ввода-вывода

3. Написать программу, которая выводит на экран оценку студента (отлично, хорошо, удовлетворительно, неудовлетворительно) в зависимости от введенных баллов.

Пример ввода/вывода представлен на рис. 4.32.

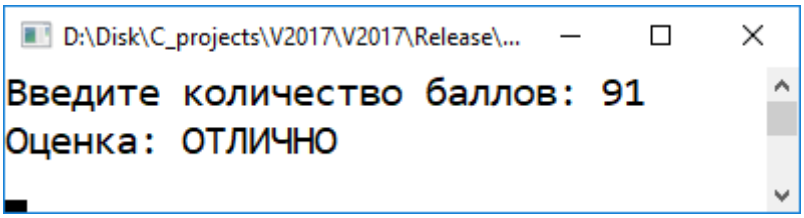


Рис. 4.32. Внешний вид окна ввода-вывода

4. Используя оператор ***switch***, написать код программы вывода на экран значения переменной от 1 до 5. В случае если переменная не входит в данный диапазон, то выводится фраза «Другое значение!!!».

Пример ввода/вывода представлен на рис. 4.33.

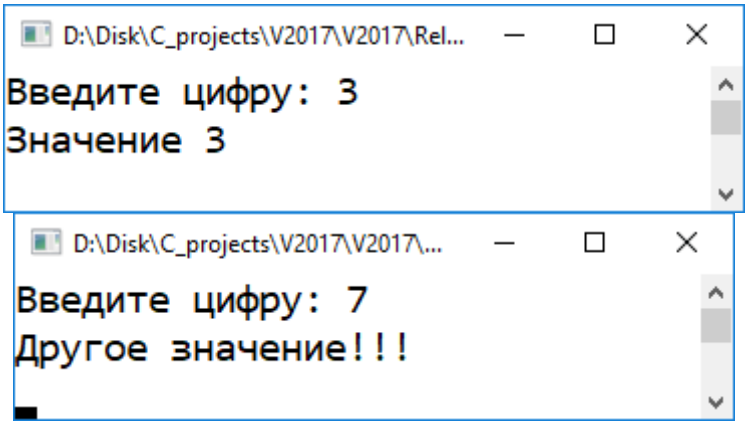


Рис. 4.33. Внешний вид окна ввода-вывода

Лабораторная работа №6

Массивы

1. Цель работы

Целью работы является изучение принципов работы с массивами в языке Си.

2. Программа работы

Выполните следующие пункты работы:

1. Написать код программы расчета суммы элементов массива в соответствии с вариантом (табл. 4.13).

Таблица 4.13

Варианты заданий

Вариант	Массив
1	{1,25,3,4,-10,50,-100,2,50,9}
2	{2,4,-3,4,-20,21,-10,22,37,-9}
3	{3,15,13,14,-15,-50,100,-2,10,-9}
4	{4,5,78,-80,95,51,-30,15,12,5}
5	{27,-8,-7,-9,-5,89,-50,25,24,23}
6	{-25,56,85,90,1,2,5,8,78,-93}

Пример выполнения программы для массива {1,2,-8,9,-48,50,21,37,-5,10} представлен на рис. 4.34.

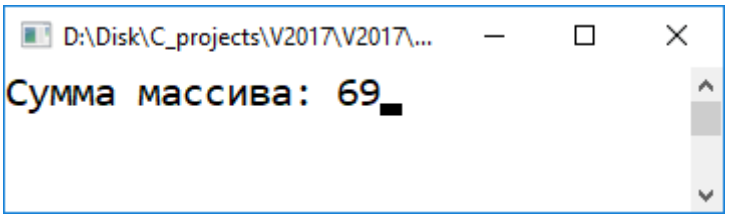


Рис. 4.34. Пример выполнения программы для массива {1,2,-8,9,-48,50,21,37,-5,10}

2. Написать код программы определения минимального значения в массиве в соответствии с вариантом (табл. 4.14).

Таблица 4.14

Варианты заданий

Вариант	Массив
1	{1,5,3,4,-10,50,-10,2,5,9}
2	{2,4,-3,4,-20,1,-1,2,37,-9}
3	{31,150,13,1,-15,-50,10,-2,1,-9}
4	{14,51,78,-8,95,51,-30,25,12,5}
5	{7,-8,-7,-9,-5,8,-50,5,24,2}
6	{-25,6,5,9,1,2,5,8,8,-3}

Пример выполнения программы для массива {1,2,-8,9,-48,50,21,37,-5,10} представлен на рис. 4.35.

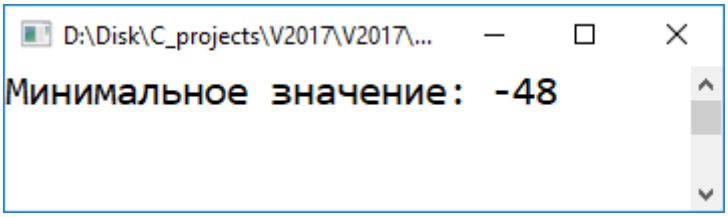


Рис. 4.35. Пример выполнения программы для массива {1,2,-8,9,-48,50,21,37,-5,10}

3. Написать программу, которая выводит на экран массив состоящий из суммы элементов двух других массивов в соответствии с вариантом (табл. 4.15).

Таблица 4.15

Варианты заданий

Вариант	Массив
1	{1,5,3,4,-10,50,-10,2,5,9} {7,-8,-7,-9,-5,8,-50,5,24,2}
2	{2,4,-3,4,-20,1,-1,2,37,-9} {14,51,78,-8,95,51,-30,25,12,5}
3	{31,150,13,1,-15,-50,10,-2,1,-9} {-25,6,5,9,1,2,5,8,8,-3}
4	{14,51,78,-8,95,51,-30,25,12,5} {31,150,13,1,-15,-50,10,-2,1,-9}
5	{7,-8,-7,-9,-5,8,-50,5,24,2} {14,51,78,-8,95,51,-30,25,12,5}

6	{-25,6,5,9,1,2,5,8,8,-3} {1,5,3,4,-10,50,-10,2,5,9}

Пример выполнения программы для массивов {1,2,-8,9,-48,50,21,37,-5,10} и {-25,6,5,9,1,2,5,8,8,-3} представлен на рис. 4.36.

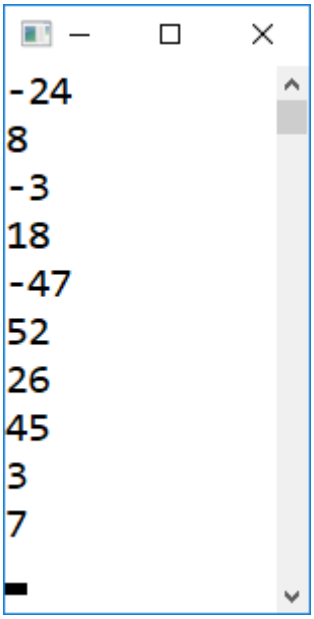


Рис. 4.36. Пример выполнения программы для массивов {1,2,-8,9,-48,50,21,37,-5,10} и {-25,6,5,9,1,2,5,8,8,-3}

4. Написать программу, которая производит замену всех отрицательных чисел в массиве на 0 и выводит его на экран (табл. 4.16).

Таблица 4.16

Варианты заданий

Вариант	**Массив**
1	{1,5,3,4,-10,50,-10,2,5,9}
2	{2,4,-3,4,-20,1,-1,2,37,-9}
3	{31,150,13,1,-15,-50,10,-2,1,-9}
4	{14,51,78,-8,95,51,-30,25,12,5}
5	{7,-8,-7,-9,-5,8,-50,5,24,2}
6	{-25,6,5,9,1,2,5,8,8,-3}

Пример выполнения программы для массивов {1,2,-8,9,-48,50,21,37,-5,10} представлен на рис. 4.37.

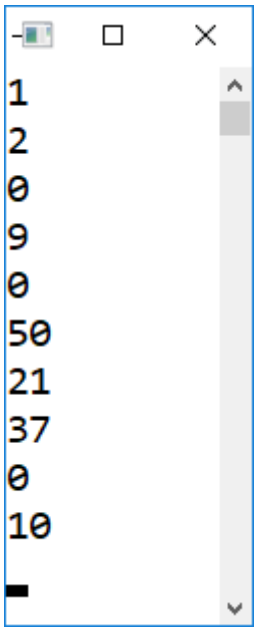


Рис. 4.36. Пример выполнения программы для массивов {1,2,-8,9,-48,50,21,37,-5,10}

Лабораторная работа №7

Функции

1. Цель работы

Целью работы является изучение использования функций в языке Си.

2. Программа работы

Выполните следующие пункты работы:

1. Написать код программы вывода на экран различной информации согласно табл. 4.17 в зависимости от нажатой клавши. Вывод каждой информации должен осуществляться в соответствующей функции.

Таблица 4.17

Данные вывода на экран

Клавиша	**Данные для вывода**
F	ФИО
B	Год рождения
A	Возраст
G	Группа

2. Написать код программы возведения числа в степени 2,3,4 или 5. Возведение в степень и вывод на экран организовать в соответствующей функции. Степень вводится с клавиатуры.

Список литературы

1. Торгаев С. Н. , Воробьева Г. С. , Мусоров И. С. , Чертихина Д. С. Основы микропроцессорной техники: микропроцессор INTEL 8080: Учебное пособие. - Томск : Изд-во ТПУ, 2014 - 208 с.

2. GenDocs.ru [Электронный ресурс] / Лекции - Основы микропроцессорной техники – Режим доступа: http://gendocs.ru/v24363, свободный. – Загл. с экрана. – Яз. рус. (дата обращения 09/04/2018).

3. Керниган, Брайн У.. Язык программирования С : пер. с англ. / Б. Керниган, Д. Ритчи. — 2-е изд., перераб. и доп.. — М. [и др.]: Вильямс, 2006. — 289 с.: 23 см.. — Серия книг по программированию от Prentice Hall. — Предметный указатель: с. 284-289.

4. code-live.ru [Электронный ресурс] / Портал о программировании – Режим доступа: https://code-live.ru/tag/cpp/, свободный. – Загл. с экрана. – Яз. рус. (дата обращения 01/04/2018).

5. Костюкова, Нина Ивановна. Язык Си и особенности работы с ним : учебное пособие / Н. И. Костюкова, Н. А. Калинина. — Москва: БИНОМ. Лаборатория знаний Интернет-Университет информационных технологий, 2006. — 205 с.. — Основы информационных технологий. — Библиогр.: с. 205.

СОЛДАТОВ АЛЕКСЕЙ ИВАНОВИЧ
ТОРГАЕВ СТАНИСЛАВ НИКОЛАЕВИЧ
ЛЕЖНИНА ИННА АЛЕКСЕЕВНА
ГРОМОВ МАКСИМ ЛЕОНИДОВИ
Хан Вэй
Костина Мария Алексеевна

ОСНОВЫ ПРОГРАММИРОВАНИЯ НА ЯЗЫКЕ СИ

Учебное пособие