ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)

В.В. Кручинин

ТЕХНОЛОГИИ ПРОГРАММИРОВАНИЯ

Руководство к организации самостоятельной работы

ТОМСК – 2012

Федеральное агентство по образованию

ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)

Кафедра промышленной электроники

В.В. Кручинин

ТЕХНОЛОГИИ ПРОГРАММИРОВАНИЯ

Руководство к организации самостоятельной работы

2012

Кручинин В.В.
Технологии программирования: Руководство к организации самостоятельной работы. — Томск: Томский государственный университет систем управления и радиоэлектроники, 2012. — 52 с.

СОДЕРЖАНИЕ

1 ВВЕДЕНИЕ

Целью курса является изучение принципов использования систем быстрой разработки программного обеспечения (RAD). К таким системам относятся системы визуального программирования Delphi, Cbuilder, VisualC и т.д. Рассматривается технология программирования RAD, изучаются составные части: редакторы, библиотеки компонент, репозитарий и др., изучаются основные компоненты, сборку приложения из компонент, создания собственных компонент.

В результате изучения курса студенты должны иметь представление об особенностях создания и функционирования приложений для ОС Windows, основанных на применении визуального программирования. Уметь проектировать приложения, используя систему визуального программирования, реализовывать их на языке программирования Си++.

Дисциплина «Технологии программирования» базируется на курсах «Информатика», «Операционные системы», «Программирование», «Объектно-ориентированное программирование» и «Базы данных».

2 СОДЕРЖАНИЕ ЛЕКЦИОННОГО КУРСА

Лекция 1. (2 часа) Введение в событийно-ориентированное программирование. Основные понятия и структуры. Главная функция WinMain. Оконный класс, регистрация класса, создание основного окна, цикл обработки сообщений.

Лекция 2. (2 часа) Обработка событий от клавиатуры и мышки, создание простейшего редактора, особенности обработки событий WM_PAINT и WM_CHAR.

Лекция 3. (2 часа) Ресурсы. Общая схема построения ресурсов, меню, иконки, изображения, тексты,

Лекция 4. (2 часа). Организация диалоговых окон, построение ресурса, обработка события WM_INIT, функции MessageBox, CreateDialog, DialogBox, диалоговые процедуры.

Лекция 5. (2 часа). Шрифты. Типы шрифтов, функции и структуры для создания и манипулирования шрифтами.

Лекция 6. (2 часа). Обработка текста, скроллинг.

Лекция 7. (2 часа). Стандартные элементы управления.

Лекция 8. (2 часа) Графика. Понятие контекста устройства, графические объекты и функции. Построение простейшего графического редактора.

Лекция 9. (2 часа) Введение в визуальное программирование.

Лекция 10. (2 часа) Библиотеки визуальных компонент, понятие компоненты, свойства , методы события.

Лекция 11. (2 часа) Понятие проекта, панели инструментов, инспектора объектов.

Лекция 12. (2 часа). Понятие формы, основные свойства, события и методы

Лекция 13. (2 часа). Структура VCL-классов.

Лекция 14. (2 часа). Пример разработки простого приложения.

Лекция 15. (2 часа). Техника программирования простых приложений.

Лекция 16. (2 часа). Создание собственных компонентов.

3 ЛАБОРАТОРНЫЕ РАБОТЫ (40 часов)

Лабораторная работа № 1. Основные элементы Windows-приложения (4 часа).

1. Главная программа WinMain.
1.1. Параметры (HINSTANCE, LPSTR, cmShow).
1.1. Регистрация класса (WNDCLASS, RegisterClass).
1.2. Создание окна (CreateWindows, HWND) ShowWindow. UpdateWindow.
1.3. Цикл обработки сообщений (GetMessage, TranslateMessage, DispatchMessage).

2. Оконная процедура (WndProc).
2.1. Параметры (hwnd, Msg, wParam, lParam).
2.2. Wm_Create.
2.3. Wm_Paint.
2.4. Wm_Destroy.

Лабораторная работа № 2. Обработка сообщений от клавиатуры и мышки (4 часа).

1. WM_KEYDOWN.
2. WM_KEYUP.
3. WM_CHAR.
4. WM_SYSKEYDOWN.
5. WM_SYSKEYUP.

6. WM_MOUSEMOVE.
7. WM_LBUTTONDOWN.
8. WM_RBUTTONDOWN.
9. WM_LBUTTONUP.
10. WM_RBUTTONUP.
11. WM_LBUTTONDBLCLK.
12. Простейший редактор строки (InvalidateRect). Виртуальные клавиши.

Лабораторная работа № 3. Ресурсы (4 часа).

1. Организация меню (WM_COMMAD).
2. Хранение и отображение в окне растровых изображений.
3. Курсоры и иконки.
4. Текст.
5. Создание и использование собственных ресурсов.

Лабораторная работа № 4. Диалоговые ящики (DialogBox) (4 часа).

1. Использование MessageBox.
2. Описание диалоговой панели в ресурсе.
3. Создание оконной процедуры диалоговой панели.
4. Вызов вызов диалоговой панели в WinMain.
5. Вызов диалоговой панели в меню (About).
6. Диалоговый ящик — часы (WM_TIMER).

Лабораторная работа № 5. Обработка шрифтов(4 часа).

1. EnumFontFamelies.
2. Диалоговая панель для просмотра шрифтов.

Лабораторная работа № 6. Графический редактор(4 часа).

Ввод и отображение графической информации

Лабораторная работа № 7. Изучение основных элементов визуального программирования. Использование формы для создания приложения.

Лабораторная работа № 8. Использование стандартных компонент. Tlabel, Tedit, Tbutton.

Лабораторная работа № 9. Создание и использование DLL (4 часа).

Лабораторная работа № 10. Создание собственных компонент (4 часа).

4 САМОСТОЯТЕЛЬНАЯ РАБОТА (126 часов)

Целью самостоятельной работы является формирование и закрепление навыков, знаний и умений по созданию приложений с использованием RAD систем.

Самостоятельная работа включает индивидуальные задания по проектированию и реализации конкретных приложений для ОС Windows, ориентированных на решение конкретных задач.

Индивидуальные задания в 8 семестре включают в себя следующие этапы работ:

- анализ задачи, обзор литературы;
- формирование требований к программе;
- построение алгоритма и его анализ;
- разработка структуры и интерфейса программы;
- программная реализация;
- тестирование и отладка;
- представление программы и оформление отчета.

Индивидуальное задание завершается в 9 семестре выполнением курсового проекта.

5 ПРАКТИЧЕСКИЕ ЗАНЯТИЯ (16 часов)

Перечень практических занятий:

1. Работа с редактором форм (2 часа).
2. Работа с редактором меню (2 часа).
3. Использование компонент ListBox и ComboBox (2 часа).
4. Использование компонент Label, ScrollBar, Panel (2 часа).
5. Стандартные диалоговые окна (2 часа).
6. Использование файлов ресурсов (2 часа).
7. Создание группы проектов (2 часа).
8. Обработка исключений (2 часа).

6 КУРСОВОЙ ПРОЕКТ (22 часа)

Курсовой проект предназначен для закрепления теории и приобретения практических навыков в области использования систем визуального проектирования программ. Курсовое проектирование связано с выполнением самостоятельной работы студента и является продолжением выполнения индивидуального задания. Примерные темы индивидуальных заданий:

1. Создание программы визуализации объектов и явлений различной природы.

2. Разработка программы моделирования различных электронных схем, редакторы электронных схем.

3. Разработка программ учебного назначения и компьютерных учебников.

4. Разработка программ системного характера, архиваторы, кодировщики, интерпретаторы.

5. Разработка различного вида тренажеров.

6 МЕТОДИКА ФОРМИРОВАНИЯ ТЕКУЩЕГО РЕЙТИНГА

Лабораторные занятия выполняются согласно расписанию занятий.

Собеседование проводится во время экзаменационной сессии.

Максимальный рейтинг по дисциплине составляет 120 баллов и определяется по таблице 1. Для получения оценки «отлично» требуется набрать не менее 100 баллов, «хорошо» — 80 баллов.

Таблица 1 — Распределение максимального рейтинга по элементам контроля

№	Виды контроля	Максим. балл
1	Посещение лекций	20
2	Лабораторная работа № 1	4
3	Лабораторная работа № 2	4
4	Лабораторная работа № 3	4
5	Лабораторная работа № 4	4
6	Лабораторная работа № 5	4
7	Лабораторная работа № 6	4
5	Лабораторная работа № 7	4
6	Лабораторная работа № 8	4
7	Лабораторная работа № 9	4
8	Лабораторная работа № 10	4
9	Самостоятельная работа (индивид. задание)	50
10	Собеседование	10
11	Всего баллов	120

Для оценки текущей работы студента по курсовому проектированию рейтинг выстроен так, что оценивается каждый этап курсового проекта. Распределение баллов по этапам приведено в таблице 2.

Таблица 2 — Распределение максимального рейтинга по курсовому проекту

№	Виды контроля	Максим. балл
1	Техническое задание	5
2	Обзор литературы	15
3	Выбор решения	15
4	Разработка структуры	15
5	Разработка алгоритма	20
6	Разработка программы	20
7	Оформление пояснительной записки	10
8	Защита проекта	20
9	Всего баллов	120

7 ЛИТЕРАТУРА

7.1 Основная литература

1. Borland C++Builder 3. Освой самостоятельно — М.: Бином, 1999. — 736 с.

2. Петзолд,Ч. Программирование для Windows 95: Все секреты программирования для Windows 95. Т.1 // Мастер: Руководство для профессионалов. — Санкт-Петербург, изд-во BHV, 1997. — 752 с.: ил. (2 экз.)

3. Петзолд Ч. Программирование для Windows 95: Все секреты программирования для Windows 95. Т.2 // Мастер: Руководство для профессионалов. — Санкт-Петербург, изд-во BHV, 1997. 368 с.: ил. (2 экз.)

4. Кручинин, В.В. Алгоритмические языки и технология программирования: Учебное пособие. — Томск: ТУСУР, 2001. — 126 с. (Электронный ресурс образовательного портала ТУСУРа http://portal.tusur.ru/cesir/?resource=24).

7.2. Дополнительная литература

1. Фролов А.В., Фролов Г.В. Графический интерфейс GDI в MS Windows // Библиотека системного программиста. "ДИАЛОГ-МИФИ", 1994. — 288 с.: ил. (2 экз.)

2. Фролов А.В., Фролов Г.В. Мультимедиа для Windows // Библиотека системного программиста. "ДИАЛОГ-МИФИ", 1994. — 284 с.: ил. (2 экз.)

3. Фролов А.В., Фролов Г.В. Операционная система Windows 95 // Б-ка системного программиста. "ДИАЛОГ-МИФИ", 1996. — 288 с.: ил. (2 экз.)

4. Сван, Том. Программирование для Windows в Borland C++. БИНОМ, 1995. — 480 с.: ил. (2 экз.)

5. Фролов А.В., Фролов Г.В. Microsoft Visual C++ и MFC: Программирование для Windows 95 и Windows NT. Ч.1 // Б-ка системного программиста. "ДИАЛОГ-МИФИ". 1995. — 288 с. (2 экз.)

6. Фролов А.В., Фролов Г.В. Microsoft Visual C++ и MFC: Программирование для Windows 95 и Windows NT. Ч.2 // Б-ка системного программиста. "ДИАЛОГ-МИФИ". 1995. — 272 с. (2 экз.)

ЛАБОРАТОРНЫЙ ПРАКТИКУМ ПО КУРСУ «ТЕХНОЛОГИИ ПРОГРАММИРОВАНИЯ»

Лабораторная работа № 1. «Первая программа »

Цель работы

Познакомиться с основными сервисами интегрированной системы C++ Builder. Получить навыки работы с компонентами C++ Builder (Label, Button, Edit, Memo, ListBox и т.д.). Изучить их свойства, методы и обработчики событий. Написать и отладить небольшое приложение с указанными компонентами.

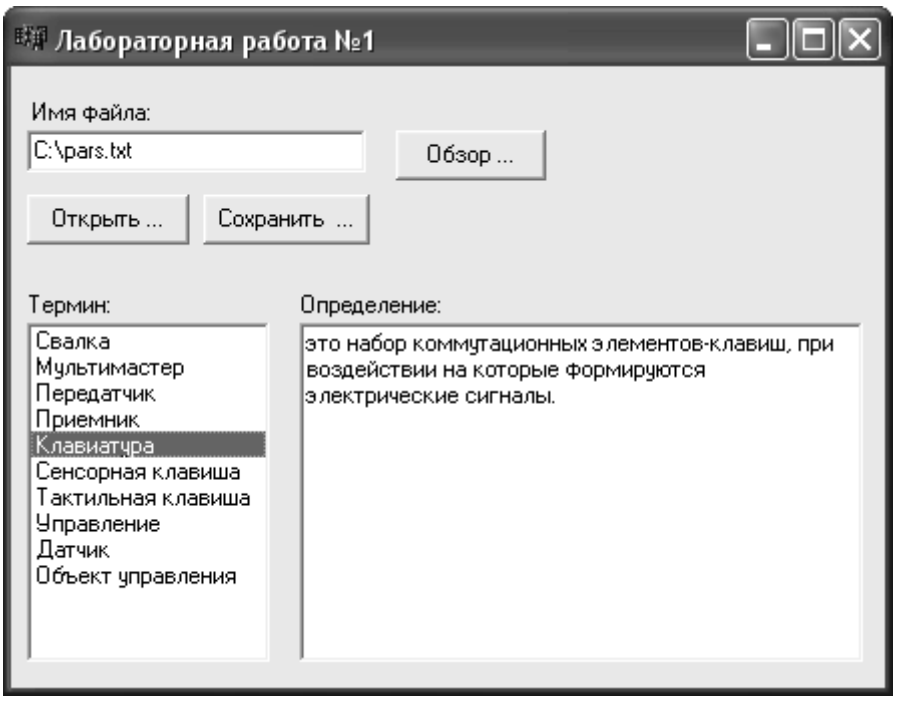
Задание

1. Нанести на форму:
   Button — Открыть, Сохранить, Обзор.
   Label — Термин, Значение, Имя файла.
   Memo — отображение значения термина.
   ListBox — отображение списка терминов.
   OpenDialog — вызывает диалоговое окно в котором можно выбрать имя файла.

2. Создать текстовый файл следующей структуры:
   <Термин>
   <Имя> Свалка </Имя>
   <Определение>Куча мусора </ Определение >
   </Термин>
   Ввести 10 терминов на свободную тему.

3. По нажатию кнопки «Обзор» открывается диалоговое окно (OpenDialog), в котором выбирается созданный текстовый файл. При этом в Edit отображается имя файла. После нажатия кнопки «Открыть» данные из файла выводятся в ListBox. При выделении пункта ListBox в Memo отображается его значение.

Исходный код программы и графическое представление



В класс TForm добавляются следующие поля и методы.

```
…
private:     // User declarations
   //Функция Заполнения списка терминов по исходному файлу
   bool Parser(TList *list, char *pszNameFiles);
   //Функция получения строки между тэгами
   AnsiString SetString(char *pszEndTag, char *&car);
   //Функция получения структуры термина
   tagTerm* GetTerm(char *&car);
   //Функция получения списка терминов из структурированной строки
   void GetTerms(TList *list, char *pszString);
   TList *list;
…
```

Описание используемых методов и событий.

```
void __fastcall TForm1::BrowswButtonClick(TObject *Sender)
{
   //Открываем диалоговое окно
   if(OpenDialog1->Execute())
   {
      //Если нажата кнопка Открыть(Open) присваеваем
      //имя выбранного файлае Edit1
      Edit1->Text = OpenDialog1->FileName;
   }
}

void __fastcall TForm1::OpenButtonClick(TObject *Sender)
```

```
{
   int count;
   //Заполняем список структур
   Parser(list, Edit1->Text.c_str());
   count = list->Count;
   for(int i = 0; i < count; i++)
   {
      //Добавляем в ListBox записи
      ListBox1->Items->Add(((tagTerm*)list->Items[i])->Name);
   }
   //Отображаем в Memo значение первого термина
   Memo1->Text = ((tagTerm*)list->Items[0])->Definition;
}

void __fastcall TForm1::ListBox1Click(TObject *Sender)
{
   //Узнаем индекс выбранного пунка в ListBox
   int index = ListBox1->ItemIndex;
   //Отображаем определение термина в Memo
   Memo1->Text = ((tagTerm*)list->Items[index])->Definition;
}

//Функция Заполнения списка терминов по исходному файлу
bool TForm1::Parser(TList *list, char *pszNameFiles)
{
   int iFileHandle;
   int iFileLength;
   int iBytesRead;
   char *pszBuffer;
   WIN32_FIND_DATA findData;
   HANDLE hFile;
   //Очищаем память под findData
   ZeroMemory(&findData, sizeof(findData));
   //Проверяем существует ли файл
   hFile = FindFirstFile(pszNameFiles,&findData);
   if(INVALID_HANDLE_VALUE != hFile){
      //Откpываем файл
      iFileHandle = FileOpen(pszNameFiles, fmOpenRead);
      //Какова длина файла
      iFileLength = FileSeek(iFileHandle,0,2);
      FileSeek(iFileHandle,0,0);
      //Выделяем память под содержимое файла
      pszBuffer = new char[iFileLength+1];
      //Заполняем буфер содержимым файла
      iBytesRead = FileRead(iFileHandle, pszBuffer, iFileLength);
      //Закрываем файл
      FileClose(iFileHandle);
      //Заполняем список терминов
      GetTerms(list, pszBuffer);
      delete [] pszBuffer;
```

```
      pszBuffer = NULL;
      return true;
   }else return false;
}

//Функция получения списка терминов из структурированной строки
void TForm1::GetTerms(TList *list, char *pszString)
{
   while(*pszString)
   {
      if(match("<Термин>", pszString))
      {
         list->Add(GetTerm(pszString));
         continue;
      }
      pszString++;
   }
}

//Функция получения структуры термина
tagTerm* TForm1::GetTerm(char *&car)
{
   tagTerm *term = new tagTerm;
   while(!match("</Термин>", car) && *car)
   {
      if(match("<Имя>", car))
      {
         term->Name = SetString("</Имя>",  car);
      }

      if(match("<Определение>", car))
      {
         term->Definition = SetString("</Определение>",  car);
      }
      car++;
   }
   return term;
}

//Функция получения строки между тэгами.
AnsiString TForm1::SetString(char *pszEndTag, char *&car)
{
   AnsiString str = "";
   while(!match(pszEndTag, car) && *car)
   {
      str += *car++;
   }
   return str;
}
```

```
//Функция пропускает служебные символы
void SkipBlanks(char *&PointBuff)
{
    while(true)
    {
            switch (*PointBuff)
            {
                    case 8: //табуляция
                    case 26: //конец файла
                    case 10: //возврат каретки
                    case 13: //перевод строки
                    case ' ': //пробел
                       PointBuff++; break;
                    default: //если это не служебный
                       return;
            }
    }
}

//Функция Сравнивает две строки и если они равны то перемещает каретку
bool match(char *pszTag, char *&car)
{
    char *pStr = car;
    SkipBlanks(pStr);
    while(*pszTag)
   {
      if((*pszTag++)!=(*pStr++))
         return false;
   }
    car = pStr;
    return true;
}
```

Контрольные вопросы

1. Записать основные возможности интегрированной системы CBuieder для быстрой разработки приложений.

2. Определить понятия компоненты ее свойства, методы и обработчики событий.

3. Записать предназначение, основные свойства, методы и обработчики событий компоненты TButton.

4. Дать определение основным свойствам, методам и обработчикам событий компоненты TLabel.

5. Записать предназначение, основные свойства, методы и обработчики событий компоненты TMemo.

6. Записать предназначение, основные свойства, методы и обработчики событий компоненты TListBox.

7. Записать предназначение, основные свойства, методы и обработчики событий компоненты TForm.

Лабораторная работа № 2. «Текстовый редактор»

Цель работы

Создать полноценный текстовый редактор с возможностью изменения шрифта, его размера и цвета. Сохранение и открытие файла формата .rtf и .txt.

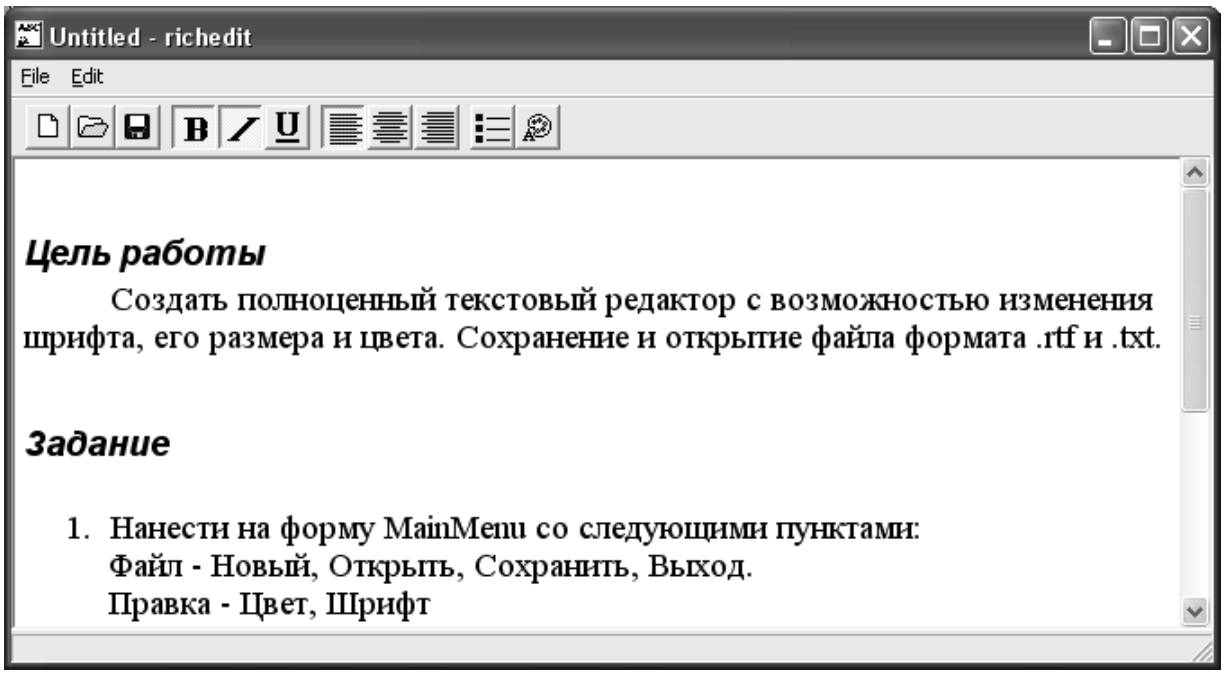
Задание

1. Нанести на форму MainMenu со следующими пунктами:
   Файл — Новый, Открыть, Сохранить, Выход.
   Правка — Цвет, Шрифт.
   ToolBar с кнопками:
   Новый, Открыть, Сохранить, Жирный, Курсив, Подчеркивание, Маркированный список, Нумерованный список, Цвет.
   RichEdit — для ввода текста (Align=alClient).
   OpenDialog — вызывает диалоговое окно в котором можно выбрать имя файла.
   SaveDialog — вызывает диалоговое окно в котором можно выбрать имя файла.
   ColorDialog — вызывает диалоговое окно в котором можно выбрать цвет.
   FontDialog — вызывает диалоговое окно в котором можно выбрать параметры шрифта.

2. При выборе пункта меню Файл — Новый содержимое RichEdit очищается. При выборе пункта Файл — Открыть вызывается OpenDialog. При выборе пункта Файл — Сохранить как.. вызывается SaveDialog. При выборе пункта Правка — Цвет вызывается ColorDialog. При выборе Правка — Шрифт вызывается FontDialog.

3. При нажатии кнопки Новый содержимое RichEdit очищается. При нажатии кнопки Открыть вызывается OpenDialog. При нажатии кнопки Сохранить файл сохраняется под текущим именем. При нажатии кнопки Жирный выделенный фрагмент текста становится жирным. При нажатии кнопки Курсив выделенный текст становится курсивом. При нажатии кнопки Подчеркивание выделенный фрагмент текста становится подчеркнутым. При нажатии кнопки Маркированный список в необходимом месте ставится маркер. При нажатии кнопки Нумерованный список в необходимом месте начинается нумерация. При нажатии кнопки Цвет вызывается ColorDialog.

Исходный код программы и графическое представление




```
//Событие выделение текста (перемещение каретки)
void __fastcall TMainForm::SelectionChange(TObject */*Sender*/)
{
  TParaAttributes *Paragraph = RichEdit1->Paragraph;
  TTextAttributes *SelAttributes = RichEdit1->SelAttributes;
  char sizebuf[6];

       BoldButton->Down = SelAttributes->Style.Contains(fsBold);
       ItalicButton->Down = SelAttributes->Style.Contains(fsItalic);
       UnderlineButton->Down = SelAttributes->Style.Contains(fsUnderline);
       BulletsButton->Down = static_cast<bool>(Paragraph->Numbering);

    switch(AsInt(Paragraph->Alignment))
    {
      case 0:
        LeftAlign->Down = True;
        break;

      case 1:
        RightAlign->Down = True;
        break;

      case 2:
        CenterAlign->Down = True;
        break;
        }
  }
}
```

```
//Нажатие на кнопку «Жирный»
void __fastcall TMainForm::BoldButtonClick(TObject* /*Sender*/)
{
   if(BoldButton->Down)
     CurrText()->Style = CurrText()->Style << fsBold;
   else
     CurrText()->Style = CurrText()->Style >> fsBold;
}

//Нажатие на кнопку «Курсивный»
void __fastcall TMainForm::ItalicButtonClick(TObject* /*Sender*/)
{
   if(ItalicButton->Down)
     CurrText()->Style = CurrText()->Style << fsItalic;
   else
     CurrText()->Style = CurrText()->Style >> fsItalic;
}
//Нажатие на кнопку «Подчеркнутый»
void __fastcall TMainForm::UnderlineButtonClick(TObject* /*Sender*/)
{
   if(UnderlineButton->Down)
     CurrText()->Style = CurrText()->Style << fsUnderline;
   else
     CurrText()->Style = CurrText()->Style >> fsUnderline;
}

//Нажатие на кнопку «Выравнивание: Справа, Слева, По Центру»
void __fastcall TMainForm::AlignClick(TObject* Sender)
{
   TControl *oAliBtn = dynamic_cast<TControl*>(Sender);
   RichEdit1->Paragraph->Alignment = static_cast<TAlignment>(oAliBtn->Tag);
}

//Нажатие на кнопку «Список»
void __fastcall TMainForm::BulletsButtonClick(TObject* /*Sender*/)
{
   RichEdit1->Paragraph->Numbering = static_cast<TNumberingStyle>(BulletsButton-
>Down);
}

//Выбор пункта меню File-Open
void __fastcall TMainForm::FileOpenClick(TObject* /*Sender*/)
{
  if (OpenDialog->Execute())
  {
   RichEdit1->Lines->LoadFromFile(OpenDialog->FileName);
   SetFileName(OpenDialog->FileName);
   RichEdit1->SetFocus();
   RichEdit1->Modified = False;
   RichEdit1->ReadOnly = OpenDialog->Options.Contains(ofReadOnly);
```

```
  }
}

//Выбор пункта меню File-Save
void __fastcall TMainForm::FileSaveClick(TObject* Sender)
{
  if ( !strcmp(FFileName.c_str(), Reconst_SUntitled.c_str()) )
    FileSaveAsClick(Sender);
  else
  {
    RichEdit1->Lines->SaveToFile(FFileName);
    RichEdit1->Modified = False;
  }
}

//Выбор пункта меню File-Save as…
void __fastcall TMainForm::FileSaveAsClick(TObject* /*Sender*/)
{
  if ( SaveDialog->Execute() )
  {
    // Options + OverwritePrompt = True, thus no need to check.
    RichEdit1->Lines->SaveToFile(SaveDialog->FileName);
    SetFileName(SaveDialog->FileName);
    RichEdit1->Modified = False;
  }
}

//Выбор пункта меню File-Exit
void __fastcall TMainForm::FileExitClick(TObject* /*Sender*/)
{
  Close();
}

//Выбор пункта меню Edit-Font
void __fastcall TMainForm::SelectFont(TObject* /*Sender*/)
{
  FontDialog1->Font->Assign(RichEdit1->SelAttributes);

  if(FontDialog1->Execute())
    CurrText()->Assign( FontDialog1->Font );

  RichEdit1->SetFocus();
}
```

Контрольные вопросы

1. Определите назначение и основные функции текстового редактора.
2. Записать предназначение, основные свойства, методы и обработчики событий компоненты TRichEdit.
3. Записать предназначение, основные свойства, методы и обработчики событий компоненты ToolBar
4. Записать предназначение, основные свойства, методы и обработчики событий компоненты OpenDialog.

Лабораторная работа № 3. «Графический редактор» (режим рисования)

Цель работы

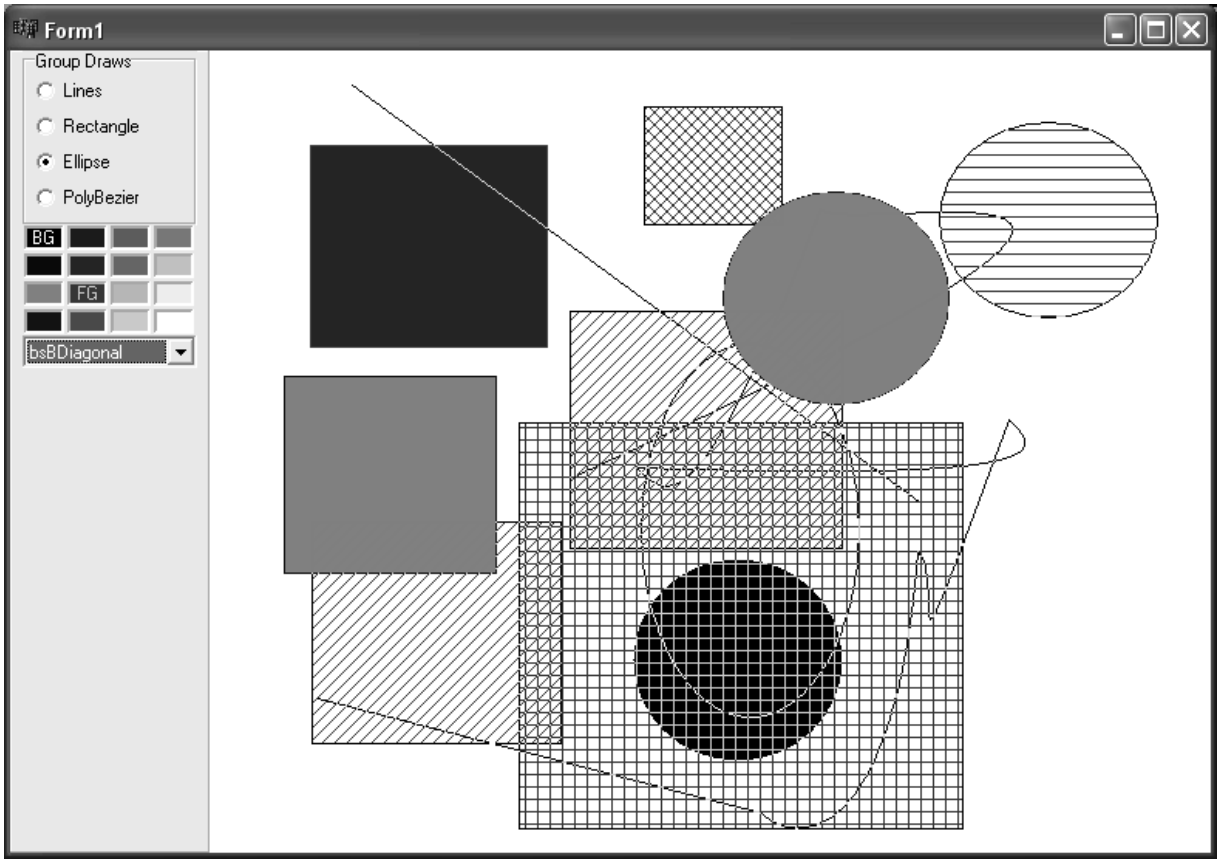
Создать простевший графический редактор с функциями рисования линий, прямоугольников, эллипсов и т.д. Для этого создать режим резиновой фигуры.

Задание

1. Нанести на форму Panel со следующими разделами:
   ColorGrid — для отображения активного цвета.
   RadioGroup — для используемого элемента (Квадрат, Эллипс, Линия, Полилиния, Стерка).
   ComboBox — для отображения типа штриховки.
   Image — будет представлять собой область для рисования.
2. Палитра цветов (ColorGrid) должна содержать не менее 12 цветов. RadioGroup для отображения фигуры должна содержать пункты: Линия, Квадрат, Эллипс, Полилиния, Стерка. Выпадающий список типов штриховки (ComboBox) должен содержать 8 значений: Полная заливка, Нет заливки, Горизонтальная штриховка, Вертикальная штриховка, Штриховка с наклоном вправо, Штриховка с наклоном влево, Квадратная штриховка, Квадратная наклонная штриховка.
3. При рисовании одной фигуры поверх другой нижняя закрывается верхней. Стерка должна удалять не всю фигуру, а только необходимую пользователю часть.

Исходный код программы и графическое представление

```
//Обработка события нажатия левой кнопки мыши
void __fastcall TDoodleForm::Image1MouseDown(TObject *Sender,
   TMouseButton Button, TShiftState Shift, int X, int Y)
{
   if (FillButton->Down)
   {
      if (Button == mbLeft)
         Image1->Canvas->Brush->Color = FGShape->Brush->Color;
      else
         Image1->Canvas->Brush->Color = BGShape->Brush->Color;
      Image1->Canvas->FloodFill(X, Y, Image1->Canvas->Pixels[X][Y], fsSurface);
      return;
   }
```




```
if (Button != mbLeft)
   return;

if (EraseButton->Down)
{
   Image1->Canvas->Pen->Color = BGShape->Brush->Color;
   Image1->Canvas->Brush->Color = BGShape->Brush->Color;
   Image1->Canvas->Pen->Width = 13;
   Image1->Canvas->Rectangle(X-1, Y-1, X, Y);
   Image1->Canvas->MoveTo(X,Y);
   return;
}

if (PencilButton->Down)
{
   Image1->Canvas->Pen->Color = FGShape->Brush->Color;
   Image1->Canvas->Brush->Color = BGShape->Brush->Color;
   Image1->Canvas->MoveTo(X,Y);
   return;
}

InitialX = X;
InitialY = Y;
```

```
   TmpImage = new TImage(this);
   TmpImage->Picture = Image1->Picture;
}

//Обработка события перемещения курсора мыши
void __fastcall TDoodleForm::Image1MouseMove(TObject *Sender, TShiftState Shift,
   int X, int Y)
{
   if (!Shift.Contains(ssLeft))
      return;

   if (FillButton->Down)
      return;

   if (PencilButton->Down)
   {
      Image1->Canvas->LineTo(X,Y);
      return;
   }

   if (EraseButton->Down)
   {
      Image1->Canvas->LineTo(X,Y);
      return;
   }

   DrawShape(X, Y);
}

void __fastcall TDoodleForm::Image1MouseUp(TObject *Sender, TMouseButton But-
ton,
   TShiftState Shift, int X, int Y)
{
   if (Button != mbLeft)
      return;

   if ((FillButton->Down) || (PencilButton->Down))
      return;

   if (EraseButton->Down)
   {
      Image1->Canvas->Pen->Width = 1;
      return;
   }

   DrawShape(X, Y);

   delete TmpImage;
}
//Функция рисующая фигуру на экране.
```

```
void __fastcall TDoodleForm::DrawShape(int X, int Y)
{
   TRect bounds;

   Image1->Picture = TmpImage->Picture;
   Image1->Canvas->Brush->Color = FGShape->Brush->Color;
   Image1->Canvas->Pen->Color = FGShape->Brush->Color;
   if (X < InitialX)
   {
      bounds.Left = X;
      bounds.Right = InitialX;
   }
   else
   {
      bounds.Right = X;
      bounds.Left = InitialX;
   }

   if (Y < InitialY)
   {
      bounds.Top = Y;
      bounds.Bottom = InitialY;
   }
   else
   {
      bounds.Bottom = Y;
      bounds.Top = InitialY;
   }

   if (CircleButton->Down)
      Image1->Canvas->Arc(InitialX, InitialY, X, Y, X, Y, X, Y);
   else if (SolidCirButton->Down)
      Image1->Canvas->Ellipse(InitialX, InitialY, X, Y);
   else if (SquareButton->Down)
      Image1->Canvas->FrameRect(bounds);
   else if (SolidSqButton->Down)
      Image1->Canvas->FillRect(bounds);
}
```

Контрольные вопросы

1. Назовите предназначение и основные методы и свойства компоненты TCanvas.
2. Назовите предназначение и основные методы и свойства компоненты TImage.
3. Назовите основные этапы создания режима «резиновая нить».
4. Определите предназначение кривых Безье.

Лабораторная работа № 4. «Графический редактор (режим редактирования)»

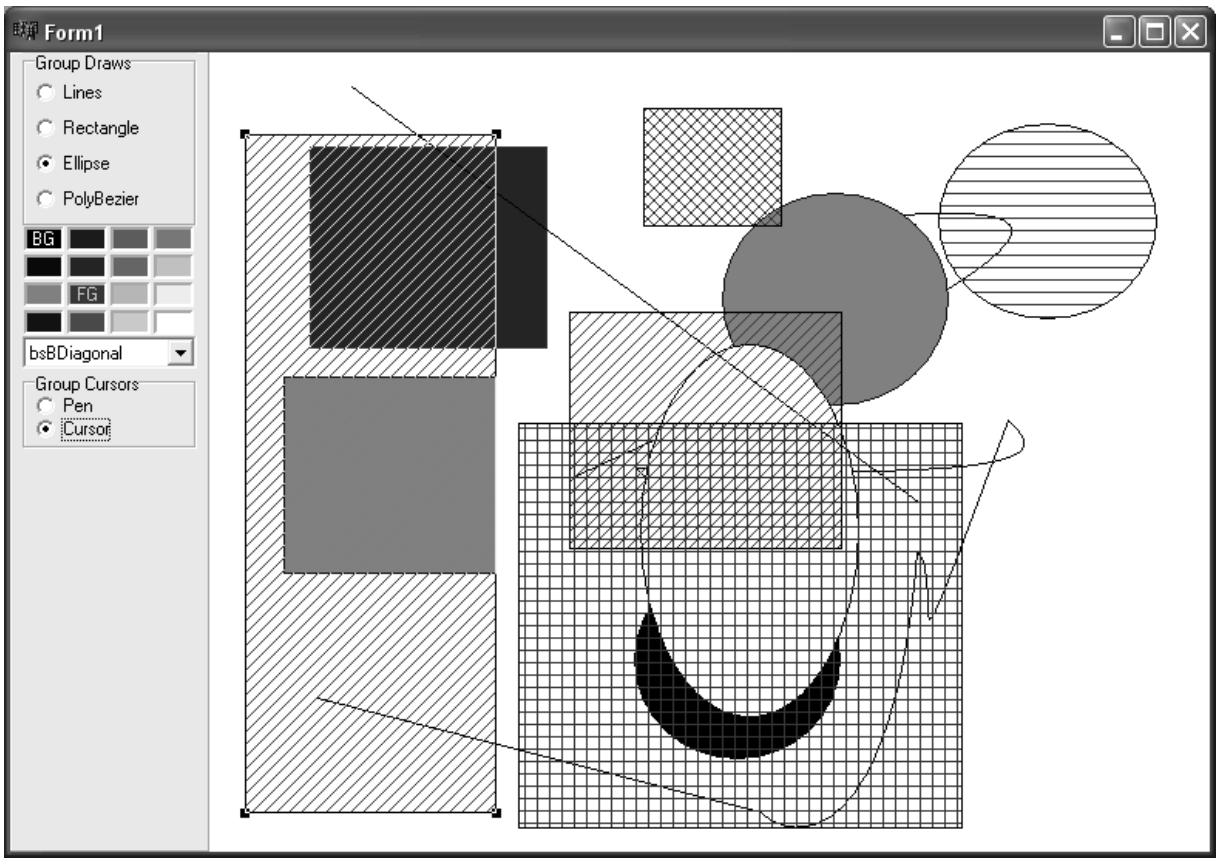
Цель работы

Создать графический редактор, который хранит графические элементы, как объекты. В режиме редактирования должна присутствовать возможность изменения размеров объекта и его расположения на экране.

Задание

1. Нанести на форму Panel со следующими разделами:
   ColorGrid — для отображения активного цвета.
   RadioGroup — для отображения рисуемой фигуры и для отображения активного режима.
   ComboBox — для отображения типа штриховки.
   Image — будет представлять собой область для рисования.
2. Палитра цветов (ColorGrid) должна содержать не менее 12 цветов. RadioGroup для отображения фигуры должна содержать пункты: Линия, Квадрат, Эллипс, Полилиния. RadioGroup для отображения активного режима должна содержать пункты: Курсор (режим редактирования), Ручка (режим рисования). Выпадающий список типов штриховки (ComboBox) должен содержать 8 значений: Полная заливка, Нет заливки, Горизонтальная штриховка, Вертикальная штриховка, Штриховка с наклоном вправо, Штриховка с наклоном влево, Квадратная штриховка, Квадратная наклонная штриховка.
3. При рисовании одной фигуры поверх другой нижняя закрывается верхней. В режиме редактирования границы выделенного объекта должны отображаться квадратиками (Линия — 2 квадратика, Квадрат, Эллипс, Полилиния — 4 квадратика). При наведении на границы должна быть возможность изменения размеров объекта. При выделении объекта должны быть возможности: 1) перемещение объекта по рабочей области; 2) удаления нажатием кнопки “Delete” на клавиатуре; 3) изменение цветов границ и штриховки.

Исходный код программы и графическое представление




```
//Базовый класс фигуры
class CDraws
{
protected:
   int nTop; //Y0
   int nLeft; //X0
   int nRight; //X1
   int nBottom; //Y1
   TCanvas *pCanvas;
   TColor cPenColor;
   TColor cBrushColor;
   TBrushStyle sBrushStyle;
   virtual TColor ReadPenColor();
   virtual void WritePenColor(TColor Color);
   virtual TColor ReadBrushColor();
   virtual void WriteBrushColor(TColor Color);
   virtual TBrushStyle ReadBrushStyle();
   virtual void WriteBrushStyle(TBrushStyle BrushStyle);
public:
   CDraws();
   virtual int Focused(TPoint pPoint){ return 0; }
   //Метод перезаписывает координаты фигуры и перерисовывает её
   virtual void ReWritableSize(TRect rDeltaRect);
   //Метод рисующий выделение
```

```
    virtual void Select(){}
    //Метод рисующий фигуру
    virtual void Draw(TRect rRect){}
    virtual void Writable();
    virtual void SetPenColor(TColor cColor);
    virtual void SetBrushColor(TColor cColor);
    virtual void SetBrushStyle(TBrushStyle bsStyle);
    virtual void SetCanvas(TCanvas *Canvas);
    __property TColor PenColor = {read=ReadPenColor, write=WritePenColor};
    __property TColor BrushColor = {read=ReadBrushColor, write=WriteBrushColor};
    __property TBrushStyle BrushStyle = {read=ReadBrushStyle,
write=WriteBrushStyle};
};

//Класс фигуры «Линия»
class CLines : public CDraws
{
protected:
    int LineFunc(int x);
public:
    bool IfLine(TPoint pPoint);
    virtual int Focused(TPoint pPoint);
    virtual void Select();
    virtual void Draw(TRect rRect);
    virtual void Writable();
};

//Класс фигуры «Прямоугольник»
class CRectangle : public CDraws
{
    virtual int Focused(TPoint pPoint);
    virtual void Select();
    virtual void Draw(TRect rRect);
};

//Класс фигуры «Эллипс»
class CEllipse : public CRectangle
{
    virtual void Draw(TRect rRect);
};

class CPolyBezier : public CRectangle
{
    class CPoint
    {
    public:
        TPoint nPoint[4];
        CPoint *pnNext;
        CPoint()
        {
```

```
        for(int i = 0; i < 4; i++)
          nPoint[i] = Point(0, 0);
      }
    };
protected:
    bool bOne;
    bool bEdit;
    CPoint *pnNew;
    CPoint *pnStart;
    CPoint *pnEnd;
public:
    CPolyBezier();
    virtual void Draw(TRect rRect);
    virtual void ReWritableSize(TRect rDeltaRect);
    virtual void Writable();
};

//Обработка события нажатия левой кнопки мыши
void __fastcall TForm1::ImageMouseDown(TObject *Sender,
     TMouseButton Button, TShiftState Shift, int X, int Y)
{
  if(Button == mbRight)
  {
    if(RadioGroup1->ItemIndex == 3)
    {
      DblClick = false;
      pNew->pnDraw->Writable();
    }
  }
  else
  {
    pCanvas->Pen->Mode = pmNotXor;
    Click = true;
    X2 = X;
    Y2 = Y;
    if(bSelect)
    {
      pNew->pnDraw->Select();
      bSelect = false;
    }
    if(G_Cursors->ItemIndex == 0)
    {
      pCanvas->Pen->Color = CColorGrid1->BackgroundColor;
      pCanvas->Brush->Color = CColorGrid1->ForegroundColor;
      pCanvas->Brush->Style = (TBrushStyle)StylePen->ItemIndex;
      if(RadioGroup1->ItemIndex == 3)
      {
        if(!DblClick)
        {
          Add(RadioGroup1->ItemIndex);
```

```
            DblClick = true;
          }
          pNew->pnDraw->Draw(Rect(X, Y, X, Y));
        }
        else
        {
          Add(RadioGroup1->ItemIndex);
          pNew->pnDraw->ReWritableSize(Rect(X, Y, X, Y));
        }
        X0 = X;
        Y0 = Y;
        X1 = X;
        Y1 = Y;
        pCanvas->Brush->Style = bsClear;
    }
    else
    {
        if(pEnd)
        {
          if(GetDraws(X, Y))
          {
            pNew->pnDraw->Select();
            bSelect = true;
            Size = pNew->pnDraw->Focused(Point(X, Y));
          }
        }
    }
  }
}

//Обработка события перемещения курсора мыши
void __fastcall TForm1::ImageMouseMove(TObject *Sender, TShiftState Shift,
      int X, int Y)
{
    if(bSelect && G_Cursors->ItemIndex == 1)
    {
        switch(pNew->pnDraw->Focused(Point(X, Y)))
        {
          /*устанавливаем вид курсора в зависимости от его положения на изображе-
нии выделенной фигуры*/
          case 2:
          case 3:
            Image1->Cursor =  crSizeNWSE;
          break;

          case 4:
          case 5:
            Image1->Cursor =  crSizeNESW;
          break;
```

```
        case 6:
        case 7:
           Image1->Cursor =  crSizeWE;
        break;

        case 8:
        case 9:
           Image1->Cursor =  crSizeNS;
        break;

        case 1:
           Image1->Cursor =  crSizeAll;
        break;

        default:
           Image1->Cursor =  crDefault;
        break;
     }
  }
  if(Click)
  {
     if(G_Cursors->ItemIndex == 0)
     {
        //Режим рисования
        pNew->pnDraw->ReWritableSize(Rect(0, 0, 0, 0));
        pNew->pnDraw->ReWritableSize(Rect(0, 0, X-X2, Y-Y2));
        X1 = X;
        Y1 = Y;
     }
     else
     {
        //Режим Редактирования
        if(bSelect)
        {
           switch(Size)
           {
              case 1:
                 //Убираем выделение
                 pNew->pnDraw->Select();
                 //Стираем нарисованную фигуру
                 pNew->pnDraw->ReWritableSize(Rect(0,0,0,0));
                 //рисуем новую фигуру
                 pNew->pnDraw->ReWritableSize(Rect(X-X2,Y-Y2,X-X2,Y-Y2));
                 //устанавливаем выделение
                 pNew->pnDraw->Select();
              break;
              case 2:
                 pNew->pnDraw->Select();
                 pNew->pnDraw->ReWritableSize(Rect(0,0,0,0));
                 pNew->pnDraw->ReWritableSize(Rect(X-X2,Y-Y2,0,0));
```

```
            pNew->pnDraw->Select();
        break;
        case 3:
            pNew->pnDraw->Select();
            pNew->pnDraw->ReWritableSize(Rect(0,0,0,0));
            pNew->pnDraw->ReWritableSize(Rect(0, 0,X-X2,Y-Y2));
            pNew->pnDraw->Select();
        break;
        case 4:
            pNew->pnDraw->Select();
            pNew->pnDraw->ReWritableSize(Rect(0,0,0,0));
            pNew->pnDraw->ReWritableSize(Rect(X-X2, 0,0,Y-Y2));
            pNew->pnDraw->Select();
        break;
        case 5:
            pNew->pnDraw->Select();
            pNew->pnDraw->ReWritableSize(Rect(0,0,0,0));
            pNew->pnDraw->ReWritableSize(Rect(0,Y-Y2,X-X2,0));
            pNew->pnDraw->Select();
        break;
        case 6:
            pNew->pnDraw->Select();
            pNew->pnDraw->ReWritableSize(Rect(0,0,0,0));
            pNew->pnDraw->ReWritableSize(Rect(X-X2,0,0,0));
            pNew->pnDraw->Select();
        break;
        case 7:
            pNew->pnDraw->Select();
            pNew->pnDraw->ReWritableSize(Rect(0,0,0,0));
            pNew->pnDraw->ReWritableSize(Rect(0,0,X-X2,0));
            pNew->pnDraw->Select();
        break;
        case 8:
            pNew->pnDraw->Select();
            pNew->pnDraw->ReWritableSize(Rect(0,0,0,0));
            pNew->pnDraw->ReWritableSize(Rect(0,Y-Y2,0,0));
            pNew->pnDraw->Select();
        break;
        case 9:
            pNew->pnDraw->Select();
            pNew->pnDraw->ReWritableSize(Rect(0,0,0,0));
            pNew->pnDraw->ReWritableSize(Rect(0,0,0,Y-Y2));
            pNew->pnDraw->Select();
        break;
        default:
            Image1->Cursor =  crDefault;
        break;

    }
}
```

```
        }
      }
        X2 = X;
        Y2 = Y;
}

void __fastcall TForm1::ImageMouseUp(TObject *Sender, TMouseButton Button,
      TShiftState Shift, int X, int Y)
{
    Click = false;
    pCanvas->Brush->Style = bsSolid;
    pCanvas->Pen->Mode = pmCopy;
    if(pNew)
    {
        pCanvas->Pen->Mode = pmNotXor;
        if(bSelect) pNew->pnDraw->Select();
        pCanvas->Pen->Mode = pmCopy;
//        pNew->pnDraw->Writable();
        RePaint();
        pCanvas->Pen->Mode = pmNotXor;
        if(bSelect) pNew->pnDraw->Select();
    }
}
```

Контрольные вопросы

1. Запишите представление графических объектов в графическом редакторе необходимое для создания режима редактирования.

2. Назовите предназначение и основные методы и свойства компоненты TPanel.

3. Перечислите способы выделения графических объектов в окне редактирования.

4. Опишите способы хранения графических изображений в файлах.

Лабораторная работа № 5 «Работа с БД»

Цель работы

Получить навыки работы с БД через Builder. Создание «Ежедневника», с использованием MS Access.

Задание

1. Создаем БД «Ежедневник» (Planner.mdb) с одной таблицей Shedule с полями Дата и Запланированное мероприятие.
2. На форме должны быть:
   Label — текущая дата и день недели.
   Button — Сегодня, Завтра, Эта неделя, Все.
   ADOConnection — для соединения с БД.
   ADODataSet — для набора данных из БД.
   DataSource — связь между данными и их отображением на форме.
   CheckBox — если понадобится отобразить SQL команду.
   DBGrid — для отображения мероприятий.
   DBNavigator — для перемещения по данным и их изменения.
3. При запуске программа автоматически должна выводить список мероприятий на «сегодня», или при запуске в пятницу, субботу и воскресение «на сегодня и ближайшие дни». Программа должна позволять вносить изменения данных в БД (добавление, удаление и редактирование записей).

Исходный код программы и графическое представление

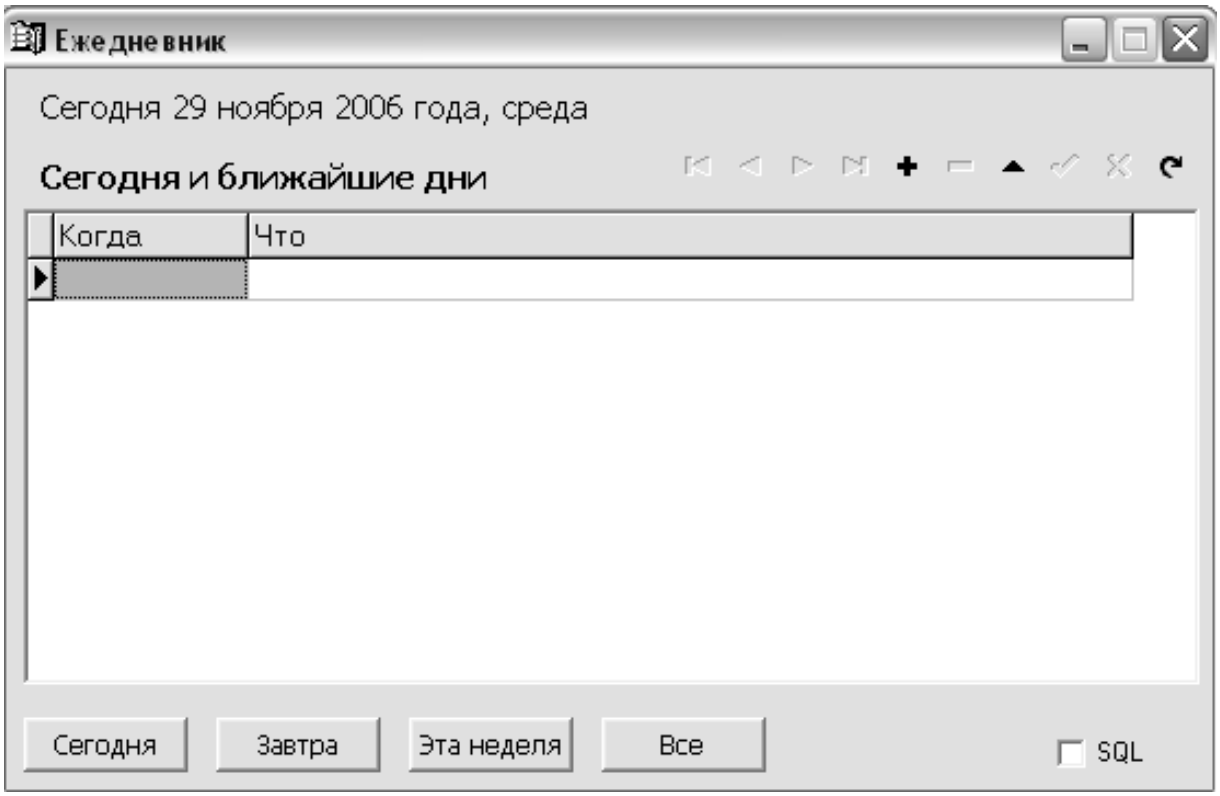
```
#include <vcl.h>
#pragma hdrstop
#include "Planner_.h"
#pragma package(smart_init)
#pragma resource "*.dfm"

TForm1 *Form1;

__fastcall TForm1::TForm1(TComponent* Owner)
   : TForm(Owner)
{
}

#include <DateUtils.hpp>
#include <ComObj.hpp>  // для доступа к EOleException

AnsiString stDay[7] = {"воскресенье","понедельник","вторник", "среда",
                       "четверг","пятница","суббота"};
```




```
AnsiString stMonth[12] = {"января","февраля","марта",
                          "апреля","мая","июня","июля",
                          "августа","сентября","октября",
                          "ноября","декабря"};

void __fastcall TForm1::FormShow(TObject *Sender)
{
      TDateTime Today,    // сегодня
          NextDay;  // следующий день (не обязательно завтра)

   Word  Year, Month, Day; // год, месяц, день

   Today = Now ();

   DecodeDate(Today, Year, Month, Day);

   Label1->Caption = "Сегодня " + IntToStr(Day) + " " +
                     stMonth[Month-1] + " " +
                     IntToStr(Year) + " года, " +
                     stDay[DayOfWeek(Today) -1];

   Label2->Caption = "Сегодня и ближайшие дни";

   // вычислим следующий день
   // если сегодня пятница, то, чтобы не забыть
   // что запланировано на понедельник, считаем, что следующий
   // день - понедельник
```

```
   switch ( DayOfWeek(Today) ) {
      case 6  : NextDay = Today + 3; break; // сегодня пятница
      case 7  : NextDay = Today + 2; break; // сегодня суббота
      default : NextDay = Today + 1; break;
   }

    ADODataSet1->CommandText =
       "SELECT * FROM schedule WHERE aDate BETWEEN DateValue('" +
       FormatDateTime("dd/mm/yyyy",Today) + "') AND DateValue('" +
       FormatDateTime("dd/mm/yyyy",NextDay) + "') ORDER BY aDate";

   // если надо отобразить SQL-команду
   if ( CheckBox1->Checked) ShowSQL();

   // если БД не зарегистрирована как источник данных ODBC,
   // возникает исключение EOleException

   try
   {
   ADODataSet1->Open(); // открыть набор данных (выподнить
                  // SQL-команду ADODataSet1->CommandText
   }

   catch ( EOleException &e)
   // чтобы тип EOleException был доступен, в программу
   // надо поместить директиву #include <ComObj.hpp>
   {

     ShowMessage ("Ошибка обращения к БД. База данных Planner.mdb должна"
              "быть зарегистрирована \n в системе как источник данных ODBC "
              "под именем dplaner");

     Button1->Enabled = false;
     Button2->Enabled = false;
     Button3->Enabled = false;
     Button4->Enabled = false;
     return;
   }

   if ( ! ADODataSet1->RecordCount )
     ShowMessage("На сегодня и ближайшие дни никаких дел не запланировано.");
}

// щелчок на кнопке Сегодня
void __fastcall TForm1::Button1Click(TObject *Sender)
{
   AnsiString today = FormatDateTime("dd/mm/yyyy",Now());

   Form1->Label2->Caption = "Сегодня";
```

```
   ADODataSet1->Close(); // закрыть набор данных

   // изменить критерий запроса
   ADODataSet1->CommandText =
       "SELECT * FROM schedule WHERE aDate = DateValue('" + today +"')";

   if ( CheckBox1->Checked) ShowSQL(); // отобразить запрос

   ADODataSet1->Open(); // открыть набор данных с новым запросом
}

// щелчок на кнопке Завтра
void __fastcall TForm1::Button2Click(TObject *Sender)
{

   AnsiString tomorrow = FormatDateTime("dd/mm/yyyy", Now() +1 );

   Label2->Caption = "Завтра";

   ADODataSet1->Close();

   // изменить критерий запроса
   ADODataSet1->CommandText =
     "SELECT * FROM schedule WHERE aDate = DateValue('" +
                 tomorrow + "')";

   if ( CheckBox1->Checked) ShowSQL();

   ADODataSet1->Open();  // выполнить запрос

   if ( ! ADODataSet1->RecordCount )
   {
      ShowMessage("На завтра никаких дел не запланировано!");
   }
}

// щелчок на кнопке На этой неделе
void __fastcall TForm1::Button3Click(TObject *Sender)
{
  // "эта неделя" – от текущего дня до конца недели (до воскресенья)
  TDateTime Present, eWeek;

  Label2->Caption = "На этой неделе";

  Present= Now(); // Now – возвращает текущую дату


  eWeek = EndOfAWeek(YearOf(Present),WeekOf(Present));

  /*
```

```
    Для доступа к StartOfWeek, EndOfAWeek, YearOf и WeekOf
    надо подключить DateUtils.hpp (см. директивы  #include )
   */

   ADODataSet1->Close();

   ADODataSet1->CommandText =
        " SELECT * FROM schedule WHERE aDate BETWEEN DateValue('" +
          FormatDateTime("dd/mm/yyyy",Present)+ "') AND DateValue('" +
          FormatDateTime("dd/mm/yyyy",eWeek)+"') ORDER BY aDate";

   if ( CheckBox1->Checked) ShowSQL();

   ADODataSet1->Open();

   if ( ! ADODataSet1->RecordCount )
       ShowMessage("На эту неделю никаких дел не запланировано.");
  }

  //  Щелчок на кнопке Все
  void __fastcall TForm1::Button4Click(TObject *Sender)
  {
     ADODataSet1->Close();

     ADODataSet1->CommandText = "SELECT * FROM schedule ORDER BY aDate";
     if ( CheckBox1->Checked) ShowSQL();

     ADODataSet1->Open();

     Label2->Caption = "Все, что намечено сделать";
  }

  // отображает SQL - команду
  void __fastcall TForm1::ShowSQL(void)
  {
     ShowMessage ( ADODataSet1->CommandText );
  }
```

Контрольные вопросы

1. Назовите основные компоненты для рабы с базой данных.
2. Укажите методы, свойства и обработчики событий компоненты DBGrid.
3. Назовите предназначение и свойства компоненты DBNavigator.
4. Раскройте технологию работы с OLE-объектами.

МЕТОДИЧЕСКИЕ УКАЗАНИЯ К ВЫПОЛНЕНИЮ КУРСОВОГО ПРОЕКТА ПО КУРСУ «ТЕХНОЛОГИИ ПРОГРАММИРОВАНИЯ»

Введение

Курсовой проект является самостоятельной работой направленной на развитие и закрепление у студентов представлений и навыков анализа, полученных в ходе изучения теоретического материла и практических занятий в предыдущем семестре. В качестве курсового проектирования предлагается выполнить комплекс работ, связанных построением приложения работающего под операционной системой Windows с применением средств быстрой разработки программ.

1 Задание на курсовой проект

1.1 Задание на курсовой проект является индивидуальным и никогда не повторяется.

1.2 Исходная задача формулируется в виде совокупности требований к разрабатываемому приложению.

1.3 Конечным результатом проекта является программа, работающая под операционной системой Windows.

1.4 По результатам проекта, на последней неделе обучения (предшествующей зачетной) проводится семинар, на котором студент докладывает о результатах полученных в ходе проектирования.

Допускается выбирать в качестве проектируемых (анализируемых) приложений анализируемые или проектируемые программы из: других курсовых проектов, НИРС, реальных производственных задач (если студент выполняет их по месту будущего распределения). Однако проектирование такого приложения должно быть сопряжено создание интерфейсного приложения, работающего под операционной системой Windows.

2 План работы

В процессе выполнения курсового проекта необходимо осуществить следующие действия:

2.1 Произвести обзор литературы по предложенной теме.

2.2 Сформулировать требования к приложению.

2.3 Разработать модель и алгоритмы.

2.4 Обосновать выбор среды реализации.

2.5 Разработать интерфейс приложения.

2.6 Осуществить кодирование программы.
2.7 Произвести тестирование и отладку.
2.8 Оформить отчет.
2.9 Оформить инструкции по установке и удалению приложения.
2.10 Оформить инструкцию пользователя.

Таблица 1 — Сроки выполнения
пунктов здания

2.1.	1 марта
2.2.	15 марта
2.3.	1 апреля
2.4.	7 апреля
2.5.	15 апреля
2.6.	1 мая
2.7.	7 мая
2.8,2.9,2.10	15 мая

2.1 Обзор литературы

Обзор литературы по заданной теме является важной составляющей курсового проектирования. Этот этап работы состоит из поиска литературы, анализа и выводов. Поиск литературы можно начинать с библиотеки ТУСУР, если достаточной литературы нет, по поиск можно расширить используя систематические каталоги и реферативные журналы или осуществить поиск в сети интернет. Необходимо четко сформулировать критерии поиска и по заданным критериям записать список литературы и краткие рефераты по заданной теме, освещенные в данном первоисточнике.

Анализ первоисточников позволяет выявить тенденции развития или противоречия в подходах решения данного класса задания. Результатом анализа обычно является некоторая таблица

Свойства, критерии, атрибуты	Источник, системы, программа

После того как представлена некоторая таблица, формируются выводы относительно задания.

2.2 Формулировка требований

Для формулировки требований можно использовать системный анализ, который обеспечивает всестороннее рассмотрение программы.

Обычно на этом этапе работ дается определение программы как некоторой системы, обеспечивающей решение некоторой проблемы. Далее строится список «заинтересованных сторон» относительно этой программы. Затем этот список сортируется по важности (значимости). Не значимые элементы удаляются. Далее подробнее рассматривается каждый из оставшихся в списке элементов. Формулируются требования к разрабатываемой системы относительно данной «заинтересованной стороны».

2.3 Разработать модель и алгоритмы

На основе выявленных требований формулируется модель. Эта модель может быть записана в виде некоторого множества формул или представлена некоторой дискретной моделью, которая записана в виде грамматики некоторого языка, графа, реляционной таблицы и т.д.

На основе полученной модели строятся алгоритмы. Форма записи может быть разная: в виде последовательности шагов, в виде блок-схемы, в виде графа переходов, в виде UML и т.д.

После записи алгоритмов производится теоретические исследования:

1) вычисляется алгоритмическая сложность;
2) определяется множество вариантов;
3) сходимость к результату с заданной точностью;
4) устойчивость полученного результата;
5) определение емкости памяти;
6) возможность распараллеливания;
7) критические участки;
8) определение исключительных ситуаций (циклы, взаимоблокировки).

2.4 Обоснование выбора среды реализации

В данном разделе необходимо обосновать выбор системы программирования

Borland C, CBuilder, Visual C++ и т.д.

2.5 Разработать интерфейс приложения

Важным элементом приложения является интерфейс. Здесь критерии должны быть следующие:

- дружественность;
- простота использования;
- использование пиктограмм;
- использование принципа «Что вижу, то и имею»;
- создание режима помощи.

2.6 Осуществить кодирование программы

На этапе кодирование производится реализация приложения средствами выбранной системы программирования. Как правило, приложение разбивается на модули, каждый модуль кодируется и отлаживается отдельно. Затем производится сборка приложения и комплексная отладка.

3 Содержание отчета

Содержание отчета должно соответствовать «методическим указаниям на выполнение курсовых и дипломных проектов и работ» доступным в библиотеке вуза и должно содержать

4 Примерные темы курсового проекта

1. Разработка компьютерного учебника по дисциплинам, связанным со специальностью 210106.
2. Разработка тренажера по обучению в дисциплинах, связанных со специальностью 210106.
3. Разработка тестовых учебных программ.
4. Разработка генераторов вопросов и заданий.
5. Разработка программ моделирования.
6. Разработка графических интерфейсов для систем моделирования.
7. Разработка интерфейсных программ с системам обработки информации.

5 Литература

1. Проектирование ресурсов Windows-приложений / Под. ред. П.В. Гусак. — Киев: Диалектика,1993. — 224 с.: ил. — ISBN 5-5-7707-5042-1.
2. Калверт Ч. Программирование в Windows: Освой самостоятельно за 21 день / Под ред. Д. Зарецкого. — М.: БИНОМ,1995. — 1008 с. — (Club Computer). — ISBN 5-89350-019-9.

3. Мешков А.,Тихомиров Ю. Visual C++ и MFC. Программирование для Windows NT и Windows 95: В 3 т. — СПб.: BHV-Санкт-Петербург, 1997. — 464 с.

4. Мюррей У., Паппас К. Создание переносимых приложений для Windows: Пер.с англ. — Спб.:BHV-Санкт-Петербург,1997. — 816 с.: ил. — ISBN 5-7791-0032-2.

5. Петзолд Ч. Программирование для Windows 95: В 2-х т.: Пер. с англ. Т.1. — Спб.:BHV-Санкт-Петербург,1997. — 752 с. — ISBN 1-55615-676-6; 5-7791-0022-9.

6. Петзолд Ч. Программирование для Windows 95; в 2-х т.: Пер. с англ. Т.2. — Спб.:BHV-Санкт-Петербург,1997. — 752 с. — ISBN 5-7791-0022-9.

7. Румянцев П.В. Азбука программирования в Win 32 API. — 2-е изд., стереотип. — М.: Радио и связь; Горячая линия — Телеком,1999. — 272 с.: ил. — ISBN 5-256-01491-9

8. Сван Том Форматы файлов Windows. — М.: БИНОМ,1995. — 288 с.: ил. — ISBN 5-7503-0014-5.

9. Финогенов К.Г. Прикладное программирование для Windows на Boland C++. — Обнинск: Принтер,1999. — 286 с.: ил

10. Фролов А.В., Фролов Г.В. Графический интерфейс GDI в MS Windows. — М.: Диалог; Мифи, 1994. — 288 с. — (Б-ка системного программиста; Т.14). — ISBN 5-86404-047-9 (Т.14); 5-86404-004-5.

11. Фролов А.В. Программирование для Windows NT. — М.: Диалог; Мифи, 1996. — 272 с. — (Б-ка системного программиста; Т.26). — ISBN 5-86404-082-7 (Т.26).

12. Фролов А.В. Программирование для Windows NT. — М.: Диалог; Мифи,1997. — 272 с. — (Б-ка системного программиста; Т.27). — ISBN 5-86404-087-8 (Т.27); 5-86404-04-5.

13. Харрис Лоуренс. Программирование OLE: Пер.с англ. — М.: БИНОМ,1995. — 464 с.: ил. — (Computer Club). — ISBN 5-7503-0056-0.

14. Хонекамп Дирк, Вилькен Петер. Введение в профессиональное программирование под Windows: Пер. с нем. — М.:ЭКОМ, 1996. — 654 с. — ISBN 5-87373-023-7.

15. Шилдт Герберт MFC: Основы программирования. — Киев: BHV-Киев,1997. — 560 с.

16. Шилдт Герберт Программирование на C и C++ для Windows 95: Пер. с англ. — Киев: BHV-Киев, 1996. — 400 с.: ил. — ISBN 5-7733-0011-7.

6 Рейтинговые оценки

№ п/п	Содержание работы	Количество балов
1.	Формализация технического задание	15
2.	Формирование математической модели алгоритма	15
3.	Исследование модели или алгоритма	10
4.	Составление программы	10
5.	Создание интерфейса	15
6.	Отладка функций	15
7.	Комплексная программы	15
8.	Тестирование	15
9.	Выступление на семинаре (публикация)	15
10.	Дополнительное задание	20
11.	Оформление пояснительной записки (оценивается корректность терминологии)	5
12.	Защита курсового проекта на комиссии	10
	Всего	150

Рейтинговая оценка может корректироваться в зависимости от сложности и качества работ по каждому пункту.

ПРИЛОЖЕНИЕ

ЭТАПЫ РАЗРАБОТКИ КОМПЬЮТЕРНЫХ УЧЕБНЫХ ПРОГРАММ

ВВЕДЕНИЕ

Компьютерные учебные программы можно отнести к сложным программным системам. Это объясняется тем, что, во-первых, учебный процесс слабо формализуем, во-вторых, сама предметная область может быть достаточно сложной для обучения и соответственно для реализации ее модели в компьютерной учебной программе, в-третьих, сложность может представлять направленность КУП для определенной группы обучаемых.

Известно, что для сложных программных систем жизненный цикл можно представить в виде шести этапов:

1. Выявление и анализ требований, предъявляемых к компьютерным учебным программам.
2. Определение спецификаций.
3. Проектирование.
4. Кодирование.
5. Тестирование и отладка.
6. Эксплуатация и сопровождение.

Рассмотрим каждый этап этого цикла.

ВЫЯВЛЕНИЕ И АНАЛИЗ ТРЕБОВАНИЙ

Выявление и анализ требований, как правило, производятся с помощью системного анализа. Возьмем за основу методику системного анализа. Эта методика первоначально предполагает выявление всех «заинтересованных сторон» — участников проблемной ситуации. На рис. 1 представлены основные целеполагающие системы для КУП.

ТРЕБОВАНИЯ СО СТОРОНЫ ОБУЧАЕМОГО

Это прежде всего «обучаемый» — конечный пользователь КУП. Требования со стороны обучаемого можно разделить на три группы:

– психолого-педагогические;
– инженерно-психологические;
– медицинские.

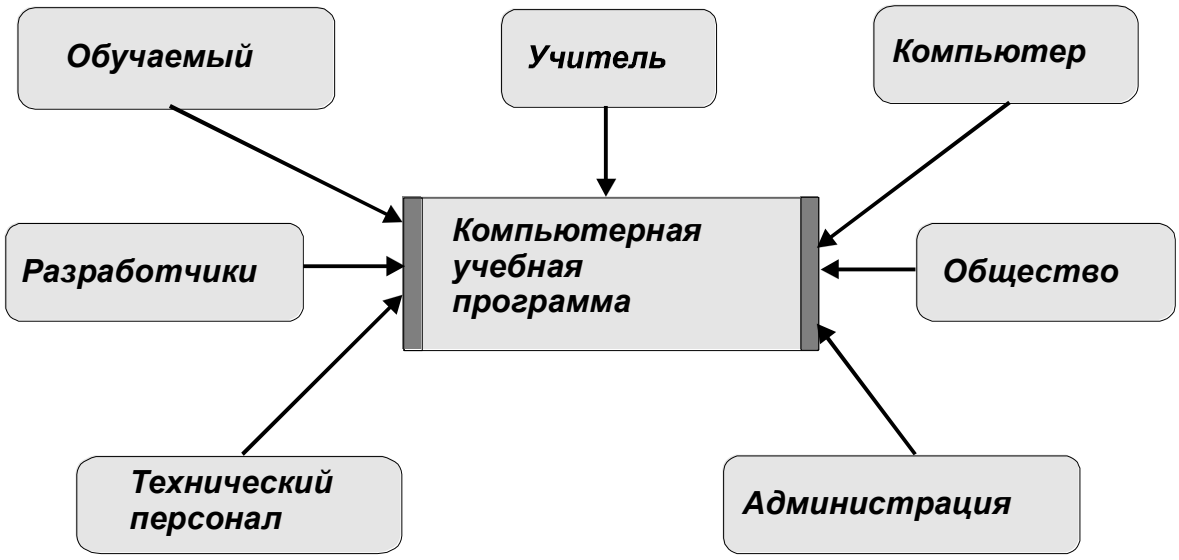


Рис. 1 — Основные целеполагающие системы

Психолого-педагогические требования в целом определяют эффективность учебного процесса. Важнейшим требованием здесь является требование «компьютерная учебная программа должна научить». При этом:

1. КУП должна адаптироваться к физиологическим и психологическим особенностям обучаемого (память, темперамент, реакция, физическое и умственное развитие, возраст, зрение, слух).

2. КУП должна быть основана на деятельностном подходе в формировании психики, эрудиции и нравственных качеств.

3. КУП должна обеспечить постоянную и положительную мотивацию деятельности обучаемого.

4. КУП должна использовать комбинированные приемы обучения, которые развивают и используют как абстрактно-логическое, так и образно-эмоциональное мышление, интуицию обучаемого.

5. КУП должна впитывать в себя последние достижения в области педагогических наук.

Инженерно-психологические требования определяют интерфейс между обучаемым и КУП. Здесь требования будут следующие:

1. Простота работы с КУП.

2. Дружелюбность интерфейса.

3. Приспособление к требованиям конкретного обучаемого, (например, настройка цвета и шрифта текста, возможность увеличения шрифта).

4. Организация комфортного интерфейса.

Медицинские требования определяют факторы КУП, которые влияют на здоровье обучаемого. Эти требования не только определяют влияние компьютера на обучаемого, но и влияние самой КУП. Прежде всего это касается зрения, психики и нервной системы. Некоторые требования, напри-

мер время обучения с помощью некоторой КУП для разных групп учащихся, определяются федеральными санитарными правилами, нормами и гигиеническими нормативами.

ТРЕБОВАНИЯ СО СТОРОНЫ УЧИТЕЛЯ

Учитель непосредственно организует обучение предмета с помощью готовой КУП. Запускает при необходимости программу, наблюдает за ходом работы учащегося, приходит на помощь при возникновении трудностей. Регистрирует текущие успехи учащегося. Основные требования со стороны учителя следующие:

1. Обеспечение различных форм организации работы с классом — от коллективной до полностью индивидуальной с каждым учащимся.
2. Обеспечение различных видов связи учителя с обучаемыми: электронная почта, доски объявлений, переадресация учащегося к учителю для личного контакта; вмешательство учителя в ход обучения на любой стадии, связь со всеми обучаемыми или с каждым в отдельности; возможность негласного контроля.
3. Различные формы накопления опыта: протоколирование процесса обучения; статистический анализ; регистрация востребуемости тех или иных разделов КУП.
4. Возможность внесения изменений в КУП (по крайней мере, адаптацию КУП для конкретного вида обучения).

ТРЕБОВАНИЯ АДМИНИСТРАЦИИ

Важнейшим требованием администрации является повышение эффективности процесса обучения с использованием компьютерных учебных программ. Здесь эффективность толкуется в самом общем смысле. Т.е. это может быть сокращение прямых и косвенных затрат на образование, или повышение качества обучения, или создание комфортной творческой атмосферы педагогического коллектива. Кроме указанных были выявлены следующие требования: информационное обеспечение административных функций (сбор данных и статистический анализ, составление отчетов); соблюдение требований стандартизации и унификации.

ТРЕБОВАНИЯ ТЕХНИЧЕСКОГО ПЕРСОНАЛА

Основным требованием со стороны технического персонала является снижение затрат на эксплуатацию разрабатываемой КУП. Это требование предусматривает:

1) простоту запуска и настройку разрабатываемой КУП;
2) минимизацию объемов требуемой памяти;

3) минимизацию времени выполнения;
4) использование стандартных технических устройств.

ТРЕБОВАНИЯ РАЗРАБОТЧИКОВ

В процессе разработки компьютерной учебной программы выделяются три основные составляющие: психолого-педагогическая, организационно-экономическая и техническая. Психолого-педагогическая составляющая обеспечивает педагогические цели и методы достижения их в разрабатываемой КУП. Организационно-экономическая составляющая обеспечивает реализацию и тиражирование данной КУП при заданных финансовых, трудовых и временных ограничениях. Техническая составляющая собственно реализует КУП в виде программного, информационного и иных обеспечениях.

«Разработчики» — довольно большой коллектив специалистов, состоящий из: программистов, психологов, методистов, художников, музыкантов, звуковых режиссеров, мультипликаторов, сценаристов, экономистов, менеджеров по маркетингу и рекламе, юристов, медиков, специалистов по тестированию, организаторов. Условно всех разработчиков можно разделить на 11 групп (см. рис. 2).

Рис. 2 — Основные группы разработчиков компьютерных учебных программ

1. Группа методистов определяет цели, содержание и этапы обучения с помощью разрабатываемой КУП, основываясь на последних достижениях психолого-педагогической науки и практики. Определяет педагогическую технологию обучения /69/. Эта группа формирует учебный материал для представления ее в КУП, разрабатывает контрольные вопросы и задания, определяет алгоритм оценивания знаний. В состав этой группы входят высококвалифицированные эксперты в области преподавания данного предмета (раздела знаний), обладающие большим педагогическим опытом.

2. Группа психологов обеспечивает разработку разнообразных тестов состояния здоровья, тестов способностей и достижений. Также эта группа решает вопросы разработки эффективного интерфейса между обучаемым и КУП.

3. Группа программистов непосредственно реализует КУП. Программистов можно разделить на системных и прикладных. Системные программисты обеспечивают интерфейс КУП с вычислительной системой конкретного класса компьютера. Прикладные программисты обеспечивают разработку отдельных модулей КУП.

4. Художественная группа обеспечивает компьютерное представление учебного материала. При этом решаются следующие задачи: выбор или разработка шрифтов для представления текстовой учебной информации. Ввод, редактирование и форматирование текста. Разработка, ввод и редактирование иллюстраций представленных в графической форме. Создание компьютерной анимации и визуальных эффектов. Запись и редактирование видеоклипов или видеофильмов.

5. Музыкальная группа обеспечивает музыкальное сопровождение, разработку и ввод различных звуковых эффектов в КУП. Сопровождение подачи учебного материала голосом диктора.

6. Медицинская группа обеспечивает определение параметров КУП, влияющих на здоровье учащихся. Это прежде всего, предполагаемое время работы с данной КУП.

7. Экономическая группа обеспечивает экономическую обоснованность и целесообразность разработки КУП, производит поддержку финансового состояния предприятия при реализации данного проекта.

8. Группа маркетинга и рекламы обеспечивает разработку стратегии рекламной компании, разработку рекламных материалов (роликов, демонстрационных версий, рекламных листков и писем и др.), определение рынков сбыта и цен на разрабатываемую КУП, работу по заключению дилерских соглашений на тиражирование проектируемой КУП.

9. Юридическая группа обеспечивает правовую поддержку, которая включает разработку вопросов правовой защиты всех заинтересованных сторон: обучаемого, учителя, администрации и разработчиков.

10. Тестирующая группа — довольно большая группа разнообразных специалистов, производящая тестирование разрабатываемой КУП. Прежде всего, это преподаватели, имеющие опыт работы с подобными программами.

11. Группа управления проектом обеспечивает управление, планирование и координацию отдельных этапов разработки компьютерной учебной программы. В эту группу входят руководитель проекта, его помощники по отдельным направлениям, технические работники.

КОМПЬЮТЕР

Развитие компьютерной техники привело к ситуации, когда на рынке компьютеров для образования осталось всего два типа. Это IBM PC – подобные и Макинтоши фирмы Apple. Основным требованием для компьютера, предназначенного для обучения, является оснащение его средствами мультимедиа. Другим важным требованием является возможность подключения компьютера в компьютерную сеть Интернет.

Еще одним важным аспектом с точки зрения реализации КУП является выбор программной платформы. В настоящее время имеется достаточно много разнообразных операционных систем: MS DOS, Windows 3.1, Windows-95, Windows-NT, Unix, OS/2 и т.д. В настоящее время одной из самых распространенных операционных систем, используемых в образовании, является ОС Windows-95.

Следует также отметить, что развитие компьютерной техники для образовательных целей не стоит на месте и в недалеком будущем появятся принципиально новые классы компьютеров. Прообразом таких компьютеров являются компьютеры-блокноты. Главные черты новых компьютеров:

1. Основным устройством ввода будет ручка типа шариковой. Здесь компьютер будет распознавать почерк обучаемого.

2. Экран компьютера будет расположен не вертикально, а горизонтально. Таким образом, компьютер будет лежать на парте как обычная тетрадь, в которой можно писать и рисовать.

3. В компьютере будет реализован речевой ввод и вывод информации.

ТРЕБОВАНИЯ СО СТОРОНЫ ОБЩЕСТВА

В целом общество также выдвигает ряд требований к разрабатываемой КУП. Это прежде всего требования соблюдения безопасности использования знаний. Т.е. знания, полученные с помощью данной КУП, не должны использоваться во вред людям и природе. Другим важным требованием является повышение культурного уровня обучаемого. Т.е. обучае-

мый должен получать не только чистые знания по данной конкретной теме, но и выдающиеся примеры приобретения или применения этих знаний.

СОСТАВЛЕНИЕ СПЕЦИФИКАЦИИ

После выявления требований необходимо провести их анализ и записать спецификацию на разрабатываемую компьютерную учебную программу. Здесь под спецификацией будем понимать описание компьютерной учебной программы, по которому производится ее проектирование/70/. Это описание может быть составлено на естественном языке, или может быть использован некоторый формализованный язык.

В спецификации должно быть записано: платформа и тип компьютера, определен подход к проектированию компьютерной учебной программы, определены виды представления учебного материала (например, использует ли данная КУП аудио- или видео- учебную информацию). Если в данной КУП предусматриваются тесты, то определяются тип вопроса, способы ввода ответа и его анализа, определяются способы оценивания ответов и их фиксации и т.д.

ПРОЕКТИРОВАНИЕ

Этап проектирования предполагает разработку алгоритмов и информационной базы, для компьютерного представления учебного материала. На этом этапе готовится учебный материал определяется его структура, пишется текст, готовятся иллюстрации, подготавливается видеоматериал, подбирается аудиоматериал. Для тестовых программ готовятся вопросы, разрабатываются алгоритмы анализа и оценивания ответов. Разрабатываются алгоритмы представления и предъявления учебной информации. Ниже будут подробно рассмотрены конкретные методы проектирования для определенного класса компьютерных учебных программ.

КОДИРОВАНИЕ

На этапе кодирования непосредственно создается компьютерная учебная программа. Кодирование включает создание информационной базы, состоящей из текста, иллюстраций, анимации, аудио- и видеоинформации. Причем вся эта учебная информация связана в виде некоторой информационной сети. Например, отдельные фрагменты текста, иллюстраций и т.д. представлены в виде отдельных компьютерных страниц (кадров), которые в свою очередь могут быть связаны с другими кадрами.

Алгоритмы, разработанные на этапе проектирования, реализуются программно с использованием некоторой инструментальной среды.

На данном этапе производится комплексная сборка всех модулей КУП и отладка программы.

ТЕСТИРОВАНИЕ

Тестирование является необходимым и важным этапом создания КУП. На данном этапе производится проверка функционирования всех составных частей КУП, определяются реальные характеристики. Первоначально тестируются отдельные модули программы. Далее производится тестирование информационного обеспечения. Производится проверка представления и предъявления информации с точки зрения обучения данному предмету, производится определение корректности вопросов и способов их оценивания. Проверяется связанность учебной информации (например, все ссылки должны быть на реальные кадры). Далее производится проверка режима ожидания ввода, т.е. всех ситуаций ввода информации со стороны обучаемого.