МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ
«БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

КАФЕДРА ИНТЕЛЛЕКТУАЛЬНЫХ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

к лабораторным работам по дисциплине
Проектирование программ в интеллектуальных системах
"Разработка windows-приложений"
для студентов специальности «Искусственный интеллект»

БРЕСТ 2015

УДК 681.3 (075.8)
ББК с57

Целью указаний является информационное и методическое обеспечение цикла лабораторных работ по основам разработки оконных и консольных приложений на языке Visual C++ в Microsoft Visual Studio, применению библиотек MFC, STL, особенностям языка C++/CLI. Приведенные в указаниях наборы действий характерны для "базовых" вариантов Studio (без использования NET) и соответственно могут иметь свои особенности в разных версиях Studio. Работа с языком C++/CLI ведется в NET-версиях Studio. Все задания указаний, помеченные символом *, а также контрольные задания выполняются по указанию преподавателя.

Составители: Муравьев Г.Л., доцент, к.т.н.;
Мухов С.В., доцент, к.т.н.;
Савицкий Ю.В., к.т.н., доцент

ЛАБОРАТОРНАЯ РАБОТА
"Каркасы windows-приложений. Типовой каркас приложения (ТКП)"

ЦЕЛЬ РАБОТЫ: 1. Ознакомиться со структурой и особенностями программирования оконных windows-приложений. 2. Изучить каркасы windows-приложений. 3. Изучить структуру типового каркаса оконных приложений (ТКП), использование ТКП для создания пользовательских приложений.

СОСТАВ ОТЧЕТА: 1. Описание каркасов windows-приложений (Empty, Simple или аналогичных): характеристика интерфейса, предоставляемых возможностей; файловый состав (структура проекта, дерево папок, состав, назначение файлов и их соподчиненность по включению); функциональный состав (схема иерархии функций приложения, назначение и прототипы функций). 2. Аналогичное описание ТКП. Диаграммы прецедентов, состояний, компонентов. 3. Выводы: а) о последовательности инициализации ТКП; б) о причинах и условиях посылки сообщений ТКП, рассмотренных в работе; в) перечень ситуаций "перерисовки"; г) выводы по способу реализации завершения работы приложения.

ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ. Работа выполняется по пособию [1]. Все практические действия выполняются в системе Visual Studio.

1. Изучить теоретический материал об особенностях оконных приложений (§§ 1.1-1.3, 2, 3.1, 3.2). Обратить внимание на понятие "оконное приложение" в § 1.1, структуру приложения в § 3.1, функции WinMain и обработчики в § 3.2.

Работа с каркасами Empty и Simple

2. Создать каркас windows-приложения (тип Empty или аналогчный) средствами мастера (например, Win32 Application). Изучить его свойства (состав интерфейса; файловый состав; функциональный состав). Для этого в системе Visual Studio создать новый проект (New Project): выбрать пункт меню File-New, в появившемся диалоговом окне выбрать вкладку проекты Projects и выбрать пункт Win32 Application, т.е. создание windows-приложения; набрать имя проекта в строке Project name, установить переключатель типа проекта в An empty project (пустой проект), завершить создание каркаса (дерево проекта можно увидеть в окне рабочего пространства – вкладка FileView !) и запустить приложение.

3. Создать проект из одного файла с одной, главной функцией, выводящей сообщения с помощью функций MessageBox. Для этого доработать каркас – создать файл в папке Source Files проекта приложения: выбрать пункт меню File-New, в появившемся диалоговом окне выбрать вкладку файлы Files и пункт C++ Source File, т.е. создание файла с текстом на языке C++; набрать имя файла в поле File name и завершить его создание; включить в созданный файл заголовок #include <windows.h>, создать пустую функцию WinMain, например, как показано ниже и выполнить приложение

```
int  WINAPI  WinMain (HINSTANCE H1, HINSTANCE H2, LPSTR Str, int I)
{ return 0; }   ;
```

- включить в приложение вывод окна сообщения MessageBox (1.4.3 [2]). При необходимости исправить неточности! Соответствующую справочную информацию можно по-

лучить по F1 в разделе CWindow::MessageBox – Microsoft Foundation classes ..., либо выделить строку – MessageBox и нажать клавишу F1. При этом, как правило, появится вкладка "найденные разделы", где следует уточнить, что ищется. В данном случае это API-функция. При работе с классами это может быть CWnd::MessageBox в разделе MFC and Template... . Функция MessageBox имеет прототип

РезультатВыбора **MessageBox** (ДескрипторРодительскогоОкна, Сообщение, НазваниеОкнаСообщения, СоставКнопокОкнаСообщения) .

Здесь состав кнопок задается кодами MB_OK, MB_OKCANCEL, MB_YESNOCANCEL и другими или их числовыми эквивалентами, а тип возвращаемого результата зависит от нажатой кнопки и кодируется ее дескриптором как IDYES, IDOK, IDCANCEL и т.д. Например, включить последовательно

MessageBox (NULL, "Работает приложение", "Сообщение1", 1),
MessageBox (hWnd, "Работает приложение", "Сообщение2", MB_OK),
MessageBox (hWnd, "Работает приложение", "Сообщение3", MB_YESNOCANCEL) .

Для каждой новой вставки снова выполнить приложение. Исправить ошибки.

Для иллюстрации использования средств из библиотеки классов MFC включить в приложение функцию **AfxMessageBox**, для чего заменить заголовочный файл <windows.h> на файл <afxwin.h>, выполняющий роль шлюза для доступа к классам MFC, и подключить библиотеки через главное меню – Project-Settings-General-Use MFC. Просмотреть справочную информацию по прототипу функции, включить в приложение вывод окна сообщения, например,

AfxMessageBox ("Вот ОНО и заработало !", 1)

и выполнить приложение. Проанализировать возвращаемый код для одного из MessageBox (с двумя и более кнопками) – выводить сообщение о типе нажатой кнопки.

4. Создать проект из двух файлов – главный (с расширением cpp) с текстом программы – функцией WinMain, заголовочный (с расширением h) с командами препроцессора. Для этого повторить п.3, модифицировать приложение, заголовочный файл подключить к главному командой препроцессора #include "*.h". Выполнить приложение.

5. Создать каркас Windows-приложения (тип Simple или аналогичный) средствами мастера (аналогично п.2). Изучить его свойства: интерфейс; файловый состав; функциональный состав; ресурсный состав.

6. Создать проект, модифицировав созданный каркас путем вставки в WinMain функций MessageBox (аналогично п.3).

Работа с ТКП

7. Изучить теоретический материал о сообщениях и типовом каркасе приложения (§§ 4.1-4.3, 5.4). Изучить сообщения (§ 4.3). Создать приложение на базе типового каркаса приложения ТКП, взяв за основу автоматический каркас типа Simple (или аналогичный). Изучить его свойства: интерфейс; файловый состав; функциональный состав; ресурсный состав. Для этого: создать каркас проекта, тип Simple (см. п.5); в основной файл проекта (*.cpp) вместо имеющегося там текста вставить текст ТКП (см. § 5.4 пособия), сохранив только команду #include "stdafx.h"; выполнить приложение.

8. Модифицировать приложение, последовательно размещая MessageBox с сообщением о выполненной функции: а) после создания окна в CreateWindow; б) после вывода и после обновления окна UpdateWindow (hWnd); в) сразу в секции case WM_DESTROY; г) в секции case WM_PAINT в место для вставки фрагмента пользователя.

Для каждого варианта вставки осуществить запуск приложения и рассмотреть его работу (вывод сообщений) при свертывании-развертывании окна, перекрытии, перемещении окна, попытке изменения размеров окна.

Закомментировать case WM_DESTROY: PostQuitMessage(0); . Выполнить несколько запусков ТКП и убедиться, что после закрытия приложений – они по-прежнему активны – но без окон – присутствуют в списке процессов компьютера!!! Разработать автоматную диаграмму приложения.

9. Повторить создание приложения (п. 7). Для этого: создать каркас проекта, тип Simple (см. п.5); в основной файл проекта (*.cpp) вместо имеющегося там текста вставить текст ТКП (см. приложение 7 [1]), сохранив только команду #include "stdafx.h"; выполнить приложение и при необходимости устранить ошибки, вызванные присутствием в тексте недоопределенных элементов. Вставить функции MessageBox.

10. Изучить теоретический материал по созданию окна приложения (§§ 4.8, 4.9). Создать приложение, модифицировав приложение на базе ТКП. Изменить его интерфейс (вид окна, состав элементов окна – кнопок и т.п.). Для этого: создать приложение на базе ТКП (см. п.7); выполнить отмену опции окна – WS_OVERLAPPEDWINDOW и обеспечить тот же состав окна с применением опции окна – WS_OVERLAPPED путем поэлементного включения соответствующих элементов (кнопок и т.п.) через оператор "или" (см. приложение 5 [1]); выполнить отмену опции окна ТКП – WS_OVERLAPPED и обеспечить тот же состав окна, но путем поэлементного включения соответствующих элементов (кнопок и т.п.).

Внести изменения в параметры: WNDCLASS (см. приложение 1 [1], приложение 2 [1], приложение 3 [1], приложение 4 [1]); функции CreateWindow (см. приложение 5[1]). Например, добавить в окно рамки, полосы прокрутки, убрать или добавить командные кнопки, внести изменения в координаты и размеры окна, использовав соответствующие параметры функции (x, y, Width, Height); функции ShowWindow (см. приложение 6 [1]).

11. Обеспечить попеременное переключение окна – "спрятано", "показано", "свернуто", "развернуто".

КОНТРОЛЬНЫЕ ЗАДАНИЯ

1. Создать приложение с ТКП "вручную", не используя мастер, поменяв размеры окна (200 на 200), заголовок окна, добавив полосы прокрутки.

2. Добавить к приложению обработчик сообщения WM_QUIT – осуществить вывод окна сообщения MessageBox о пришедшем сообщении.

3. Повторить п.1 в заданной версии Visual Studio.

4. Обеспечить попеременное переключение окна ТКП ("спрятано"-"показано"), использовав для этого обработчик сообщения WM_LBUTTONDOWN – нажатие левой клавиши "мыши" или нажатие клавиши на клавиатуре.

ЛАБОРАТОРНАЯ РАБОТА
"Разработка приложений на базе ТКП. Организация вывода"

ЦЕЛЬ РАБОТЫ: 1. Ознакомиться с особенностями организации вывода в оконных windows-приложениях. 2. Ознакомиться с особенностями управления сообщениями.

СОСТАВ ОТЧЕТА: Описание приложений (п.п. 9-11): состав интерфейса и его возможности; файловый состав (структура проекта, дерево папок, состав, назначение файлов и их соподчиненность по включению); функциональный состав (привести схему иерархии функций приложения, описать назначение и прототипы функций). Диаграммы прецедентов, состояний, компонентов, классов. Соответствующие листинги.

ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ. Работа выполняется по методическим пособиям [1, 2]. При разработке приложений следует учитывать: 1. Особенности использования контекста устройства (КУ – HDC hdc) – системного ресурса для хранения атрибутов объектов, связанных с рисованием. Используется в приложениях для вызова функций GDI. Как правило, КУ выделяется только в одном месте программы. Например, создание hdc = BeginPaint(hWnd, &ps) и разрушение EndPaint(hWnd, ps). 2. Особенности ввода-вывода данных, где, как правило, используется строковый тип. Для преобразований типов данных применяют любые подходящие функции, например sscanf, sprintf из библиотек С, или аналогичные функции (см. § 1.4.1.Преобразование типов данных [2]).

Вывод текстовых данных в клиентскую область окна

1*. Создать ТКП, описать схему иерархии модулей. Выполнить контрольные задания.

2. Изучить теоретический материал (§ 1.4.3 [2]).

3. Создать приложение на базе ТКП. Добавить строку приветствия (координаты вывода – 0, 0), для чего вставить в обработчик WM_PAINT пользовательский фрагмент

```
char  szText [25] =  "Работает ТКП" ;
.....
TextOut ( hdc, 0, 0, szText, strlen ( szText ) );
```

Организовать вывод данных, например,

```
int  X = 2007;  char  szXasString[100];
.....
wsprintf ( szXasString, "Значение X - %d", X );
TextOut ( hdc, 0, 130, szXasString, strlen ( szXasString ) );  .
```

Вывести в столбик для значений X от 1 до 10 (с шагом 0,5) пары значений: X – квадрат X. Запустить приложение. Выполнить свертывание-развертывание окна, перемещение, изменение размеров окна. Убедиться в автоматической перерисовке окна.

4. Модифицировать предыдущее приложение. Для этого: изменить формат вывода и увеличить число обрабатываемых чисел до 75; добавить вывод номера перерисовки (переменная int ReDrawNumber = 0). Вид окна приведен на рисунке 1.

5*. Создать приложение для табулирования заданной функции. Значения вывести в виде таблицы. Параметры – левая, правая границы, число точек табулирования.

6*. Модифицировать предыдущее приложение, организовав вычисления на базе пользовательских классов.

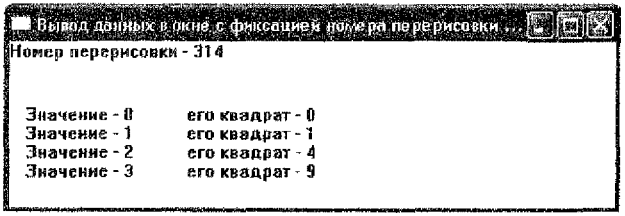

Рисунок 1 – Фрагмент интерфейса

Вывод графической информации в клиентскую область окна

7. Создать приложение на базе ТКП. Организовать вывод графических данных. Основные графические примитивы рисуются с использованием библиотечных функций: дуги эллипса Arc, ArcTo; эллипсы и окружности Ellipse; линии от текущей точки до точки, указанной в функции LineTo(hdc, x, y); прямоугольники Rectangle; полигоны Polygon; связанные отрезки прямых PolyLine. Также используют функцию рисования точки COLORREF SetPixel (hdc, x, y, COLORREF crColor), функцию перемещения курсора к указанной точке с сохранением координаты текущей точки в структуре типа LPPOINT – BOOL MoveToEx(hdc, x, y, LPPOINT pPoint). Фрагмент использования функций приведен ниже

```
                                    ...
case WM_PAINT:
        hdc = BeginPaint( hWnd, &ps );
        hPen = CreatePen(PS_SOLID,1,RGB (0, 255, 0) );
        SelectObject (hdc, hPen);
        Ellipse ( hdc, 100, 100, 300, 300 );
        MoveToEx(hdc, 50, 50, NULL);
        LineTo(hdc, 400,400);
          .....
        ValidateRect(hWnd,NULL);
        EndPaint(hWnd, &ps);
break;  ...
```

Обработка сообщений

8. Создать приложение на базе ТКП. Включить чувствительность к нажатию левой клавиши "мыши" (сообщение WM_LBUTTONDOWN) – выводить окно MessageBox. Для этого создать в функции-обработчике дополнительную секцию

```
case WM_LBUTTONDOWN:
      // < ФРАГМЕНТ ПОЛЬЗОВАТЕЛЯ >
break;          ,
```

аналогичную, например,

```
case WM_PAINT:    .....    break;   .
```

Повторить п.п. 3-4, выполняя все действия в секции case WM_LBUTTONDOWN (секция case WM_PAINT должна быть пустой). Запустить приложение. Выполнить свертыва-

ние-развертывание окна, перемещение, перекрытие, изменение размеров окна. Проанализировать характер пере овки содержимого окна.

9. Создать приложение, в котором подсчитываются события – нажатие левой клавиши мыши. А результат – число нажатий – выводится в секции WM_PAINT только по нажатии правой клавиши мыши. Разработать автоматную диаграмму.

9.1. Модифицировать приложение. Результат выводить в секции WM_PAINT сразу при его обновлении. Для этого инициировать пере овку окна функцией InvalidateRect (hWnd, NULL, TRUE). Разработать автоматную диаграмму.

10. Создать приложение для вывода текста "Работает ТКП", начиная с позиции X, Y. За начальное значение координат взять 0, 0. По сообщению WM_LBUTTONDOWN увеличивать координаты точки вывода на 50 единиц и инициировать пере овку. По сообщению WM_RBUTTONDOWN уменьшать координаты точки вывода на 50 единиц и инициировать пере овку. Разработать автоматную диаграмму.

11. Создать приложение для вывода координат указателя "мыши" в виде X = ..., Y = ... За начальное значение координат взять 0, 0. По сообщению WM_LBUTTONDOWN считывать текущие координаты курсора мыши. По сообщению WM_RBUTTONDOWN инициировать вывод координат. Обеспечивать пере овку окна. Блокировать обновление вывода по другим причинам пере овки. Разработать автоматную диаграмму.

КОНТРОЛЬНЫЕ ЗАДАНИЯ 1. Создать приложение на базе ТКП. Выводить символ, задаваемый с клавиатуры, в координатах указателя "мыши", задаваемых "щелчком" ее левой клавиши. Начальное значение координат взять как X = 0, Y = 0. Обеспечивать пере овку и обновление окна как при изменении координат так и при вводе нового символа. Для хранения данных использовать иерархию классов ДАННЫЕ, СИМВОЛ, КООРИНАТА. Предварительно разработать и представить диаграмму классов и диаграмму состояний приложения!

2. Создать windows-приложение и повторить п. 3, использовав при выводе данные реальной длины строк, а также данные об интервалах между строками (см. Справочные данные следующей работы).

СПРАВОЧНЫЕ ДАННЫЕ

Сообщение WM_LBUTTONDOWN содержит координаты курсора "мыши" момента нажатия левой клавиши. Соответственно флаги ввода содержатся в wParam; координаты указателя X, Y в LOWORD(lParam) и HIWORD(lParam).

Сообщение WM_CHAR содержит код нажатой клавиши (TCHAR) wParam – результат сообщения WM_KEYDOWN, обработанный функцией TranslateMessage.

ЛАБОРАТОРНАЯ РАБОТА "Каркасы. Создание интерфейсов. Диалоговые окна"

ЦЕЛЬ РАБОТЫ. 1. Изучить создание Windows-приложений с использований каркаса Hello (или аналогичного каркаса). 2. Изучить технологию работы с диалоговыми окнами и элементами управления (ЭУ). Применением диалогового окна в составе главного, в качестве главного окна. 3. Изучить использование кнопок, окон редактирования.

СОСТАВ ОТЧЕТА. 1. Описание каркаса Hello: состав интерфейса; файловый состав (структура проекта, папок, состав, назначение файлов и их соподчиненность по включе-

нию); функциональный состав (схема иерархии функций приложения, назначение и прототипы функций). Диаграммы состояний, компонентов. 2. По заданиям п. 5, 6: диаграммы состояний; листинги. 3. По заданиям п. 8, 11: диаграммы состояний.

ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ. Работа выполняется по пособиям [2, 3].

1*. Выполнить контрольные задания.

Каркас HELLO

2. Изучить теоретический материал – каркас Hello (§ 4.1.1-4.1.3 [3]).

3. Создать каркас Hello средствами мастера. Изучить его свойства: интерфейс; файловый, функциональный, ресурсный состав.

4. Создать каркас. Изменить интерфейс приложения (§ 4.2.1-4.2.3 [3]): изменить названия пунктов меню (вкладка Resource); поменять название главного окна на свою фамилию; изменить текст в окне-справке; по нажатию в окне-справке кнопки ОК выводить окно MessageBox с сообщением выхода из окна-справки; по завершении работы с приложением выводить окно MessageBox с соответствующим предупреждением.

5. Повторить п. 9 работы "Разработка приложений на базе ТКП. Организация вывода".

6. Создать приложение для вывода в клиентское окно массива строк ("Use", "function", "prototype","Get","Text","ExtentPoint( )") в виде

Use
function prototype
GetTextExtentPoint( )

Для этого использовать данные реальной длины строк и данные об интервалах между строками (см. раздел Справочные данные).

Диалоговые окна

7. Изучить теоретический материал по работе с графическими ресурсами приложений (§ 1.2, § 1.3 [2]).

8. Использование диалогового окна в составе главного (см. § 2.1.1 [2]).

ЗАДАНИЕ (демонстрационный пример). *Создать приложение, при запуске которого в качестве главного выводится масштабируемое окно с рамкой и клиентской областью, а за ним сразу же автоматически диалоговое окно с кнопками. При этом, главное окно остается на экране, но теряет активность. Нажатие кнопок диалогового окна приводит к его закрытию. Примерный вид интерфейса приведен ниже.*

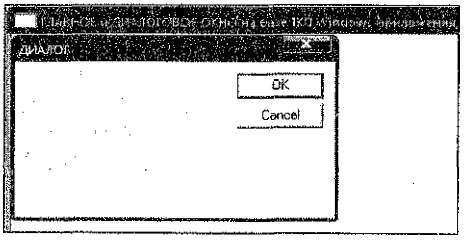

Рисунок 2 – Оконный интерфейс

Для этого: 8.1. Изучить теоретический материал и разработать диаграмму состояний. 8.2. Воспроизвести пример. 8.3. Модифицировать приложение – добавить вывод подтверждения нажатия кнопок окна. 8.4. Модифицировать приложение – разрушать диалоговое окно только при нажатии кнопки Cancel. 8.5. Модифицировать приложение – при завершении работы с диалоговым окном разрушать и главное окно.

9. Использование диалогового окна в качестве главного (§ 2.1.2 [2]).

ЗАДАНИЕ (демонстрационный пример). *Создать приложение, при запуске которого в качестве главного выводится диалоговое окно. Нажатие его кнопок приводит к выводу соответствующих сообщений, а нажатие Cancel – к его закрытию.*

Элемент управления окно редактирования

10. Изучить теоретический материал по использованию ЭУ (§1.4.2 [2]).

11. Диалоговое окно с ЭУ редактирования в качестве главного (§ 2.1.3 [2]).

ЗАДАНИЕ (демонстрационный пример). *Создать приложение, при запуске которого в качестве главного должно выводиться диалоговое окно с окнами редактирования, подсказками, кнопкам), предназначенное для задания исходных данных и их эхо-вывода. По кнопке Ввести производится ввод из поля ввода и эхо-вывод введенного значения.*

Вид диалогового окна приведен ниже. Окно включает окна редактирования (Edit Box), статичные окна (Static Text), кнопку Ввести.

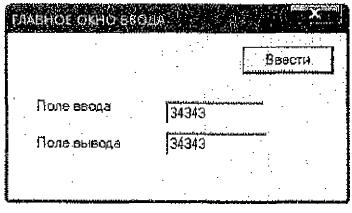

Рисунок 3 – Фрагмент интерфейса

Для этого: 11.1. Изучить теоретический материал и разработать диаграмму состояний. 11.2. Воспроизвести демонстрационный пример.

КОНТРОЛЬНЫЕ ЗАДАНИЯ

1. Разработать приложение на базе каркаса Hello для табуляции функции и вывода результатов в клиентскую область (КО) главного окна (ГО).

2. Разработать приложение с интерфейсом из двух окон – главного окна и диалогового (ДО), запускаемого щелчком правой клавиши мыши. Состав ДО: кнопка Отказ для его разрушения и возврата в ГО. Для этого прототипировать приложение, спроектировать и описать: интерфейсные формы (окна); диаграммы пользовательских классов (при использовании классов); диаграмму состояний интерфейсных форм.

3. Разработать приложение (на базе предыдущего) с интерфейсом из двух окон – ГО и ДО, запускаемого щелчком правой клавиши мыши. Состав ДО: ЭУ для ввода массива. При завершении ввода выводить массив в КО главного окна. Все действия с данными (массивом чисел) описать, используя свой – пользовательский класс (классы). Для этого прототипировать приложение – спроектировать и описать: интерфейсные формы; диаграмму классов; диаграммы состояний интерфейсных форм.

СПРАВОЧНЫЕ ДАННЫЕ

При выводе данных в клиентской области окна требуется вычислять координаты начальной позиции для вывода первого символа каждой строки, а при переходе на новую строку окна вычислять новое значение координаты y. Для определения новой координаты X, после вывода текущей строки, надо использовать метод MFC CDC::GetTextExtent() вычисления ее реальной длины в логических единицах

CSize CDC::GetTextExtent(LPCTSTR lpszString, int nCount) const;
CSize CDC::GetTextExtent(const CString &str) const;

Для строки lpszString из nCount = strlen(lpszString) символов метод возвращает ее длину в виде объекта класса CSize, унаследованного от структуры SIZE

typedef struct tagSIZE { int cx; int cy; } SIZE

В API используется аналогичная функция

GetTextExtentPoint(HDC hdc, LPCTSTR lpszString, int nCount, LPSIZE lpSize) ,

где lpSize – указатель на результат – структуру типа SIZE.

Для перехода на новую строку – вычисления нового значения координаты Y надо использовать данные из структуры TEXTMETRIC, которая хранит атрибуты (метрики) текущего шрифта, связанного с КУ. В том числе, поле tmHeight задает полную высоту символов используемого шрифта, tmExternalLeading определяет расстояние (интервал) между строками. Для доступа к структуре использовать метод MFC CDC::GetTextMetrics( ) либо функцию API

BOOL GetTextMetrics(HDC hdc, LPTEXTMETRIC lptm),

где lptm – указатель на результат – структуру TEXTMETRIC.

ЛАБОРАТОРНАЯ РАБОТА
"Создание интерфейсов. Диалоговые окна. Меню. ЭУ. Каркас Hello"

ЦЕЛЬ РАБОТЫ. 1. Изучить использование типовых ЭУ (кнопок, окон редактирования, списков). 2. Изучить управление меню, технологию создания интерфейсов с использованием меню и диалоговых окон. 3. Изучить типовые диалоговые окна. 4. Изучить создание приложений на базе каркаса Hello.

СОСТАВ ОТЧЕТА. 1. По заданиям п. 2, 7 – диаграммы состояний. 2. По заданию п. 8: формы интерфейса; диаграмма состояний; описание обработчиков; диаграмма компонентов; листинг приложения. 3. По заданию п. 9: диаграмма состояний. 4. По заданию п. 11: диаграмма состояний; описание обработчиков; листинг приложения.

ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ. Работа выполняется по пособиям [2, 3].

1*. Выполнить контрольные задания.

Окна. Диалоговые окна

2. Разработать приложение с диалоговым окном в качестве главного окна (на базе ТКП) с единственной кнопкой ОК и без системных кнопок. Его функция – создать окно с

клиентской областью и передать ему управление. При этом диалоговое окно должно остаться на экране. Окно типа главное воспринимает сообщение нажатия левой кнопки мыши (действие – вывод информационного сообщения о событии; закрытие приложения). Приложение также может быть закрыто и системной кнопкой.

Модификации: а) блокировка кнопки ОК ДО при запуске ГО; б) попеременное нажатие кнопки ОК должно приводить к визуализации-девизуализации ГО.

ПОЯСНЕНИЯ. Конкретный экземпляр окна с рамкой создается функцией hWnd = CreateWindow(...), которая возвращает его дескриптор. Само окно после визуализации активно, но если его закрыть, то конкретный экземпляр окна теряется как и его дескриптор. Поэтому для повторной визуализации окна того же стиля необходимо заново создать его экземпляр функцией CreateWindow и получить новый дескриптор. Соответственно в WinMain( ) следует создать стиль второго окна, создать его экземпляр, создать и активизировать диалоговое окно.

Обрабатываемое сообщение диалогового окна – WM_COMMAND: IDOK, посылаемое по нажатии кнопки ОК. Действие – визуализация окна с рамкой, созданного в главной функции. Соответственно в обработчике диалогового окна для этого сообщения надо визуализировать второе окно – команды ShowWindow(*ДескрипторОкнаСРамкой, ПараметрВизуализации*) и UpdateWindow (*ДескрипторОкнаСРамкой*), а секцию закрытия следует не использовать и блокировать обработку повторного нажатия кнопки ОК.

Обрабатываемые сообщения второго окна: 1. Сообщение WM_RBUTTONDOWN. Действие – вывод информационного сообщения о событии. 2. Сообщение WM_DESTROY по закрытию окна. Действие – посылка сообщения на закрытие приложения PostQuitMessage(0). Соответственно в обработчике второго окна надо реагировать на эти события.

Для этого: 2.1. Разработать: диаграмму состояний; описать обработчики. 2.2. Реализовать приложение.

3. Разработать приложение с главным окном. Его функция – запуск при нажатии левой кнопки мыши диалогового окна, содержащего кнопку завершения работы приложения. При запуске ДО разрушать ГО без завершения работы приложения. Собственное разрушение ГО должно быть заблокировано.

Меню

4. Изучить теоретический материал по созданию и использованию пользовательского меню в составе главного окна (см. § 2.2 [2]).

5. Воспроизвести демонстрационный пример. Разработать диаграмму состояний.

ЗАДАНИЕ (демонстрационный пример). *Разработать приложение на базе ТКП с окном с рамкой в качестве главного, содержащее простейшее пользовательское меню.*

Примерный вид интерфейса показан ниже. Здесь два пункта типа POPUP – ВВОД, ВЫВОД, четыре пункта типа MENUITEM – Строка 1, Строка 2 (подпункты пункта ВВОД), Строка 1, Строка 2 (подпункты пункта ВЫВОД).

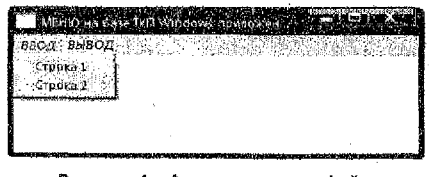

Рисунок 4 – Фрагмент интерфейса

6. Внести изменения в меню (подпункт "Строка2" пункта "ВВОД" переименовать в "Выход" и генерировать сообщение для завершения работы приложения; по выбору каждого конечного пункта меню выводить комментирующее сообщение; подпункт "Строка1" пункта "ВЫВОД" сделать типа POPUP и далее добавить список подпунктов "Дополнительный1", "Дополнительный2").

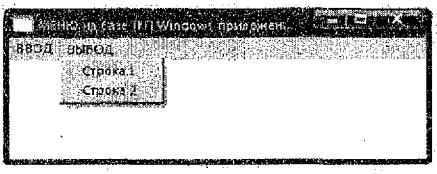

Рисунок 5 – Фрагмент интерфейса

Элемент управления список

7. Использование диалогового окна со списком и окошком редактирования в качестве главного (см. § 2.1.4 [2]).

ЗАДАНИЕ (демонстрационный пример). *Разработать приложение с диалоговым окном в роли главного. Цель приложения: управление списком строк – фамилий (просмотр, выбор, добавление, удаление, редактирование, расположение в алфавитном порядке). При запуске приложения в качестве главного окна выводится диалоговое, содержащее пустой список, кнопки управления списком и кнопку Завершить (Cancel).* Примерный вид окна представлен на рисунке ниже. Состав ресурсов: список; окно редактирования; кнопки; рамки – ЭУ типа Group Box и т.д.

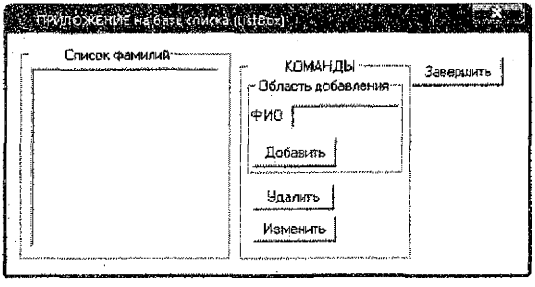

Рисунок 6 – Интерфейс приложения

Для этого: 7.1. Изучить теоретический материал (§ 2.1.4 [2]) и разработать диаграмму состояний. 7.2. Воспроизвести демонстрационный пример.

8. Модифицировать приложение для работы со списком строк (п. 7), обеспечив константную инициализацию списка заранее заданными строками (фамилиями). Для этого: 8.1. Разработать: формы; диаграмму состояний; обработчики; диаграмму компонентов. 8.2. *Реализовать приложение.*

Меню и диалоговые окна

9. Изучить теоретический материал по совместному использованию диалоговых окон и меню и воспроизвести пример (см. § 2.3 [2]).

ЗАДАНИЕ (демонстрационный пример). *Разработать приложение на базе ТКП для многократного ввода-вывода строк и фиксации числа введенных строк. Создать интерфейс на основе окна с рамкой и меню. Через меню вызывается диалоговое окно для ввода и окно сообщения для вывода строки. В клиентскую область главного окна выводится число введенных строк. Вид интерфейса показан ниже. Главное окно содержит меню с двумя пунктами. По пункту ВВОД выводится диалоговое окно ввода строки. По пункту ВЫВОД выводится строка и количество строк.*

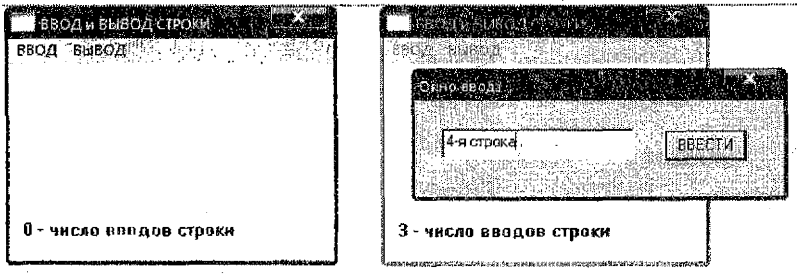

Рисунок 7 – Фрагмент интерфейса

Типовые диалоговые окна

10. Изучить теоретический материал по использованию типовых диалоговых окон (см. § 2.4 [2]) и воспроизвести пример.

10.1. ЗАДАНИЕ (демонстрационный пример). *Создать приложение с окном с рамкой. По сообщению WM_ LBUTTONDOWN выводить окно типа "Сохранить как. По сообщению WM_RBUTTONDOWN выводить окно типа "Открыть".*

10.2. ЗАДАНИЕ (демонстрационный пример). *Модифицировать приложение. Создать приложение с окном. По сообщению WM_RBUTTONDOWN визуализировать диалоговое окно типа "Открыть" с последующим открытием выбранного файла, по сообщению WM_LBUTTONDOWN визуализировать диалоговое окно типа "Сохранить как" с последующим сохранением информации в выбранном файле.*

Работа с каркасом типа Hello

11. Создать проект (см. § 4.3.6 [3]) на базе шаблона Hello. Модифицировать меню приложения. По выбору подпункта Enter активизировать диалоговое окно для ввода числа. Введенное вещественное число возводить в квадрат, а полученный результат выводить с помощью окна сообщений. Фрагменты интерфейса приведены на рисунке 8.

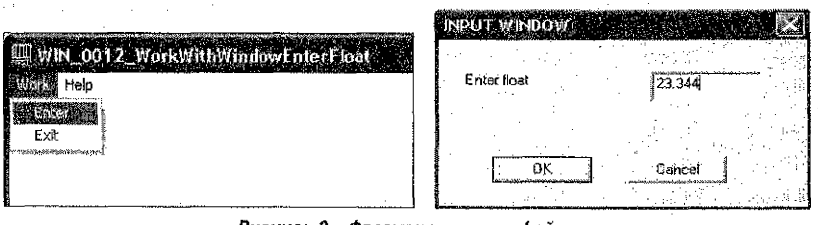

Рисунок 8 – Фрагменты интерфейса

КОНТРОЛЬНЫЕ ЗАДАНИЯ

1. Разработать приложение с интерфейсом из двух окон – главного окна и диалогового, запускаемого щелчком правой клавиши мыши. Состав диалогового окна: ЭУ список и другие ЭУ для его управления в соответствии с заданием (последовательность модификаций приложения в п.п. 1.1-1.4). Прототипировать каждый вариант приложения, описать: интерфейсные формы, диаграммы классов (при использовании классов), диаграммы состояний интерфейсных форм.

1.1. Обеспечить константную инициализацию списка (3-4 записи) и выполнение команды Удалить для выделенной записи списка. 1.2. Обеспечить выполнение команды Добавить для внесения в список новой записи. 1.3. Обеспечить вывод текущего содержимого списка в клиентскую область главного окна при завершении работы с диалоговым. Для работы с данными (списком чисел) описать пользовательский класс (классы). 1.4. Обеспечить сохранение и инициализацию списка из "системного" файла.

2. Разработать приложение с диалоговым окном в качестве главного окна и списком для работы с номерами телефонов. Примерный вид интерфейса представлен ниже.

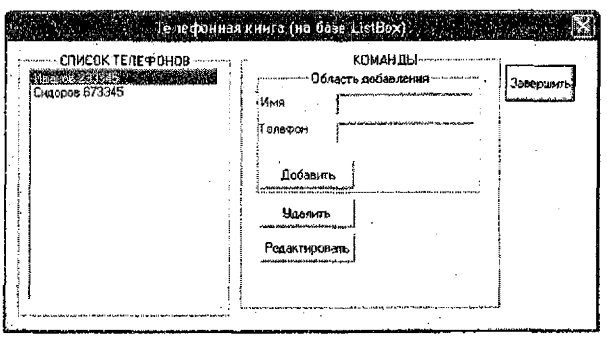

Рисунок 9 – Интерфейс приложения

3. Модифицировать приложение (п. 9 текущей работы). 3.1. Оформить все секции обработчика в виде системы функций. 3.2. Модифицировать приложение, реализуя вывод-вывод массива строк. Вывод строк производить либо в клиентской области окна, либо в ЭУ список диалогового окна. 3.3. Модифицировать приложение, сохраняя и загружая массив строк из заданного файла.

4. Модифицировать приложение (п. 10 текущей работы). 4.1. Реализовать запись в заданный файл и считывания из заданного файла константного массива строк (например, строк вида "Запись1", "Запись2" и т.д.). 4.2. Модифицировать приложение для записи в заданный файл и считывания из заданного файла произвольного массива строк. 4.3. Модифицировать предыдущее приложение, добавив управление приложением через меню.

5. Разработать приложение с использованием меню и диалоговых окон. 5.1. Для ввода вещественных данных и вывода результатов их обработки. 5.2. Для ввода массива вещественных данных и вывода результатов их обработки.

6. Разработать приложение для табулирования заданных функций (параметры табулирования вводит пользователь; результаты выводятся как в табличном так и графическом виде в одном или разных окнах).

ЛАБОРАТОРНАЯ РАБОТА "Проектирование приложений. Диаграммы UML"

ЦЕЛЬ РАБОТЫ. 1. Изучить содержание типовых этапов объектно-ориентированной разработки программ. 2. Изучить использование диаграмм UML при разработке программ. 3. Выполнить объектно-ориентированный анализ с элементами проектирования и реализации приложения с документированием результатов на языке UML.

СОСТАВ ОТЧЕТА.

1. Описание варианта задания. Произвольное описание предметной области.

Описание проекта:

2. Прецеденты. 2.1. Список прецедентов и диаграмма. 2.2. Текстовое описание основных и альтернативных потоков событий прецедентов. 2.3. Уточненная диаграмма, перечень сценариев.

3. Прототипирование. 3.1. Список форм. 3.2. Описание (дизайн) форм. 3.3. Диаграммы состояний приложения, прецедентов, форм.

4. Классы. 4.1. Список, общее описание классов. 4.2. Диаграмма классов предметной области на уровне отношений.

5. Первичное описание структуры приложения и обработчиков.

6. Диаграммы последовательностей и видов деятельности для прецедентов.

7. Уточненное описание классов и диаграммы классов предметной области.

8. Диаграммы видов деятельности для прецедентов и методов.

9. Диаграмма классов (с учетом вида каркаса, используемых библиотек).

10. Диаграмма пакетов (состав системы, подсистемы), диаграмм компонентов

11. Диаграмма развертывания.

12. Приложение, инструкция использования.

ИСХОДНЫЕ ДАННЫЕ (предметная область). Список записей заданного типа с возможностью осуществления операций как с самим списком так и с составляющими его записями. Варианты организации списка, типы записей, а также варианты работы с ними приведены ниже. Для работы со списком может потребоваться регистрация и авторизация пользователей. Информация (сам список, логины) может храниться в "системном" или произвольных файлах.

1. Тип ЗАПИСЕЙ: а – строка (текст); б – фамилия, телефон; в – название фирмы, стоимость акции; г – год, расходы, доходы; д – предмет, аудитория, время занятия; е – группа, староста, численность студентов.

2. Тип ХРАНЕНИЯ СПИСКА записей: а – как внутренние данные; б – как "системный" файл; в – как произвольный файл.

3. Тип СПИСКА: а – произвольный; б – упорядоченный (отсортированный) по заданному полю (полям) записей.

4. Тип КАРКАСА приложения (интерфейса): а – на базе окна типа главное с управлением задачами (прецедентами) с помощью меню; б – на базе диалогового окна с управлением задачами (прецедентами) с помощью системы кнопок.

5. Тип организации АВТОРИЗАЦИИ пользователей: а – без регистрации и авторизации пользователей; б – с авторизацией пользователей и хранением постоянного перечня логинов во внутреннем списке приложения; в – с регистрацией и авторизацией пользователей и хранением перечня логинов во внутреннем списке приложения; г – с авторизацией пользователей и хранением постоянного перечня логинов в "системном" фай-

ле приложения; д – с регистрацией и авторизацией пользователей и хранением перечня логинов в "системном" файле приложения.

6. Типы ОПЕРАЦИЙ со СПИСКОМ: а – создание нового списка с "автоматическим" заполнением начальным константным содержимым; б – создание нового пустого списка; в – создание нового списка с "ручным" заполнением начальным содержимым; г – вывод содержимого списка в виде перечня записей; д – просмотр содержимого списка "запись за записью"; е – поиск первой записи списка (по заданному номеру, по шаблону); ж – поиск всех записей списка (по заданном шаблону); з – очистка содержимого списка; и – удаление списка; к – создание копии списка.

7. Типы ОПЕРАЦИЙ с ЗАПИСЬЮ: а – добавление новой записи (в конец списка, в начало списка, в соответствии с типом списка); б – редактирование выбранной записи (изменение полей с последующим сохранением или отменой изменений); в – удаление записи (с конца списка, с начала списка, указанной (-ых) записей, в том числе найденной (-ых) записей).

Таблица 1 – Варианты заданий

№	Атрибуты варианта						
	1	2	3	4*	5	6	7
1	б	а (без файла)	б	а	д (файл)	а, г	б
2	в	а (без файла)	а	б	г (файл)	в, г	в
3	г	в (произвольный файл)	б	а	в (без файла)	б, г, е	а
4	д	в (произвольный файл)	а	б	б (без файла)	а, г	б, в
5	е	в (произвольный файл)	б	а	а (без авторизации)	в, д, г, е	в
6	а	в (произвольный файл)	а	б	б	в, г	а, в
7	а	в (произвольный файл)	б	а	д	а, з	а, б
8	б	б (системный файл)	а	б	б	а, ж	а, в
9	г	б (системный файл)	б	а	в	а, г	б, в
10	д	б (системный файл)	а	б	-	б, г, е	а, б, в

ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ. Выполняется с использованием [4]. Перечень этапов работы представлен в таблице 1. Состав работ описан ниже. В текущей работе выполняется этап 1, остальные этапы дорабатываются самостоятельно с предъявлением и обсуждением результатов на последующих работах. Выполнение работы завершается представлением единого отчета и демонстрацией приложения.

Таблица 2 – Перечень этапов

Выполняемые работы
ЭТАП 1 – Объектно-ориентированный анализ предметной области
1. Описание предметной области, автоматизируемых задач (результаты интервью).
2. Выявление и описание прецедентов (построение диаграммы и описание потоков).
3. Прототипирование приложения: - первоначальное выявление интерфейсных форм (сопутствующих классов); - дизайн форм; - описание диаграммы состояний приложения; - описание диаграмм состояний для прецедентов; - описание диаграмм состояний для отдельных форм.
4. Выявление, первоначальное описание классов, классов предметной области, диаграммы классов предметной области на уровне "первичных" отношений.
5. Первичное описание структуры приложения и обработчиков.

Продолжение таблицы 2

ЭТАП 2 – Объектно-ориентированное проектирование
6. Описание диаграмм последовательностей, видов деятельности прецедентов.
7. Уточненное описание структуры классов, диаграммы классов предметной области.
8. Описание диаграмм видов деятельности методов.
Реализация интерфейса приложения (макетирование).
ЭТАП 3 – Объектно-ориентированное проектирование и реализация
9. Уточненное описание диаграммы классов с учетом конкретизаций каркаса приложения, используемых библиотек, введения дополнительных классов.
10. Описание структуры приложения – диаграммы пакетов, компонентов.
11. Описание диаграммы развертывания приложения.
12. Реализация приложения, тестирование, документирование.

Этап 1. Объектно-ориентированный анализ

ИСХОДНЫЕ ДАННЫЕ. Заданная предметная область – обработка списка записей указанного типа. Выполнение этапа включает:

1. Описание предметной области и автоматизируемых задач (интервью). Выполняется в произвольной текстовой форме с использованием графических, табличных, формульных и других описаний. Результаты должны максимально полно, достоверно, детально описывать объект автоматизации и автоматизируемые задачи.

2. Выявление и описание прецедентов – вариантов использования системы (задач, функций, сценариев) в виде диаграммы прецедентов с текстовым описанием основных и альтернативных потоков событий каждого из прецедентов. Выполняется путем: анализа текста интервью, документов предметной области и выявления назначения системы; уточнения пользовательских функций системы, уточнения перечня задач, решаемых системой для пользователя; описания отношений прецедентов.

Соответственно описание прецедентов (решаемых с помощью системы пользовательских задач) включает: первоначальное описание диаграммы прецедентов; уточненное описание диаграммы прецедентов с учетом отношений включения, расширения, обобщения.

Для каждой задачи (прецедента) уточняется последовательность использования компонентов системы, в т.ч. выявляются исполнители (инициаторы и потребители), пред- и пост-условия, причины запуска, основные этапы работы системы при реализации задачи (прецедента). Выявляются события и формируются текстовые описания потоков событий. Соответственно для каждого прецедента выполняется: текстовое описание основного и альтернативных потоков событий.

3. Прототипирование приложения включает: первоначальное выявление интерфейсных форм (сопутствующих классов). Для этого надо проанализировать текстовое описание каждого прецедента, выбрать из текста существительные – претенденты на роль интерфейсных форм и сформировать их список; дизайн форм (определение состава, вида формы); описание диаграммы состояний приложения, прецедентов, отдельных форм.

4. Идентификация (выявление) и первоначальное описание классов, классов предметной области по результатам интервью и описаниям прецедентов (потоков событий). Построение диаграмм. Для этого следует:

4.1. Выявить классы, участвующие в потоке событий каждого прецедента. Для этого надо проанализировать текстовое описание каждого прецедента и выбрать из текста существительные – претенденты на роль классов или атрибутов классов.

4.2. Сформировать общий список классов. Для этого надо сравнить классы, выявленные в каждом из прецедентов, и внести в список в единственном экземпляре.

4.3. Выполнить описание классов. При этом: первоначальное описание следует вести на русском языке; для названий классов, атрибутов, методов следует подбирать наиболее близкие по смыслу, лаконичные названия также на русском языке. В том числе, для каждого класса (объекта) делается описание: смыслового названия; обязанностей (назначения); примерного состава основных свойств и методов. Для каждого члена-свойства указываются: название, тип, смысловое назначение. Для каждого члена-метода указываются: название, прототип, смысловое назначение.

4.4. Первоначальное описание отношений сущностей типа класс (объект) в виде диаграмм классов (объектов) для классов предметной области на уровне отношений с указанием атрибутов – роли, направления, кратности отношений.

5. Описание структуры (модульной структуры) приложения, событий запуска, прототипов, состава обработчиков.

Этап 2. Объектно-ориентированное проектирование

ИСХОДНЫЕ ДАННЫЕ. *Результаты выполнения этапа "Анализ предметной области".* Порядок выполнения:

6. Описание порядка взаимодействия объектов во времени в терминах передачи сообщений и реализации ими видов деятельности, действий в виде диаграмм последовательностей и видов деятельности. Выполняется для каждого прецедента, например, анализом их текстовых описаний, диаграмм классов и объектов.

7. Уточненное описание структуры классов и диаграммы классов предметной области с учетом типов отношений. Выполняется путем анализа ранее полученной диаграммы классов и соотнесения ранее выделенных отношений с ассоциациями типа агрегация, композиция, обобщение (с указанием родителей, потомков, одиночного, множественного, прямого, косвенного типов наследования) или с зависимостями (с описанием характера зависимости).

Уточненное описание структуры классов и диаграммы классов предметной области путем анализа ранее полученной диаграммы на целесообразность введения, выделения новых классов: абстрактных классов; параметризованных классов.

8. Описание алгоритмов методов классов в терминах протекающих в них процессов, потоков, передач управления в виде диаграмм видов деятельности.

Реализация интерфейса приложения (макетирование).

Этап 3. Объектно-ориентированная реализация

ИСХОДНЫЕ ДАННЫЕ. *Результаты выполнения предыдущих этапов.* Порядок выполнения этапа:

9. Уточненное описание диаграммы классов с учетом конкретизаций каркаса приложения, используемых библиотек (STL), введения дополнительных классов.

10. Описание структуры приложения в виде диаграммы компонентов: на уровне файлов исходного текста (структура проекта приложения); на уровне компонентов – испол-

нимых файлов и компонентов информационной базы приложения – используемых файлов, баз данных, документов и т.п.

11. Описание диаграммы развертывания приложения

12. Реализация приложения в системе MS Visual Studio C++. Тестирование, документирование приложения.

ЛАБОРАТОРНАЯ РАБОТА
"Типовой каркас приложения MFC (ТКП). Разработка приложений на базе ТКП MFC"

ЦЕЛЬ РАБОТЫ. 1. Изучение типового каркаса MFC-приложения (ТКП). 2. Изучение организации обработки сообщений. 3. Организация вывода в клиентскую область главного окна, поддержка перерисовки. 4. Реализация приложений на базе ТКП.

СОДЕРЖАНИЕ ОТЧЕТА. 1. Описание ТКП (интерфейс, диаграммы классов и взаимодействия объектов). 2. Для приложений п.п. 7, 8 – диаграммы классов и компоновки.

ПОРЯДОК ВЫПОЛНЕНИЯ. Работа выполняется по методическому пособию [5].

1. Изучить теоретический материал: ознакомиться с общими сведениями об ОС, технологиях программирования, библиотеке MFC (см. §§ 1.1-1.5); изучить основные типы сообщений, прототипы соответствующих обработчиков (см. §§ 1.6-1.7); ознакомиться с методами управления контекстом устройства, перерисовкой (см. §§ 1.8-1.9); ознакомиться со средой разработки (см. § 1.10); ознакомиться с составом классов ТКП (см. § 2.1); изучить технологию обработки сообщений (см. § 2.2); изучить архитектуру ТКП, используемые классы и методы (см. §§ 2.3-2.4).

2. Создать "пустое" приложение (ТКП) в соответствии с § 1.11 и § 2.5. Это каркас приложения без обработки сообщений. Запустить. Изучить его свойства. Разработать диаграммы UML.

3. Модифицировать приложение, включив чувствительность к сообщению WM_LBUTTONDOWN (см. §§ 1.6-1.7). Для этого в классе CMainWin включить (разкомментировать) прототип функции-обработчика

afx_msg void OnLButtonDown(UINT flags, CPoint loc);

- в карте сообщений BEGIN_MESSAGE_MAP() включить (разкомментировать) чувствительность приложения к нажатию клавиши "мыши"

ON_WM_LBUTTONDOWN();

- описать (разкомментировать) соответствующую функцию-обработчик; выполнить приложение, исправить ошибки; добавить в тело обработчика вывод сообщения о его запуске с помощью метода MessageBox (см. § 3.2); добавить в тело обработчика вывод текста "Работает ТКП-MFC" в главном окне с помощью метода TextOut (см. § 3.2). Для этого получить контекст устройства – создать объект класса CClientDC (см. § 1.8); запустить приложение, выполнить манипуляции с окном (перемещение, изменение размеров, перекрывание другим приложением) и проанализировать эффект перерисовки.

4. Создать приложение (на базе ТКП), аналогичное созданному в предыдущем пункте, но включив чувствительность к сообщению WM_PAINT (см. §§ 1.6-1.7). Для этого в класс CMainWin включить прототип функции-обработчика

afx_msg void OnPaint();

- в карте сообщений BEGIN_MESSAGE_MAP() включить чувствительность приложения к сообщению перерисовки

ON_WM_PAINT();

- описать соответствующую функцию-обработчик; выполнить приложение, исправить ошибки; добавить в тело обработчика вывод сообщения о его запуске с помощью метода MessageBox (см. § 3.2); добавить в тело обработчика вывод текста "Работает ТКП-MFC" в главном окне с помощью метода TextOut (см. § 3.2). Для этого получить контекст устройства – создать объект класса CPaintDC (см. § 1.8); запустить приложение, выполнить манипуляции с окном (перемещение, изменение размеров, перекрывание другим приложением) и анализировать эффект перерисовки; добавить табуляцию любой функции с выводом результатов в табличном виде (аргумент-функция).

5. Создать приложение для вывода координат курсора "мыши", фиксируемых по щелчку ее левой клавиши. Для этого: создать ТКП и включить чувствительность к сообщению WM_LBUTTONDOWN; обеспечить вывод координат курсора "мыши", фиксируемых по щелчку ее левой клавиши, в обработчике сообщения WM_LBUTTONDOWN (см. § 1.8); запустить приложение, выполнить манипуляции с окном и анализировать эффект перерисовки.

6. Модифицировать приложение, решив проблему перерисовки – ввод координат производить по сообщению WM_LBUTTONDOWN, вывод по WM_PAINT. Для этого: включить чувствительность к сообщению WM_PAINT; модифицировать обработчик сообщения WM_LBUTTONDOWN, осуществляя в нем получение и запоминание координат курсора момента нажатия левой клавиши "мыши"; обеспечить вывод координат курсора в обработчике сообщения WM_PAINT. Запустить приложение, выполнить манипуляции с окном и анализировать эффект перерисовки; для вызова перерисовки окна после каждого щелчка левой клавиши "мыши" в обработчике сообщений "мыши" использовать метод InvalidateRect (см. § 1.9); запустить приложение, выполнить манипуляции с окном и анализировать эффект перерисовки.

7. Модифицировать приложение для вывода координат в той точке экрана, где был зафиксирован щелчок левой клавиши "мыши".

8. Создать приложение для ввода символа с клавиатуры (сообщение WM_CHAR) и эхо-вывода либо в левый верхний угол главного окна, либо в позицию, задаваемую нажатием левой клавиши мыши. Разработать диаграммы UML.

Примечание. Ввод и запоминание символа выполнять в обработчике сообщения WM_CHAR, ввод и запоминание координат выполнять в обработчике сообщения WM_LBUTTONDOWN, вывод выполнять в обработчике сообщения WM_PAINT. Вызывать перерисовку при вводе нового символа или при изменении координат вывода методом InvalidateRect(NULL).

9. Создать "пустое" приложение (ТКП) с разными вариантами стиля главного окна (см. ПРИЛОЖЕНИЕ 1, § 1.5 [5]). В том числе, создать главное окно: без возможности скроллинга в клиентской области; со скроллингом в клиентской области окна; окно без системных кнопок; окно заданного размера и местоположения (например, с координатами левого верхнего угла 10, 10 и правого нижнего – 200, 200) без системных кнопок (для

этого использовать структуру типа RECT, а затем класс CRect.. Пример задания параметров клиентской области приведен ниже)

```
RECT TheRect = {10, 10, 200, 200};
```

Или

```
RECT TheRect;
TheRect..top = 10;
TheRect..left = 10;
TheRect..bottom = TheRect.right = 200;
```

- главное окно с системным меню; окно без возможности изменения размеров; окно с возможностью изменения размеров и с начальным выводом на весь экран (см. параметры метода ShowWindow).

ЛАБОРАТОРНАЯ РАБОТА
"Разработка приложений на базе ТКП MFC. Использование STL."

ЦЕЛЬ РАБОТЫ. 1. Изучение элементов автоматизации разработки MFC-приложения. 2. Реализация MFC-приложений на базе ТКП. 3. Изучение средств библиотеки STL.

СОДЕРЖАНИЕ ОТЧЕТА.. 1. Описание ТКП, построенного "мастером" (интерфейс, состав файлов, классов, ресурсов, диаграммы классов, диаграммы взаимодействия объектов). 2. Описание приложений.

ПОРЯДОК ВЫПОЛНЕНИЯ. Работа выполняется по методическому пособию [5].

Создание приложений MFC

1. Элементы автоматизации

1.1. Создание каркаса MFC-приложения (ТКП) с помощью мастера. Для этого запустите мастер MFC (например, AppWizard (exe)). В поле Project name введите название проекта приложения и нажмите кнопку OK. В окне мастера выберите – однодокументное приложение (Single Document Interface – SDI, флажок – Single document) и откажитесь от поддержки архитектуры Документ-Вид (сбросьте флажок – Document/View architecture support). Остальные опции сохраните со значениями по умолчанию.

В версиях студии с поддержкой NET выберите – приложение MFC с одним документом и без поддержки архитектуры Document/View.

Запустите приложение. Изучите его состав и функциональные возможности.

1.2. Добавление обработчиков. Во всех каркасах MFC поддерживаются элементы автоматизации. Так включение макросов в карты сообщений и добавление соответствующих обработчиков сообщений может быть выполнено, например, мастером ClassWizard. Пусть надо добавить обработчик события – нажатие левой клавиши мыши. Для этого надо: открыть ClassWizard, нажав Ctrl-W; перейти на вкладку Message Maps; выбрать в списке Class Name – класс соответствующего окна; выбрать в списке Object IDs аналогичный пункт; в списке Messages выбрать нужное сообщение (здесь WM_LBUTTONDOWN); указать Add Function, а затем Edit Code и добавить необходимые действия в обработчик.

В версиях студии с поддержкой NET – в окне Классов выберите соответствующий класс (например, класс, поддерживающий окно – клиентскую область – CChildView). Для него в окне Свойств на вкладке Сообщения выберите нужное, добавьте обработчик (здесь выберите сообщение WM_LBUTTONDOWN и согласитесь с добавкой системного обработчика), внесите в обработчик необходимые действия. Далее используйте эти возможности при разработке приложений.

2. Создать приложение на базе ТКП для рисования плоских изображений отрезками прямых линий. Координаты точек вводить нажатием левой клавиши "мыши". Введенные координаты сохранять в динамическом массиве (например, использовать vector). Выводить изображение, путем соединения прямыми линиями соседних точек, по нажатию правой клавиши "мыши".

ПРИМЕЧАНИЕ. В начальном варианте реализовать только ввод, хранение и вывод координат. Для этого создать ТКП и обработчики сообщений. Для рисования точек, линий, геометрических фигур используются методы класса CDC, поэтому необходимо создать объект этого класса. Для хранения координаты точки можно использовать структуру POINT, класс CPoint. Для перемещения пера (без рисования линии) используется метод CDC::MoveTo(int x, int y) или CDC::MoveTo(POINT point). Для рисования прямой линии используется метод CDC::LineTo(int x, int y) или CDC::LineTo(POINT point).

3*. Модифицировать приложение (п. 2), использовав пользовательский класс ЛИНИИ для работы с координатами точек, линиями изображения.

4. Создать приложение на базе ТКП, которое выводит строки, например, как показано на рисунке ниже.

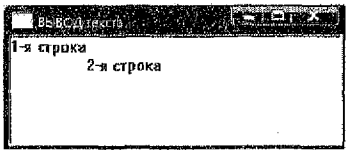

Рисунок 10 – Фрагмент интерфейса

ПРИМЕЧАНИЕ. При выводе данных в клиентской области экрана требуется вычислять координаты начальной позиции для вывода первого символа каждой строки, а при переходе на новую строку окна вычислять новое значение координаты y. Для определения новой координаты X, после вывода текущей строки, следует использовать метод CDC::GetTextExtent() вычисления ее реальной длины в логических единицах. Для перехода на новую строку окна и вычисления нового значения координаты Y надо использовать данные из структуры TEXTMETRIC, которая хранит атрибуты (метрики) текущего шрифта, связанного с КУ. В том числе, поле tmHeight задает полную высоту символов используемого шрифта, tmExternalLeading определяет расстояние (интервал) между строками. Для доступа к структуре использовать метод CDC::GetTextMetrics( ).

5*. Создать приложение, имитирующее работу упрощенного текстового редактора.

Использование библиотеки STL

6. Разработать консольное приложение для обработки набора чисел с использованием vector: создать пустой вектор нулевой длины; вывести атрибуты его исходного со-

стояния; заполнить вектор значениями с клавиатуры; вывести содержимое и состояние вектора (использовав два варианта организации цикла); добавить/удалить заданные значения; вывести содержимое и состояние вектора. 6.1.* Модифицировать приложение – действия выполнять через консольное меню. 6.2.* Реализовать приложение как оконное, выбор действий выполнять через оконное меню.

7. Разработать консольное приложение для работы с набором пользовательских объектов (например, КНИГА) с использованием vector. Состав действий: добавление, удаление, просмотр, сравнение объектов. Для сравнения объектов перегрузить оператор отношения двумя способами. 7.1.* Реализовать приложение как оконное. 7.2.* Модифицировать приложение, обеспечив хранение набора объектов в файле.

8. Разработать консольное приложение для обработки набора символов с использованием list: создать пустой список нулевой длины; вывести атрибуты его исходного состояния; заполнить значениями с клавиатуры; вывести содержимое и состояние (использовав два варианта организации цикла); добавить/удалить заданные значения; вывести содержимое и состояние список. 6.1.* Модифицировать приложение – действия выполнять через консольное меню. 6.2.* Реализовать приложение как оконное, выбор действий выполнять через оконное меню.

9. Разработать консольное приложение для работы с набором пользовательских объектов (например, КНИГА, отсортированных по названиям) с использованием list. Состав действий: добавление, удаление, просмотр. Для сортировки использовать алгоритм. 9.1.* Реализовать приложение как оконное. 9.2.* Модифицировать приложение, обеспечив хранение набора объектов в файле.

ЛАБОРАТОРНАЯ РАБОТА
"Разработка консольных приложений в C++/CLI. Использование STL"

ЦЕЛЬ РАБОТЫ. 1. Изучение базовых средств C++/CLI для работы в консольном режиме. 2. Разработка консольных приложений с использованием управляемых данных, кодов. 3. Изучение базовых средств библиотеки STL.

СОДЕРЖАНИЕ ОТЧЕТА. Описание реализованных приложений.

ПОРЯДОК ВЫПОЛНЕНИЯ. Выполнение работы ведется по методическим указаниям [6]. Теоретический материал по STL [7].

Создание приложений CLI

1. Изучите особенности системы типов языка, базовые типы и их совместимость с типами Visual C++, особенности работы с "управляемыми" данными (дескрипторами, ссылками CLI), процессы паковки и распаковки данных. Ознакомьтесь с каркасом консольного приложения, вводом-выводом данных (методы WriteLine, ReadLine, Read, ReadKey), форматами вывода. Изучите особенности организации вычислений, применения команды for each. 1.1. Воспроизведите описания переменных [6] и выполните преобразования типов. 1.2. Организуйте ввод-вывод данных – результатов вычисления заданной функции. 1.3*. Организуйте вывод результатов вычисления заданной функции в виде таблицы. 1.4*. Воспроизведите примеры из [6], определите их назначение.

2. Изучите особенности описания, инициализации, доступа, типовые методы и свойства, особенности применения массивов C++/CLI (array<...>^), особенности работы с многомерными массивами, массивами ссылок. 2.1*. Воспроизведите примеры [6], определите их назначение. 2.2. Создайте приложение для ввода и суммирования элементов одно- и двумерного массива.

3. Изучите особенности описания, инициализации, доступа, типовые методы и свойства строковых данных C++/CLI (String^), особенности перевода данных в строковый тип и обратно. 3.1*. Воспроизведите примеры [6], определите их назначение.

4. Изучите особенности описания и применения функций. Создайте приложение с функцией: 4.1. Для поиска максимального значения в одномерном массиве. 4.2. Для поиска максимального значения в двумерном массиве. 4.3*. Для сортировки числовых значений в одномерном массиве. 4.4*. Для удаления из строки пробелов (использовать метод String::Replace(L "ЗаменяемыйСимвол", L "ЗамещающийСимвол").

5. Изучите особенности описания и применения пользовательских классов типов значение и ссылка, особенности использования свойств, правила перегрузки операторов. 5.1*. Воспроизведите примеры [6], определите их назначение. 5.2. Создайте приложение для работы с ref-классом для хранения и обработки целого значения. Предусмотрите все типы конструкторов. 5.3. Добавьте свойство. 5.4. Перегрузите операторы "+" (для разных вариантов сочетаний типов операндов), "++", "=". 5.5*. Создайте приложение для работы с ref-классом для хранения и обработки списка строк.

6. Изучите особенности наследования ref-классов. 6.1*. Воспроизведите примеры [6], определите их назначение. 6.2. Создайте путем наследования приложение для работы с классом для хранения и обработки двух целых значений.

7. Изучите особенности описания и использования интерфейсов, а также описания классов на базе интерфейсов. 7.1*. Воспроизведите примеры [6], определите их назначение. 7.2. Создайте приложение для работы с ref-классом для хранения и обработки целых значений через класс-интерфейс (обеспечьте свойство для доступа к значению и метод для вывода хранимого значения). 7.3. Создайте приложение для работы с двумя ref-классами (один для хранения целого, другой – строки) через общий класс-интерфейс (обеспечьте методы установки и вывода хранимых значений).

Использование библиотеки STL

8. Разработать консольное приложение для ведения списка адресных данных (фамилия – адрес) на базе контейнера map.

9.* Разработать консольное приложение для замены слов из вводимого набора другими словами в соответствии с "кодовой" таблицей (исходное слово – слово-заменитель) на базе контейнера map.

СПИСОК ЛИТЕРАТУРЫ

1. Муравьев, Г.Л. Методическое пособие "Создания windows-приложений в системе Microsoft Visual Studio C++. Процедурный стиль". – Брест: БрГТУ, 2007. – Ч. 1. – 48 с.

2. Муравьев, Г.Л. Методическое пособие "Основы создания windows-приложений в системе Microsoft Visual Studio C++. Процедурный стиль" / Г.Л. Муравьев, В.И. Хвещук, В.Ю. Савицкий. – Брест: БрГТУ, 2008. – Ч. 2. – 48 с.

3. Муравьев, Г.Л. Методическое пособие "Создание windows-приложений в системе Microsoft Visual I Studio C++" (электронное издание). – Брест: БрГТУ, 2008. – 50 с.

4. Кендалл, С. UML. Основные концепции. – М.: Издательский дом Вильямс, 2002.

5. Муравьев, Г.Л. Методическое пособие "Основы создания windows-приложений в системе Microsoft Visual I Studio C++ на базе библиотеки MFC" / Г.Л. Муравьев, В.И. Хвещук, В.Ю. Савицкий, С.В. Мухов. – Брест: БрГТУ, 2008. – Ч. 1. – 48 с.

6. Муравьев, Г.Л. Методические указания "Программирование в среде MS Visual Studio на языке C++/CLI (консольные приложения)" (электронное издание). – Брест: БрГТУ, 2015. – 30 с.

7. Шилдт, Г. Самоучитель C++, 3-е изд. – СПб.: БХВ-Петербург, 2003. – 688 с.

СОДЕРЖАНИЕ

ЛАБОРАТОРНАЯ РАБОТА "Каркасы windows-приложений. Типовой каркас приложения (ТКП)" 3

ЛАБОРАТОРНАЯ РАБОТА "Разработка приложений на базе ТКП. Организация вывода" 6

ЛАБОРАТОРНАЯ РАБОТА "Каркасы. Создание интерфейсов. Диалоговые окна" 8

ЛАБОРАТОРНАЯ РАБОТА "Создание интерфейсов. Диалоговые окна. Меню. ЭУ. Каркас Hello" 11

ЛАБОРАТОРНАЯ РАБОТА "Проектирование приложений. Диаграммы UML" 16

ЛАБОРАТОРНАЯ РАБОТА "Типовой каркас приложения MFC (ТКП). Разработка приложений на базе ТКП MFC" 20

ЛАБОРАТОРНАЯ РАБОТА "Разработка приложений на базе ТКП MFC. Использование STL" 22

ЛАБОРАТОРНАЯ РАБОТА "Разработка консольных приложений в C++/CLI. Использование STL" 24

СПИСОК ЛИТЕРАТУРЫ 266

Учебное издание

Составители:
Муравьев Геннадий Леонидович,
Мухов Сергей Владимирович,
Савицкий Юрий Викторович

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

к лабораторным работам по дисциплине
Проектирование программ в интеллектуальных системах
"Разработка windows-приложений"
для студентов специальности «Искусственный интеллект»

Ответственный за выпуск: ***Муравьев Г.Л.***
Редактор: ***Боровикова Е.А.***
Компьютерный набор: ***Муравьев Г.Л.***
Компьютерная верстка: ***Кармаш Е.Л.***

Подписано к печати 21.09.2015 г. Бумага «Снегурочка». Формат 60x84 1/16.
Гарнитура Arial Narrow. Усл. печ. л. 1,63. Уч. изд. л. 1,75. Заказ № 1024. Тираж 50 экз.
Отпечатано на ризографе учреждения образования
«Брестский государственный технический университет».
224017, г. Брест, ул. Московская, 267.