МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение высшего образования
«МОСКОВСКИЙ АВТОМОБИЛЬНО-ДОРОЖНЫЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ (МАДИ)»
ВОЛЖСКИЙ ФИЛИАЛ МАДИ

УТВЕРЖДАЮ
Зав. кафедрой ИиТТП
Петрова А.В.

МЕТОДИЧЕСКИЕ УКАЗАНИЯ К ЛАБОРАТОРНЫМ РАБОТАМ ПО ДИСЦИПЛИНЕ «ПРОГРАММИРОВАНИЕ»

Направление подготовки
09.03.01 «Информатика и вычислительная техника»

Направленность (профиль)
Автоматизированные системы обработки информации и управления в отраслях транспортно-дорожного комплекса

Квалификация
Бакалавр

Кафедра: ИиТТП

Чебоксары

Лабораторная работа 1. Отработка приемов тестирования и отладки демонстрационного приложения.

Используя лекции, справочную систему, литературу и любые другие источники, изучить приемы отладки в среде Microsoft Visual C++ и выполнить задание.

Задание. В приведенной ниже программе исправить синтаксические ошибки. Отработать приёмы тестирования и отладки для исправления имеющихся в ней семантических ошибок. Продемонстрировать исправленный вариант программы.

Программа должна подсчитывать площадь треугольника по формуле Герона и обнаруживать ввод некорректных исходных данных.

Для проверки правильности работы программы используйте следующие тестовые наборы данных.

Тестовый набор данных	Ожидаемый результат (вывод на консоль)
a = 2; b = -1; c = 3	Неверное значение длины стороны b: длина сторона треугольника должна быть положительной!
a = 1; b = 1; c = 3	Это не треугольник!
a = 3; b = 4; c = 5	Площадь треугольника S = 6
a = 5; b = 5; c = 5	Площадь треугольника S = 7

```
#include <iostream>
using namespace std

int main()
{
	int a, b, c;
	cout<<"Введите длину стороны a треугольника: "<<endl;
	cin>> a;

	if (a < 0 || a = 0)
	{
	  cout<<"Неверное значение стороны a: длина стороны треугольника должна быть положительной!"<<endl;
	  return 0;
	}

	cout<<"Введите длину стороны b треугольника: "<<endl;
	cin>> b;

	cout<<"Введите длину стороны c треугольника: "<<endl;
	cin c;
```

```
	if (c < 0 | c = 0)
	{
	  cout<<"Неверное значение длины стороны c: длина стороны треугольника должна
быть положительной!"<<endl;
	  return 0;
	}

	// Проверка существования треугольника:
	// Сумма длин любых двух сторон треугольника д.б. > длины третьей.
	int s
	if ((a + b <= c) || (a + c > b) || (b + c > a));
	{
	  // По формуле Герона
	  float p = (a + b + c) / 2;
	  // sqrt - извлекает квадратный корень из числа
	  s = sqr(p * (p - a) * (p - b) * (p - c));
	  cout << "Площадь треугольника S = " << S << endl;
	}
	else
	  cout << "Это не треугольник!"<<endl;

	return 0;
}
```

Лабораторная работа 2. Разработка алгоритма и написание линейной программы вычисления результата выражения по заданной формуле.

Задание 1. Составить блок-схему алгоритма линейной программы вычисления результата выражения по своему варианту (задание №1 из сборника заданий «Электронный сборник заданий по дисциплинам ‘Информатика’ и ‘Программирование на языке высокого уровня’» (Л.А. Акатнова, И.А. Евстратова, Е.К.Коншина, Л.И. Муравьёва, О.Г. Скуратовская)) и определить область допустимых значений для всех (входных и промежуточных) параметров.

Задание 2. Написать программу по составленной блок-схеме.

Исходные данные (все неизвестные параметры, требуемые для вычисления результата выражения по заданию) должны загружаться из текстового файла с помощью потока. При запуске программы у пользователя запрашивается имя файла с клавиатуры. Результат открытия файлового потока должен проверяться. При неуспешном открытии потока имя файла должно запрашиваться повторно.

При чтении данных из файла должна быть реализована проверка их корректности. Если хотя бы один числовой параметр задан некорректно (содержит недопустимые символы; содержит допустимые символы, но некоторые из них находятся не на своих местах; число выходит за границы диапазона значений выбранного для него типа данных), то дальнейшее чтение значений и вычисление результата выражения не производится. Вместо этого пользователю выдается информативное сообщение, указывающее, какой параметр прочитан с ошибкой. Дополнительно пользователю можно выводить (в кавычках) всю последовательность символов, задающую некорректное значение.

Пробельные символы-разделители могут находиться в любом месте файла и в любом количестве, при этом они не должны влиять на правильность выявления наличия некорректных значений.

Вычисление результата выражения осуществляется только в том случае, если фактическое количество элементов в файле совпадает с указанным при определении массива.

Вывести на экран необходимо результат выражения, а также промежуточные вычисления всех значений, указанных в задании.

Работа программы должна начинаться с очистки окна консоли. В программе должны присутствовать все необходимые подписи, приглашения к вводу и подсказки для пользователя, выводимые на русском языке. Работа программы должна завершаться по нажатию пользователем любой клавиши.

Лабораторная работа 3. Разработка алгоритма и написание программы обработки массивов разнотипных данных, вводимых с клавиатуры и загружаемых из текстового файла.

Задание 1. Составить блок-схему алгоритма обработки массивов по своему варианту (задание №6 из сборника заданий «Электронный сборник заданий по дисциплинам ‘Информатика’ и ‘Программирование на языке высокого уровня’» (Л.А. Акатнова, И.А. Евстратова, Е.К.Коншина, Л.И. Муравьёва, О.Г. Скуратовская)) и определить область допустимых значений элементов массива.

Задание 2. Написать программу с использованием разработанного алгоритма.

Исходные данные (последовательность чисел) должны загружаться из текстового файла с помощью потока. При запуске программы у пользователя запрашивается имя файла с клавиатуры. Результат открытия файлового потока должен проверяться. При неуспешном открытии потока имя файла должно запрашиваться повторно.

Для хранения данных используется статический массив. Количество элементов в массиве – фиксированное и задаётся произвольно.

Подсчет количества числовых значений в файле должен быть совмещен с проверкой их корректности (как в лабораторной работе №1). Если хотя бы одно числовое значение задано некорректно, то дальнейший подсчет количества значений, загрузка и обработка массива не производятся, а вместо этого пользователю выдается информативное сообщение, указывающее порядковый номер некорректного значения. Дополнительно пользователю можно выводить (в кавычках) всю последовательность символов, задающую некорректное значение. Порядковый номер некорректного значения должен правильно определяться для различных видов возможных ошибок, допускаемых при задании числового значения в файле. При подсчете количества числовых значений в файле программа должна корректно обрабатывать ситуации, связанные с возможным наличием в файле «лишних» пробельных символов-разделителей (пробел, табуляция, переход на новую строку). Они могут находиться в любом месте файла и в любом количестве, при этом они не должны влиять на правильность подсчета количества числовых значений и на правильность выявления наличия некорректных значений.

Загрузка и обработка массива производятся только в том случае, если фактическое количество элементов в файле совпадает с указанным при определении массива.

Вывод результатов на экран – форматированный. Если выводится массив, то в первой строке – порядковые номера элементов, а строго под ними – значения соответствующих элементов; в начале строки с номерами – слово «Номера:», в начале строки с элементами – слово «Элементы:».

Работа программы должна начинаться с очистки окна консоли. В программе должны присутствовать все необходимые подписи, приглашения к вводу и подсказки для пользователя, выводимые на русском языке. Работа программы должна завершаться по нажатию пользователем любой клавиши.

Лабораторная работа 4. Разработка алгоритма и написание программы обработки массивов разнотипных данных с использованием динамической памяти.

Задание. Заменить в программе из лабораторной работы №2 обработку статических массивов на обработку массивов с использованием динамической памяти.

Исходная последовательность чисел должна загружаться из текстового файла с помощью потока. При запуске программы у пользователя запрашивается имя файла с клавиатуры. Результат открытия файлового потока должен проверяться. При неуспешном открытии потока имя файла должно запрашиваться повторно.

Память под хранение элементов массива должна выделяться динамически с помощью оператора new (освобождаться – с помощью delete), размер выделяемой памяти должен определяться программой автоматически по содержимому файла (а не задаваться фиксированным или вводиться пользователем с клавиатуры). Результат выделения памяти должен проверяться. При неуспешном выделении памяти должно выводиться соответствующее сообщение.

Подсчет количества числовых значений в файле, проверка элементов на корректность, вывод исходных и обработанных данных на экран консоли – как в лабораторной работе №3.

В любом месте программы продемонстрировать применение адресной арифметики для обращения к элементам массива.

Работа программы должна начинаться с очистки окна консоли. В программе должны присутствовать все необходимые подписи, приглашения к вводу и подсказки для пользователя, выводимые на русском языке. Работа программы должна завершаться по нажатию пользователем любой клавиши.

Лабораторная работа 5. Дополнение программы предыдущей лабораторной работы реализацией собственных функций и модуля.

Задание. Для программы, созданной при выполнении предыдущей л.р., необходимо провести декомпозицию всей задачи на отдельные подзадачи и реализовать их в виде функций, принимающих необходимые параметры и возвращающих полученные результаты.

При этом:

1) Весь обмен данными между всеми вызывающими и вызываемыми функциями должен осуществляться через параметры и возвращаемое значение. Использование глобальных (по отношении к функции) переменных внутри функции не допускается.

2) Нужно подумать, как сделать функции наиболее универсальными (в разумных пределах, конечно, но в предположении того, что они могут использоваться не только в данной, но и во множестве других программ — совершенно разных, но работающих с данными такой же структуры) и допускающими бОльшую степень своего повторного использования.

3) Чтобы декомпозиция была более правильная и каждая функция была достаточно универсальна, функции не должны совмещать в себе разнородные задачи, каждая из которых имеет какую-то свою специфику, т.к. это будет ограничивать универсальность функции и возможность её применения в других задачах, других программах.

Пример: функция загрузки массива не должна сама запрашивать имя файла у пользователя, т.к. это имеет привязку к конкретному способу ввода, который может не подходить для другого приложения, а значит, функция будет иметь низкую степень повторного использования. Например, если в текущем приложении имя файла вводится с консоли и запрашивается на русском языке, то в другом приложении оно может вводиться в графическом пользовательском интерфейсе и на английском языке, или может быть получено по сети, или из других источников. Также, обрабатывая возникающие при загрузке ошибки, функция загрузки массива не должна (по тем же самым причинам) сама выводить пользователю сообщение о возникшей ошибке, а должна вместо этого формировать в числовом виде (т.е. в виде, не зависящем от специфики конкретного приложения, в виде, подходящем для разных приложений) информацию о возникшей ошибке (причине и месте возникновения, еще какую-то полезную доп. информацию) и

возвращать её в вызывающую функцию, в которой эта информация уже может быть использована нужным, специфическим для каждого конкретного приложения, образом.

4) При объявлении параметров не забывать про выбор правильного способа их передачи, делающих передачу в нужном направлении не только возможной, но и эффективной.

5) Где это уместно и полезно, объявить параметры с использованием const.

6) Где это уместно и полезно, задать параметрам значения по умолчанию.

7) Если при выполнении некоторой функции могут возникать различные ошибки, то через возвращаемое значение такая функция должна возвращать код ошибки или признак успешного завершения для того, чтобы результат вызова функции можно было проанализировать и реализовать соответствующую реакцию на каждую возможную ошибку. Для представления возвращаемых значений стоит использовать перечисляемый тип.

8) Если при выполнении некоторой функции возникает ошибка, то по завершении функции аргументы для ее выходных параметров не должны содержать мусор, старые значения или частично сформированные новые.

Примерный состав функций:

1) Функция загрузки массива из файла.

Для каждого вида возможных ошибок, возникающих при выполнении этой функции, полезно через дополнительный необязательный ее параметр возвращать дополнительную информацию о возникшей ошибке для более точной и полной реакции на возникшую ошибку. Например, для ошибки открытия файла можно возвращать внутренний код ошибки открытия, который указывает на причину ошибки и который можно получить с помощью макроса errno. Для ошибки, связанной с некорректностью какого-либо его элемента, – индекс элемента и позицию указателя в файле, начиная с которой этот элемент располагается в файле.

2) Функция обработки массива по варианту задания.

3) Функция форматированного вывода массива в окно консоли.

Дополнительно полезно "научить" эту функцию выводить в текстовый файл, реализовав оба вывода только один раз, но сделав его универсальным для обоих случаев.

Также настройки формата для форматированного вывода (или формирования форматированного представления массива в виде строки) можно задавать не жестко в коде функции, а передавать через параметр(ы).

4) Если функция обработки массива изменяет исходный массив (а не создает новый), то дополнительно полезно реализовать функцию создания копии массива, чтобы можно было сохранить исходный массив для какой-то другой дальнейшей его обработки.

5) Чтобы форматированный текстовый вывод массива мог осуществляться не только в окно консоли, вместо функции п. 3 полезно реализовать функцию создания строки, содержащей форматированное текстовое представление массива. Тогда такую строку с помощью 1-2 тривиальных операторов можно будет вывести в любое место (окно консоли, окно GUI, передать по сети, сохранить в текстовый файл, вывести на печать и т.п.).

Лабораторная работа 6. Разработка алгоритма и написание программы для выполнения заданного набора действий с табличными данными с применением текстовых и двоичных файлов.

Задание 1. По своему варианту задания №11 из сборника заданий «Электронный сборник заданий по дисциплинам ‘Информатика’ и ‘Программирование на языке высокого уровня’» (Л.А. Акатнова, И.А. Евстратова, Е.К.Коншина, Л.И. Муравьёва, О.Г. Скуратовская) описать структуру одной табличной записи: смысл каждого поля, его тип, ограничения на множество, последовательность или диапазон значений.

Исходя из того, что в файле будет храниться список таких структур, выбрать оптимальный формат файлов для его хранения.

Описать все возможности и ограничения данного формата, порядок следования полей, их размеры (длины), используемые разделители полей и записей, т.е. всю ту необходимую информацию, которой было бы достаточно, чтобы на ее основе в случае текстового формата данных – неподготовленный пользователь мог во внешнем текстовом редакторе создать корректный файл исходных данных, в случае двоичного формата данных – программист мог реализовать функции экспорта и импорта данных.

Задание 2. Составить блок-схемы алгоритмов чтения и записи данных в файл.

Задание 3. Написать программу для обработки табличных данных по своему варианту.

В программе циклически осуществляется вывод запроса пользователю на выбор действия и выполнение выбранного действия. Доступные действия: «1. Создать файл с исходными данными», «2. Загрузить исходные данные», «3. Вывести исходные данные», «4. Обработать данные», «5. Завершить работу с программой». Каждое действие выбирается пользователем путем однократного нажатия на соответствующую цифровую клавишу.

При выборе действия «1» у пользователя запрашивается: имя файла, в который будут сохраняться задаваемые пользователем данные; кол-во строк таблицы; строки таблицы. Результат открытия файлового потока должен проверяться. В случае ошибки, имя файла должно запрашиваться повторно. Также необходимо проверять на существование по указанному пути файла с указанным именем и, в случае его наличия, выдавать пользователю запрос на

перезапись этого файла. Такую проверку можно реализовать средствами потока.

Для всех вводимых пользователем данных должна проверяться их корректность. В случае некорректности, соответствующее значение должно запрашиваться повторно. Имя файла можно запрашивать в последнюю очередь. Это позволит (при дополнительной программной реализации такой возможности) пользователю отказаться от записи данных в файл, но при этом сформированный массив останется в памяти и будет доступен для дальнейшей обработки.

При выборе действия «2» у пользователя должен запрашиваться ввод с клавиатуры имени файла, из которого будет загружаться таблица. Результат открытия файлового потока должен проверяться. При неуспешном открытии потока имя файла должно запрашиваться повторно. При успешной загрузке данных, исходная таблица должна сразу выводиться на экран. Должны присутствовать простейшие проверки целостности данных, загружаемых из файла.

Число строк таблицы программа должна определять автоматически по содержимому файла.

Если используется текстовый формат файлов, то при определении количества строк таблицы программа должна корректно обрабатывать ситуации, связанные с наличием в файле «лишних» пробельных символов-разделителей, которые не должны считаться ошибкой задания таблицы. Пробельные символы-разделители (пробелы, знаки табуляции) могут при задании таблицы в файле находиться в любых местах (в начале любой строки, между любыми элементами строки, в конце любой строки) и в любом количестве. Если в таблице присутствуют поля, которые могут состоять из нескольких слов, допустимо использовать собственный тип разделителей для отделения полей друг от друга.

Файл с таблицей также может содержать пустые строки или строки, содержащие только пробельные символы-разделители, в любых местах (перед первой строкой таблицы, между любыми строками таблицы, после последней строки таблицы) и в любом количестве.

Такие "пустые" (не содержащие элементов) строки в файле должны просто игнорироваться и не учитываться при подсчете строк.

Вывод результатов на экран – форматированный.

При выборе действия «3» выполняется вывод ранее созданной (в результате выполнения действия «1») или загруженной из файла в память (в результате выполнения действия «2») исходной таблицы на экран.

При выборе действия «4» выполняется фильтрация данных по заданию 11 ранее созданной или загруженной из файла в память исходной таблицы и вывод результатов обработки на экран.

Выбор пользователем действий «3» и «4» никогда не должен приводить к возникновению необработанных ошибок.

Для возврата к выбору следующего действия (т.е. в меню) после выполнения действий «1-4» должно ожидаться нажатие пользователем любой клавиши с выводом соответствующей подсказки. Меню всегда должно выводиться в чистом окне консоли. После выбора пользователем нужного действия (пункта меню) консоль также должна очищаться.

При выборе действия «5» работа программы завершается.

В программе должны присутствовать все необходимые подписи, приглашения к вводу и подсказки для пользователя на русском языке.

Лабораторная работа 7. Знакомство со средой визуальной разработки, ее встроенной справочной системой и средствами отладки

Задание 1. Создать проект VCL form Application. Ознакомиться с содержимым окон Project Manager, Tool Palette, Structure, Object Inspector. Закрыть окна и добавить их заново при помощи пункта главного меню «View» – «Tool Windows».

Задание 2. Добавить на форму компоненты TButton и TEdit, используя палитру компонентов Tool Palette. Изучить содержание файлов проекта. Найти файл, в котором находится функция WinMain приложения.

Как изменится содержимое файлов модуля формы при добавлении на нее еще одного компонента TButton?

Задание 3. Используя встроенную справочную систему, найти информацию о свойствах классов TButton и TEdit, отображаемых в окне Object Inspector на вкладке Properties.

Какие из этих свойств позволяют идентифицировать компонент? Какие из них позволяют изменить положение компонента на форме и его размеры? Какие свойства у этих компонентов схожи, какие различны?

Задание 4. Используя вкладку Events в окне Object Inspector, создать обработчик события нажатия на кнопку. Как изменились файлы модуля формы при добавлении обработчика события?

Чтобы удалить пустой обработчик события, следует удалить его определение из файла с реализацией его заголовок из заголовочного файла модуля формы. Также можно нажать на комбинацию клавиш Ctrl+S или выбрать пункт меню «File» – «Save», в этом случае обработчик события уничтожится автоматически из всех файлов модуля формы.

Добавить в обработчик события нажатия на кнопку копирование содержимого компонента TEdit в заголовок формы.

Задание 5. Ознакомиться со средствами отладки: шаг с заходом, шаг с обходом, Watch List и др.

Лабораторная работа 8. Разработка алгоритма, проектирование пользовательского интерфейса и написание программы обработки матричных данных.

Задание 1. Составить блок-схему алгоритма программы обработки матричных данных по своему варианту (задание №8 из сборника заданий «Электронный сборник заданий по дисциплинам ‘Информатика’ и ‘Программирование на языке высокого уровня’» (Л.А. Акатнова, И.А. Евстратова, Е.К.Коншина, Л.И. Муравьёва, О.Г. Скуратовская)) и определить область допустимых значений для элементов матрицы.

Задание 2. Написать программу с использованием разработанного алгоритма.

В результате выполнения л.р. необходимо сформировать:

- модуль (оформленный как полагается), содержащий следующие 3 или 4 функции:

1) загрузка матрицы из текстового файла в двумерный динамический массив с обработкой различных ошибок и возвратом информации о них.

Число строк и столбцов матрицы программа должна определять автоматически по содержимому файла.

При определении размеров матрицы программа должна корректно обрабатывать ситуации, связанные с наличием в файле «лишних» пробельных символов-разделителей, которое не должно считаться ошибкой задания матрицы.

Пробельные символы-разделители (пробелы, знаки табуляции) могут при задании матрицы в файле находиться в любых местах (в начале любой строки, между любыми элементами строки, в конце любой строки) и в любом количестве.

Файл с матрицей также может содержать пустые строки или строки, содержащие только пробельные символы-разделители, в любых местах (перед первой строкой матрицы, между любыми строками матрицы, после последней строки матрицы) и в любом количестве.

Такие "пустые" (не содержащие элементов) строки в файле должны просто игнорироваться и не учитываться при подсчете строк матрицы.

При обнаружении некорректного значения программа должна выводить пользователю информацию о его положении в матрице (номер строки, номер столбца) и, желательно, полную последовательность символов, составляющих это некорректное значение, в кавычках.

Матрица должна проверяться на прямоугольность. (Непрямоугольной считается матрица, количество элементов в строках (столбцах) которой не одинаково.)

При обнаружении непрямоугольности, программа должна выводить пользователю номер строки матрицы, в которой обнаружилось расхождение кол-ва элементов.

2) обработка матрицы по варианту задания с получением новой матрицы и/или вектора.

В результате обработки исходная матрица должна оставаться без изменений. Число параметров функции может варьироваться в зависимости от условия конкретного задания.

3) конвертирование матрицы из двумерного динамического массива в форматированное текстовое представление ([Unicode]String) в соответствии с заданным форматом элементов;

4) если в задании требуется получить вектор (как некий промежуточный или конечный результат), то нужна функция конвертирования вектора из одномерного динамического массива в форматированное (или можно неформатированное) текстовое представление ([Unicode]String); и все прочие необходимые объявления.

- консольное приложение, использующее этот модуль и выполненное во всем остальном в соответствии с требованиями л.р. №2-5, т.е. русский язык, подписи, приглашения к вводу, зацикленный ввод с проверкой корректности и т.п.). Если функция загрузки матрицы возвращает доп. информацию об ошибке, то в главном модуле приложения нужно эту информацию получать и обрабатывать, например, путем вывода пользователю информативных сообщений и, для некоторых видов ошибок, повторного запроса имени файла.
- набор файлов с исходными данными, корректными и некорректными, на которых можно полностью продемонстрировать все варианты работы программы.

Лабораторная работа 9. Разработка алгоритма, проектирование пользовательского интерфейса и написание программы обработки табличных данных с заданным набором функций.

Задание. Написать программу для обработки табличных данных по своему варианту (задание №11 из сборника заданий «Электронный сборник заданий по дисциплинам ‘Информатика’ и ‘Программирование на языке высокого уровня’» (Л.А. Акатнова, И.А. Евстратова, Е.К.Коншина, Л.И. Муравьёва, О.Г. Скуратовская)). При написании использовать разработанную в л.р №6 структуру файлов, а также алгоритмы загрузки и сохранения данных.

Приложение для редактирования и обработки табличных данных должно работать с файлами исходных данных только одного, выбранного студентом формата (двоичного или текстового). Преобразовывать данные, находящиеся в файле, из одного формата в другой, как указано в задании, не нужно. Основная цель данной л.р. – разработка оконного пользовательского интерфейса приложения для ввода, вывода и обработки табличных данных.

Что должно присутствовать в программе:

- **Для работы с табличными данными:**
  - Средства сортировки строк таблицы по убыванию и возрастанию значений в столбце, двойной щелчок на заголовок которого сделал пользователь.
  - Возможность интерактивно изменять порядок столбцов и (или) строк.
  - Возможность интерактивно изменять ширину столбцов.
  - Средства добавления/удаления строк (если не противоречит заданию, то и столбцов).
  - Если по заданию предусмотрена фильтрация данных (или удаление по какому-то условию), то должна присутствовать возможность отмены фильтра, т.е. быстрого возврата к данным до фильтрации.
- **Пункты меню «Файл»**

В меню «Файл» должны присутствовать такие стандартные пункты, как «Создать», «Открыть…», «Сохранить», «Сохранить как…», «Выход».

Пункт «Создать» предназначен для подготовки пользовательского интерфейса к вводу пользователем нового набора данных. При его выборе все элементы пользовательского интерфейса, содержащие данные текущего набора, должны очищаться от них, т.е. они должны приобретать тот же вид, который они имели на момент запуска приложения. Также, если на момент

выбора пункта «Создать» приложению известно имя файла, с которым оно в данный момент работает, то оно «забывает» его.

При выборе пункта «Открыть…»/«Сохранить как…» пользователю всегда выводится диалог выбора имени открываемого/сохраняемого файла, поэтому названия этих пунктов следует заканчивать троеточием (т.к. при выборе этих пунктов от пользователя ожидаются некоторые дальнейшие действия (в данном случае – выбор имени файла)).

При выборе пункта «Сохранить» диалог выбора имени файла для сохранения выводится, только если приложению неизвестно имя текущего файла, т.е. если пользователь в текущем сеансе работы с приложением, после его запуска или после создания нового набора данных, еще не загружал данные из какого-либо файла и не сохранял их в какой-либо файл. Ознакомиться с типичной реакцией приложений в ответ на выбор перечисленных пунктов меню можно в любом приложении-редакторе, например,«Блокнот», Microsoft Word, Microsoft Visual Studio, Embarcadero (Borland) C++ Builder/Delphi и др. Как и в большинстве других подобных приложений, пользователю должна предоставляться возможность сохранять пустой или неполный набор данных.

Пункт «Выход», находящийся внизу меню «Файл», должен отделяться от других пунктов стандартной горизонтальной чертой.

Для пунктов меню (хотя бы меню «Файл») должны быть заданы комбинации горячих клавиш и иконки.

При оформлении всех меню и их пунктов, а также при реализации их поведения, нужно придерживаться принятых стандартов, с которыми можно ознакомиться на примере других широко известных приложений.

- **Проверка сохраненности изменений в исходных данных**

Если в приложении имеются какие-либо измененные и несохраненные данные (например, исходные данные), то при попытке пользователя инициировать одно их трех действий: завершить приложение, создать новый набор данных или загрузить данные из какого-то файла, ему должен выдаваться запрос на сохранение сделанных им изменений вида «Сохранить изменения?», поскольку выполнение любого их этих трех действий приведет к потере (возможно, нежелательной) несохраненных изменений. Запрос должен содержать 3 кнопки: «Да», «Нет», «Отмена».

При нажатии на кнопку «Да» происходит сохранение в файл с текущим именем (если приложению известно имя файла, с которым оно работает, т.е. если в сеансе работы с приложением, после запуска приложения или после создания нового набора данных, пользователем уже была произведена

загрузка или сохранение данных) или появляется диалог выбора файла для сохранения (в противном случае), после чего осуществляется либо выход из приложения, либо подготовка пользовательского интерфейса к вводу нового набора данных, либо запрос у пользователя имени нового открываемого файла – в зависимости от инициированного пользователем действия. Если во время сохранения данных в файл произошла какая-либо ошибка и данные не были полностью сохранены, то пользователю должно выдаваться соответствующее сообщение и дальнейшее выполнение процедуры, связанной с текущим инициированным пользователем действием, должно прерываться. Признак наличия несохраненных изменений сбрасывается только в том случае, если данные были успешно сохранены.

При нажатии на кнопку «Нет» выход или запрос имени нового открываемого файла осуществляется сразу (без сохранения последних изменений).

При нажатии на кнопку «Отмена» окно запроса сохранения изменений данных закрывается, не приводя к каким-либо изменениям в состоянии работы приложения, т.е. пользователь просто отменяет текущее инициированное им действие — создание нового набора данных, выход из приложения или открытие нового файла. При этом признак наличия в приложении несохраненных изменений должен остаться установленным. Аналогичным образом приложение должно вести себя в случае нажатия кнопки «Отмена» в диалоге выбора файла для сохранения, если такой диалог появляется после выбора «Да» в запросе на сохранение сделанных изменений. Т.е. нажатие на кнопку «Отмена» на любом шаге (в любом диалоге (запросе)) описанной цепочки интеракций пользователя с приложением просто прерывает текущее действие, инициированное пользователем.

Описанное поведение также является типичным для большинства подобных приложений.

Аналогичные проверки и соответствующие запросы должны выводиться пользователю, если имеются несохраненные результаты работы программы.

Если несохраненных данных на момент выполнения действий, способных привести к их потере, нет, то запрос пользователю выводиться не должен.

- **Проверка корректности вводимых пользователем данных**

Для всех вводимых пользователем данных должна проверяться их корректность. Для каждого вводимого значения должен быть определен его тип и ограничения на множество, последовательность или диапазон значений. Например, при вводе гос. номера автомобиля должно проверяться его

соответствие шаблону, при вводе года рождения – нахождение в диапазоне целых чисел, например, от 1900 до текущего года, при вводе количества чего-либо – принадлежность множеству целых неотрицательных чисел, при вводе вещественных значений – корректность этого значения и его принадлежность некоторому диапазону и т.п. Корректность введенных данных можно проверять, например, непосредственно перед выполнением каких-либо расчетов (или иной обработки) на основе этих данных или при сохранении этих данных в файл. Но лучше проверку выполнять на самом раннем этапе – сразу после ввода очередного значения, чтобы ошибки не накапливались до момента полной проверки введенных данных, а обнаруживались и исправлялись пользователем постепенно, по мере их появления. Некоторые ошибки ввода можно (и лучше) предотвращать непосредственно во время ввода данных, путем задания для полей ввода масок и фильтров, не разрешающих нарушение определенного шаблона или ввод недопустимых символов. Сообщения об ошибках, связанных с некорректным вводом данных, должны содержать информацию о причине ошибки, месте ее возникновения и способе ее устранения. Перед отображением сообщения об ошибке пользователю, фокус ввода должен перемещаться на тот элемент управления (поле ввода или ячейку таблицы), который содержит неверное значение, чтобы пользователю не приходилось его искать в пользовательском интерфейсе программы, что может быть весьма затруднительно при большом количестве элементов управления на форме.

Пользовательский интерфейс

Приложение должно иметь аккуратный «масштабируемый» пользовательский интерфейс, т.е. при изменении размеров окна должно происходить изменение размеров и местоположения размещенных на нем элементов пользовательского интерфейса так, чтобы они максимально эффективно занимали всю доступную область окна. При запуске приложения его главное окно должно всегда появляться умещающимся на экране целиком, независимо от разрешения текущего графического режима. Все надписи должны быть выполнены на русском языке.

Все предупреждения, запросы, информационные сообщения и сообщения об ошибках должны быть написаны на русском языке и быть понятны неподготовленному пользователю.

Лабораторная работа 10. Дополнение программы предыдущей лабораторной работы средствами обработки исключительных ситуаций.

Задание. Дополнить программу из предыдущей лабораторной работы средствами обработки исключительных ситуаций, используя материалы лекций, встроенную справочную систему, литературу и любые другие надежные источники.

Для обработки исключительных ситуаций использовать блоки try…catch(), когда необходимо обработать возникающее исключение, и try…__finally, когда нужно освободить некоторый ресурс как в случае возникновения ошибки, так и в противном случае.

В случае возникновения исключительной ситуации пользователю выдается информативное сообщение о возникшей ошибке, и, если это возможно, класс исключения.

В тех местах программы, где могут быть сгенерированы исключения различных типов, необходимо обрабатывать каждый тип отдельно и выводить информативные сообщения об ошибках для каждого типа исключений.

Если отдельно обрабатываются исключения родительского и производных от него классов, то сначала обрабатываются все производные классы, затем родительский.

Обработчик catch(…) всегда должен обрабатываться самым последним.

Лабораторная работа 11. Разработка алгоритма и программная реализация вывода анимированного изображения или простого графического редактора с заданным набором функций.

Задание. Составить блок-схему алгоритма выполнения задания по своему варианту (задание №12 из сборника заданий «Электронный сборник заданий по дисциплинам ‘Информатика’ и ‘Программирование на языке высокого уровня’» (Л.А. Акатнова, И.А. Евстратова, Е.К.Коншина, Л.И. Муравьёва, О.Г. Скуратовская)).

Если по заданию предполагается рисование графических примитивов (не анимация), то это задание должно быть реализовано в качестве одной из функций графического редактора.

Помимо выполнения задания по варианту, графический редактор должен иметь следующие возможности:

1. рисование прямых линий различных видов (сплошных, пунктирных и т.д.) и контуров (без заливки) других графических примитивов (прямоугольников, эллипсов и т.п.);
2. рисование геометрических фигур с заливкой (вертикальными, горизонтальными полосами, диагональной штриховкой и др.)
3. выбор цвета инструмента рисования, цвета заливки и цвета фона;
4. масштабирование холста и поворот холста.

Графический редактор должен уметь загружать изображения стандартных графических форматов, таких как *.jpeg, *.png, *.bmp и др. и сохранять изображения в эти форматы.

При запуске приложения на форме должен отображаться чистый холст. Если в предыдущей лабораторной работе не была реализована проверка наличия несохраненных изменений при создании нового файла, открытии существующего и выходе из программы, то реализовать ее в графическом редакторе.

Если в задании требуется вывести анимированное изображение, то приложение должно иметь следующие функции:

1. Удаление и добавление/замена фона за анимированным объектом;
2. Анимация должна запускаться при нажатии на кнопку «Воспроизведение»;
3. Должна быть предусмотрена возможность масштабировать область воспроизведения анимации, а также, если это возможно, настройка движущегося объекта (например, если перемещается геометрическая

фигура, то предусмотреть возможность изменения ее цвета и размеров) до запуска анимации.

Для кнопки «Воспроизведение» можно использовать компонент TBitmap с подходящей по смыслу пиктограммой.

При запуске приложения должно отображаться изображение в начальном своем состоянии.

Приложение должно иметь аккуратный «масштабируемый» пользовательский интерфейс. Все подсказки, названия элементов управления и тексты запросов должны быть написаны на русском языке.

Лабораторная работа 12. Разработка приложения, выполняющего длительные операции, с неблокирующимся пользовательским интерфейсом.

Если в результате предыдущей л.р. была разработана программа с выводом анимации, то модифицировать ее так, чтобы в ней были реализованы следующие функции:

1. Старт анимации при нажатии на кнопку «Воспроизведение»;
2. Приостановка воспроизведения анимации при нажатии на кнопку «Пауза». Снятие с паузы происходит при нажатии на «Воспроизведение»;
3. Остановка анимации при нажатии на кнопку «Стоп». В этом случае изображение должно вернуться в начальное состояние.

Для кнопок «Воспроизведение», «Пауза», «Стоп» можно использовать компоненты TBitmap с подходящими по смыслу пиктограммами.

Если в результате предыдущей л.р. был реализован графический редактор, то можно дополнительно при совершения какого-либо действия пользователем (например, при нажатии на кнопку), инициировать выполнение фиктивного длительного процесса, например, запуск бесконечного цикла. Для процесса также должна быть реализована возможность его приостановить, возобновить и прервать.

Элементы пользовательского интерфейса (такие как кнопки запуска, приостановки и полной остановки анимации/процесса, закрытие программы и др.) должны оставаться доступными во время вывода анимации/выполнения процесса. Подобное поведение нужно реализовать всеми рассмотренными в лекции способами, а именно:

- при помощи обработки сообщений Windows в классе приложения (используя метод ProcessMessages либо событие OnIdle);
- при помощи компонента TTimer;
- при помощи механизмов многопоточности (класс TThread).

Список использованных источников

1. Липпман С. Язык программирования C++. Полное руководство : руководство / С. Липпман, Ж. Лажойе. — 3-е изд. — Москва : ДМК Пресс, 2006. — 1105 с. — ISBN 5-94074-040-5.

2. Культин Н.Б. Microsoft Visual C++ в задачах и примерах .— СПб. : БХВ-Петербург, 2011 .— 264 с. + 1 CD : ил.

3. Дейтел Х.М. Как программировать на C++: Пятое издание. Пер. с англ. — М: ООО «Бином-Пресс», 2008. — 1456 с.

4. Архангельский А.Я. Программирование в C++ Builder. 7-е издание. – М.: «Бином-Пресс», 2010. – 896 с. (1230 с.)

5. Седжвик Р. Алгоритмы на C++ : учебное пособие / Р. Седжвик. — 2-е изд. — Москва : ИНТУИТ, 2016. — 1772 с.

6. Джосаттис Н. Стандартная библиотека C++: справочное руководство, 2-е изд.: Пер. с англ. — М.: ООО "И.Д. Вильямс", 2014. — 1136 с.

7. Липпман С. Весь C++ от азов до совершенства, 3-е изд. – СПб.: ДМК-Пресс, 2007. – 1104 с.

8. Культин Н.Б. C++ Builder. — 2-е изд., перераб. и доп. — СПб.: БХВ-Петербург, 2012. — 464 с.

9. Чернов Э.А. Язык C++ в консольных приложениях : Учеб. пособие / ; МАДИ. Заоч. фак .— М., 2009 .— 159 с. : ил.,табл. — Библиогр.: с. 157-158.