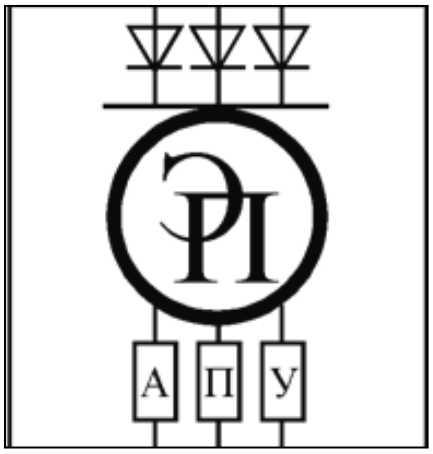
МЕЖГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«БЕЛОРУССКО-РОССИЙСКИЙ УНИВЕРСИТЕТ»

Кафедра «Электропривод и АПУ»

ИНФОРМАТИКА

Методические рекомендации к лабораторным работам для студентов специальности 1-53 01 05 «Автоматизированные электроприводы» очной и заочной форм обучения

Часть 1



Могилев 2023

УДК 004
ББК 32.97
И94

Рекомендовано к изданию
учебно-методическим отделом
Белорусско-Российского университета

Одобрено кафедрой ЭПиАПУ «11» января 2023 г., протокол № 5

Составители: В. Н. Абабурко, А. П. Корнеев

Рецензент В. В. Кутузов

Методические рекомендации к лабораторным работам для студентов специальности 1-53 01 05 «Автоматизированные электроприводы» очной и заочной форм обучения. Методические рекомендации включают цель, описание задания, обобщенный ход выполнения, содержание отчета и контрольные вопросы к каждой из лабораторных работ.

Учебное издание

ИНФОРМАТИКА

Часть 1

Ответственный за выпуск С. М. Фурманов

Корректор А. А. Подошевко

Компьютерная верстка Н. П. Полевничая

Подписано в печать . Формат 60×84/16. Бумага офсетная. Гарнитура Таймс.
Печать трафаретная. Усл. печ. л. . Уч.-изд. л. . Тираж 81 экз. Заказ №

Издатель и полиграфическое исполнение:
Межгосударственное образовательное учреждение высшего образования
«Белорусско-Российский университет».
Свидетельство о государственной регистрации издателя,
изготовителя, распространителя печатных изданий
№ 1/156 от 07.03.2019 .
Пр-т Мира, 43, 212022, г. Могилев.

Содержание

1 Лабораторная работа № 1. Архитектура персонального компьютера

Цель работы:

– ознакомление с целью преподавания дисциплины;

– практическое изучение архитектуры аппаратной части современного настольного персонального компьютера (ПК);

– изучение организации компьютерного класса в составе вычислительного центра Белорусско-Российского университета;

– получения прав доступа к программным и информационным ресурсам компьютерной сети университета;

– изучение терминологии в области информационных технологий.

Задание

Заданием к лабораторной работе является определение аппаратной части персонального компьютера, на котором выполняется лабораторная работа, а также написание реферата по заданной преподавателем тематике.

1.1 Ход выполнения работы

1.1.1 Ознакомление с целью преподавания дисциплины выполняется в соответствие с подразделом 1.1 учебной программы дисциплины, гласящим: «*целью преподавания дисциплины является обучение студентов методам решения научных и инженерных задач на персональных компьютерах*».

1.1.2 Изучение вопросов техники безопасности.

1.1.3 Изучение организации локальной вычислительной сети лаборатории и университета.

1.1.4 Регистрация в локальной вычислительной сети университета и получение прав доступа к аккаунту учебной группы и выделение рабочей области на ПК.

1.1.5 Изучение состава и характеристик аппаратной части персонального компьютера, на котором выполняется работа.

При этом обязательно определяются следующие характеристики ПК:

– тип, число ядер и тактовая частота центрального процессора;

– модель системной платы и набора системной логики;

– объем и тип оперативной памяти;

– полный объем жесткого (HDD) или твердотельного (SSD) диска и объем свободного места на диске (по возможности и его тип);

– наличие дополнительных типов внешней памяти: приводов оптических дисков, устройств чтения карт памяти и т. п.;

– тип и характеристики видеомонитора (тип и размер экрана по горизонтали, разрешающая способность по горизонтали и вертикали);

– тип видеоадаптера и объем его видеопамяти;

– тип и число каналов аудиоадаптера;

– наличие и тип акустических мониторов;

– тип (механическая, сенсорная, оптическая или иная) и число клавиш клавиатуры;

– тип корпуса системного блока;

– тип манипулятора «мышь», его интерфейс подключения и число элементов управления;

– наличие и основные характеристики периферийных устройств (принтеров, сканеров, модемов и т. п.).

1.1.6 Обзор установленного программного обеспечения ПК.

1.1.7 Определение быстродействия центрального процессора и видеоподсистемы ПК с помощью специализированных программ (тестов).

1.1.8 Теоретическое исследование заданного аппаратного устройства компьютера выполняется на основе [1, 2] и интернет-ресурсов.

1.2 Содержание отчета

Отчеты по работе № 1 оформляются индивидуально на листах формата А4 в соответствии с требованиями ГОСТ 7.32 на бумажном или электронном носителе со следующим содержанием:

– титульный лист отчета;

– состав аппаратной части персонального компьютера;

– характеристики ПК с указанием полученных данных результатов тестирования быстродействия;

– список доступного основного программного обеспечения ПК и сервера компьютерной сети;

– описание заданного аппаратного устройства ПК.

Контрольные вопросы

1 Дать определения понятий «компьютер» и «цифровой компьютер».

2 Дать определения понятий «информация» и «данные».

3 Указать какие единицы используются для определения объема информации в цифровом ПК.

4 Определить показатели для оценки качества информации.

5 Объяснить, каким образом классифицируется информация.

6 Указать основные компоненты в составе системного блока ПК.

7 Рассказать критерии классификации компьютеров.

8 Указать основные характеристики персонального компьютера.

9 Рассказать о назначении и основных характеристиках модуля центрального процессора ПК.

10 Объяснить назначение и характеристики монитора ПК.

11 Указать назначение и основные характеристики оперативной памяти персонального компьютера.

12 Перечислить устройства внешней памяти ПК.
13 Назвать устройства для ввода информации в ПК.
14 Перечислить основные периферийные устройства для ПК.
15 Объяснить основные характеристики принтера.
16 Указать основные характеристики видеоадаптера ПК.

2 Лабораторная работа № 2. Операционная система персонального компьютера

Цель работы:

– получение практических навыков работы с операционной системой (ОС) настольного ПК (семейства Windows или Linux);
– изучение состава операционной системы настольного ПК;
– подробное изучение состава программного обеспечения ПК;
– получение навыков работы с установленными стандартными приложениями операционной системы;
– практическая работа с основными утилитами настольной ОС.

Задание

Первой частью задания к лабораторной работе является составление описания ОС лабораторного ПК с подробным описанием организации его программного обеспечения (ПО). Вторая часть предполагает самостоятельное исследование заданного вида программного обеспечения настольного ПК и написание краткого реферата.

2.1 Ход выполнения работы

2.1.1 Получение доступа к аккаунту группы в компьютерной сети и регистрация на лабораторном ПК.

2.1.2 Изучение интерфейса пользователя ОС на лабораторном ПК, включающее: определение доступных пользователю элементов интерфейса рабочего стола, состав элементов системного меню, панели задач и др.

2.1.3 Определение наименования и версии установленной ОС на лабораторном ПК.

2.1.4 Изучение встроенного в состав ОС ПК и дополнительно установленных программных менеджеров файлов и анализ с их помощью структуры файловой системы ПК. При этом определяются: тип файловой системы, число логических дисков и объем доступного для пользователя свободного места на них.

2.1.5 Определение сетевого имени ПК, а также параметров компьютерной сети, установленной в учебной лаборатории, и возможности доступа к ресурсам сети университета.

2.1.6 Анализ состава ПО, установленного на ПК.

2.1.7 Использование встроенной поисковой службы в ОС для определения суммарного объема файлов по заданному критерию.

2.1.8 Работа с заданной системной программой утилитой: калькулятором, ножницами, текстовым редактором и т. п.

2.1.9 Теоретическое изучение заданного класса программного обеспечения ПК выполняется на основе [2, 4] и интернет-ресурсов.

2.2 Содержание отчета

Отчеты по работе № 2 оформляются индивидуально на листах формата А4 в соответствии с требованиями ГОСТ 7.32 на бумажном или электронном носителе со следующим содержанием:

– титульный лист отчета;

– сведения о файловой системе и объеме памяти каждого логического диска ПК;

– наименование операционной системы и номер ее версии;

– имя аккаунта пользователя и сетевое обозначение ПК;

– список ресурсов папки *документы* (*documents*) или ее аналога;

– состав доступных с компьютера сетевых ресурсов;

– описание доступных в системе принтеров и принтер, установленный по умолчанию;

– результаты поиска данных на ПК по заданному критерию;

– индивидуальное задание на исследование заданного типа ПО.

Контрольные вопросы

1 Дать определение понятия «*операционная система*».

2 Описать структуру окна приложений и документов в Windows.

3 Перечислить стандартные приложения, входящие в состав ОС Windows.

4 Назвать системные утилиты, использующиеся в ОС Windows.

5 Описать структуру Стартового меню ОС Windows.

6 Пояснить назначение и состав *Панели задач*.

7 Пояснить, каким образом в ОС настраиваются параметры *Рабочего стола*.

8 Объяснить операции, которые возможно выполнить в ОС с помощью буфера обмена.

9 Описать назначение элементов раздела *Панель управления*.

10 Показать способы получения информации о внешней памяти.

11 Отобразить сетевое окружение ПК с помощью средств ОС.

12 Пояснить настройку параметров принтеров в ОС.

13 Описать методы поиска данных с помощью средств ОС.

14 Указать способы получения данных об файловой системе ПК.

15 Назвать способы копирования файлов с помощью средств ОС.

3 Лабораторная работа № 3. Среда программирования C++Builder

Цель работы:

– изучение интегрированной среды разработчика (IDE) языка программирования C++ для настольной операционной системы;

– изучение основ языков программирования C и C++;

– получение практических навыков работы со IDE при создании линейной программы на языке C++.

Задание

Студенту следует создать консольную линейную программу на языке программирования C++ для расчета заданного выражения. При этом исходные данные переменных должны вводиться с клавиатуры, а результат расчета выводиться на экран.

3.1 Ход выполнения работы

3.1.1 Запуск интегрированной среды разработчика (IDE) языка C++.

3.1.2 Формирование проекта консольной C++ программы в IDE.

3.1.3 Подключение стандартных заголовочных файлов модулей для работы с потоками ввода-вывода, математическими функциями и функциями консольного ввода на основе [3].

3.1.4 Описание состава переменных программы.

3.1.5 Создание заголовка головной функции или ее редакция.

3.1.6 Вывод на экран запроса о последующем вводе данных выполняется передачей текстовой строки в поток ***cout*** оператором вставки.

3.1.7 Программирование ввода по запросу исходных данных с помощью стандартного потока ***cin*** и оператора извлечения.

3.1.8 Программирование заданного расчетного выражения в соответствии с синтаксисом языка C++.

3.1.9 Вывод на экран консоли результатов расчета с использованием стандартного потока ***cout*** и операторов ставки.

3.1.10 Программирование задержки закрытия окна консоли.

3.1.11 Верификация и отладка линейной программы.

3.1.12 Создание загрузочного файла программы.

3.1.13 Вычислительный эксперимент с заданными исходными данными с последующей фиксацией результатов расчета.

3.2 Содержание отчета

Отчеты по работе № 3 оформляются индивидуально на листах формата А4 в соответствии с требованиями ГОСТ 7.32 на бумажном или электронном носителе со следующим содержанием:

– титульный лист отчета;

– текст разработанной линейной программы;
– значение переменных исходных данных;
– результаты расчета по исходным данным;
– наименование и размер загрузочного (*exe*) файла программы.

Контрольные вопросы

1 Описать способы загрузки IDE языка C++ на ПК.
2 Объяснить файловые операции в IDE языка C++.
3 Описать способы сохранения информации в IDE языка C++.
4 Рассказать, каким образом в IDE языка C++ произвести компиляцию и выполнение набранной программы.
5 Пояснить работу с окнами в IDE языка C++.
6 Описать поиск и замену фрагментов текста в IDE языка C++.
7 Пояснить способы использования в IDE справочной системы.
8 Указать способы просмотра в IDE результатов расчета данных по программе.
9 Объяснить структуру линейной программы в языке C++.
10 Пояснить вывод данных на экран в языке C++.
11 Описать ввод данных в программу с клавиатуры в языке C++.
12 Указать средства отладки программы в IDE языка C++.
13 Объяснить способы приостановки автоматического закрытия окна консоли.
14 Рассказать о переменных, использованных в программе.
15 Описать создание консольного проекта в IDE языка C++.
16 Показать операторы C++, используемые для выполнения математических действий над данными.

4 Лабораторная работа № 4. Программирование базовых алгоритмов

Цель работы:
– изучение базовых конструкций алгоритмов решения задач;
– получение практических навыков по программированию базовых конструкций алгоритмов в IDE языка C++ на ПК.

Задание
При выполнении работы студентом должны быть решены две задачи.
1 Составить схему алгоритма и написать текст программы на языке C++ для решения задачи с условным переходом. Исходными данными для программы являются две переменные, значения которых вводятся с клавиатуры. Необходимо рассчитать значение искомой переменной по одному из двух альтернативных выражений в зависимости от значения переменной условия,

которое необходимо предварительно вычислить согласно заданной формуле. Значения переменной условия и переменной результата должны выводиться на экран. Все переменные имеют вещественный тип данных.

2 Разработать схему алгоритма и написать текст программы на языке программирования C++ решения циклической задачи вычисления значения функции в зависимости от значения переменной аргумента. Значения переменной аргумента должны изменяться от начального до конечного значения с заданным шагом изменения.

4.1 Ход выполнения работы

4.1.1 Составление схемы алгоритма с ветвлением для задачи 1 по [3].

4.1.2 Создание в IDE проекта для консольной программы задачи 1.

4.1.3 Редактирование или написание состава заготовки главной (main) функции программы первой задачи.

4.1.4 Подключение стандартных модулей для работы с потоками ввода-вывода и функциями консольного ввода.

4.1.5 Объявления переменных в программе первой задачи.

4.1.6 Программная реализация консольного ввода по запросу исходных данных первой задачи с использованием потока ***cin***.

4.1.7 Расчет значения промежуточной переменной-условия с использованием оператора присваивания.

4.1.8 Программная реализация алгоритма ветвления оператором ***if***.

4.1.9 Консольный вывод значений промежуточной переменной и результата с использованием потока ***cout***.

4.1.10 Программирование задержки закрытия окна консоли.

4.1.11 Верификация и отладка программы первой задачи.

4.1.12 Создание загрузочного файла программы первой задачи.

4.1.13 Вычислительный эксперимент с заданными исходными данными первой задачи с фиксацией результатов расчета.

4.1.14 Составление схемы циклического алгоритма для задачи 2.

4.1.15 Создание в IDE консольного проекта для программы задачи 2.

4.1.16 Редактирование или формирование заготовки главной (***main***) функции программы второй задачи.

4.1.17 Подключение стандартных модулей для работы с потоками ввода-вывода и функциями консольного ввода-вывода директивой ***include***.

4.1.18 Объявления переменных программы второй задачи.

4.1.19 Программирование ввода по запросу исходных данных второй задачи с использованием потока ***cin***.

4.1.20 Инициализация расчета первой точки функции и вывод ее значения на консоль.

4.1.21 Программирование основного цикла расчета значений функции (используя оператор ***while***), включающего: изменение независимой переменной, расчет зависимой переменной и вывод полученной точки на консоль потоком ***cout***.

4.1.22 Программирование задержки закрытия окна консоли.

4.1.23 Верификация и отладка циклической программы.

4.1.24 Создание загрузочного файла программы второй задачи.

4.1.25 Вычислительный эксперимент с заданными исходными данными второй задачи с фиксацией результатов расчета функции.

4.2 Содержание отчета

Отчеты по работе № 4 оформляются индивидуально на листах формата А4 в соответствии с требованиями ГОСТ 7.32 на бумажном или электронном носителе со следующим содержанием:

- титульный лист отчета;
- текст индивидуального задания первой задачи;
- схема алгоритма программы решения первой задачи;
- текст программы решения первой задачи и размер *exe*-файла;
- значения исходных данных и результата расчета первой задачи;
- текст индивидуального задания второй задачи;
- схема алгоритма решения второй задачи;
- текст второй циклической программы и размер ее *exe*-файла;
- исходные данные и результаты расчета второй программы.

Контрольные вопросы

1 Дать определение понятия «*алгоритм*».

2 Перечислить свойства алгоритма программы.

3 Указать способы описания алгоритмов программ.

4 Описать виды схем алгоритмов по ГОСТ 19.701–90.

5 Перечислить символы, используемые для изображения схем алгоритмов программ.

6 Назвать правила построения схем алгоритмов программы.

7 Рассказать об основных методах разработки схем алгоритмов.

8 Изобразить основные базовые конструкции, используемые для построения схем алгоритмов.

9 Описать средства консольного вывода данных в языке С++.

10 Показать каким образом вводятся значения переменных с клавиатуры в языке С++.

11 Запрограммировать условный переход на языке С++.

12 Реализовать на языке С++ цикл с предусловием.

13 Записать на языке С++ оператор цикла с постусловием.

14 Показать формат объявления переменных в языке С++.

15 Перечислить базовые типы скалярных данных языка С++.

16 Пояснить назначение подключенных к программе модулей.

17 Объяснить реализацию функции ***main*** в языке С++.

5 Лабораторная работа № 5. Программирование массивов

Цель работы:

– изучение регулярного типа данных на примере одномерного массива (вектора) в языках программирования С и С++;

– получение практических навыков по написанию программ расчета значений функций на заданном промежутке изменения аргумента с использованием одномерных массивов в IDE языка С++.

Задание

Необходимо составить схему алгоритма и написать текст программы на языке С++ решения циклической задачи вычисления значения зависимой переменной (функции) в зависимости от значения независимой переменной (аргумента). Значения переменной аргумента должны изменяться от начального до конечного значения с заданным шагом изменения. Все значения расчетных точек должны сохраняться в двух переменных-векторах (одномерных массивах).

5.1 Ход выполнения работы

5.1.1 Составление схемы циклического алгоритма решения по [2, 5].

5.1.2 Создание в IDE консольного проекта программы на языке С++.

5.1.3 Формирование или редактирование заготовки функции ***main***.

5.1.4 Подключение директивой ***include*** стандартных модулей для работы с потоками ввода-вывода и функциями консольного ввода.

5.1.5 Задание целочисленной константы с размером массива.

5.1.6 Объявления переменных программы: начального и конечного значений и шага аргумента, а также двух векторов (аргумента и функции).

5.1.7 Программирование ввода по запросу исходных данных (начального и конечного значений аргумента), используя поток ***cin***.

5.1.8 Расчет шага изменения аргумента, как отношение разности между конечным и начальным значением к числу точек без одной.

5.1.9 Инициализация расчета первой точки функции и вывод ее значения на консоль.

5.1.10 Программирование основного цикла расчета значений функции (оператор ***for***), включающего: расчет текущего значения вектора независимой переменной, вычисление зависимой переменной по заданному выражению и вывод результатов на консоль потоком ***cout***.

5.1.11 Программирование задержки закрытия окна консоли.

5.1.12 Верификация и отладка разработанной программы.

5.1.13 Создание загрузочного файла программы.

5.1.14 Вычислительный эксперимент с заданными исходными данными с фиксацией результатов расчета векторов функции.

5.2 Содержание отчета

Отчеты по работе № 5 оформляются индивидуально на листах формата А4 в соответствии с требованиями ГОСТ 7.32 на бумажном или электронном носителе со следующим содержанием:

- титульный лист отчета;
- текст индивидуального задания к работе;
- схема алгоритма программы;
- текст разработанной программы (***cpp***-файл);
- таблица результатов расчета векторов аргумента и функции.

Контрольные вопросы

1 Привести описание заданного одномерного массива на языке С++.

2 Дать определение типов данных «*массив*» и «*одномерный массив*».

3 Пояснить, каким образом в программе реализуется доступ к данным одномерного массива.

4 Указать, какие символы и конструкции используются в схеме алгоритма для описания обработки данных одномерных массивов.

5 Объяснить формат оператора цикла ***for*** в языке С++.

6 Показать, каким образом в языке С++ проводится начальная инициализация данных одномерных массивов.

7 Рассказать про основные операции, возможные над одномерным массивом данных в языке С++.

8 Объяснить, каким в программе на языке С++ организован ввод элементов одномерного массива с клавиатуры.

9 Показать реализацию консольного вывода значений одномерного массива в языке С++.

10 Перечислить способы форматирования консольного вывода данных с помощью ресурсов модуля ***iomanip***.

6 Лабораторная работа № 6. Поиск экстремальных значений массивов

Цель работы:

- программирование двухмерных массивов (матриц) в языке С;
- изучение алгоритмов поиска максимального (минимального) значений элемента матриц;
- применение функций ввода-вывода в программе языка С.

Задание

Разработать схему алгоритма и написать на языке С программу поиска значения экстремума (максимума или минимума) матрицы заданного типа и

размера, введенной поэлементно с клавиатуры. Результаты расчета (значение экстремума, номера строки и столбца) отображаются на экране вместе с исходной матрицей (вывод по строкам).

6.1 Ход выполнения работы

6.1.1 Составление схемы алгоритма решения задачи с использованием вложенных циклов выполняется на основе [2, 5, 6].

6.1.2 Создание в IDE консольного проекта программы на языке С.

6.1.3 Формирование или редактирование заготовки главной функции.

6.1.4 Подключение стандартных модулей для работы с функциями консольного ввода-вывода языка С.

6.1.5 Задание целочисленных констант-макросов с числом строк и столбцов матрицы директивой ***define***.

6.1.6 Объявления переменных программы: исходной матрицы, ее экстремума, двух целочисленных номеров строки и столбца экстремума, а также двух целочисленных локальных счетчиков индексов для циклов.

6.1.7 Программирование вложенных циклов (для изменения индексов строк и столбцов) поэлементного ввода по запросу исходных данных текущего элемента матрицы с помощью функции ***fscanf***.

6.1.8 Инициализация начального значения экстремума первым значением матрицы с заданием нулевых значений номеров.

6.1.9 Программирование цикла индексов строк с вложенным циклом индексов столбцов (операторы ***for***). Внутри циклов условным оператором *if* выполняется сравнение текущего элемента матрицы со значением экстремума. Если текущий элемент матрицы имеет значение лучше экстремума, то переопределяется значение экстремума и номера его столбца и строки.

6.1.10 Организация вложенных циклов ***for*** (для номеров строк и столбцов) для вывода матрицы по строкам. Внешним является цикл изменения номеров строк. Во внутреннем цикле изменения индекса столбца выводится в строку все элементы строки матрицы. Затем в конце вывода строки переводится вывод на новую строку.

6.1.11 Вывод на экран результатов поиска экстремума с индексами его строки и столбца функцией ***fprintf***.

6.1.12 Программирование задержки закрытия окна консоли.

6.1.13 Верификация и отладка разработанной программы.

6.1.14 Создание загрузочного файла программы.

6.1.15 Вычислительный эксперимент с заданной матрицей с фиксацией результатов расчета ее экстремума.

6.2 Содержание отчета

Отчеты по работе № 6 оформляются индивидуально на листах формата А4 в соответствии с требованиями ГОСТ 7.32 на бумажном или электронном носителе со следующим содержанием:

– титульный лист отчета;
– текст варианта индивидуального задания;
– схема алгоритма поиска экстремума матрицы;
– программа поиска экстремума на языке С;
– матрица исходных данных и результат расчета ее экстремума.

Контрольные вопросы

1 Дать определение понятий «*массив*» и «*двухмерный массив*».
2 Описать на языке С заданный тип двухмерного массива.
3 Пояснить реализацию в программе доступа к элементам матриц.
4 Показать, каким образом в языках С и С++ инициализируются начальные значения элементов матриц.
5 Указать, какие символы и конструкции используются в схеме алгоритма для описания обработки данных одномерных массивов.
6 Дать определение констант в языках программирования С.
7 Показать, каким образом задаются константы в языке С.
8 Записать на языке С консольный ввод значений матрицы.
9 Пояснить программную реализацию построчного вывода элементов матрицы на экран консоли в языке С.
10 Записать синтаксис оператора цикла ***for*** в языке С.
11 Объяснить алгоритм нахождения экстремального значения массивов в языках С и С++.
12 Перечислить функции языка С для вывода данных на экран.
13 Перечислить функции языка С для ввода данных с клавиатуры.
14 Указать спецификаторы описания формата данных для функций ***scanf*** и ***printf*** используемые в языке С.

7 Лабораторная работа № 7. Программирование файловых операций

Цель работы:
– изучение файлового типа данных в языках С и С++;
– получение практических навыков по написанию консольных программ на языках С и С++ для обмена данными с текстовыми файлами.

Задание
При выполнении работы необходимо написать две программы.
1 Первая программа для формирования текстового файла в стиле языка С, в который сохраняются результаты расчета однопараметрической функции на заданном интервале изменения аргумента. В каждой строке файла результата располагается значение одной расчетной точки аргумента и функции.
2 Вторая программа предназначена для чтения исходного текстового файла

с данными (созданного первой программой) в стиле языка C++. Программа должна считывать данные из файла в два массива-вектора, подсчитывая число считанных точек. Далее в программе выполняется заданный анализ данных (поиск экстремумов, сумм, средних и среднеквадратичных значений согласно варианту задания). Считанные из файла данные должны выводиться на экран вместе с результатами анализа.

7.1 Ход выполнения работы

7.1.1 Составление схемы алгоритма решения первой задачи на основе [2, 5, 6], которая должна включать следующие операции:

1) ввод по запросу значений исходных данных с клавиатуры (начального и конечного значений диапазона изменения аргумента);

2) расчет шага изменения аргумента на заданном интервале в зависимости от заданного размера массива;

3) инициализация первой точки массива-вектора аргумента и расчет соответствующей ему первой точки массива-вектора функции;

4) вывод на экран полученного значения первой расчетной точки;

5) организация цикла расчета значения переменной функции с сохранением значений в одномерном массиве;

6) в начале тела цикла необходимо вычислить текущее значение элемента массива аргумента и далее определить соответствующее значение одномерного массива функции;

7) в конце цикла нужно вывести на экран полученные текущие значений переменных аргумента и функции точки расчета;

8) сделать запрос на сохранения полученного результата в файл;

9) считать ответ в виде символа с клавиатуры;

10) при положительном ответе (символ ‘*Y*’ или ‘*y*’) выполнять следующее:

- ввести по запросу с клавиатуры имя файла для записи результата;
- открыть текстовый файл с заданным именем для записи;
- организовать цикл для вывода пар значений массивов в файл;
- вывести в теле цикла текущую точку векторов в строку файла;
- закрыть файл.

7.1.2 Создание в IDE проекта для первой программы на языке C.

7.1.3 Описание или правка заготовки главной функции программы.

7.1.4 Подключение стандартных модулей математических функций, а также консольного и файлового ввода-вывода к первой программе.

7.1.5 Описание константы-макроса для задания размера массива.

7.1.6 Описание двух переменных-векторов с помощью константы.

7.1.7 Задание файловой переменной типа ***FILE*** и символьного массива для хранения имени файла, в который будут записаны результаты.

7.1.8 Описание вещественных переменных исходных и промежуточных данных (начала и конца интервала, а также шага изменения аргумента) для расчета векторов искомой функции.

7.1.9 Объявление локальных переменных: целочисленной счетчика индексов векторов и символьной ответа на запрос сохранения результатов.

7.1.10 Ввод исходных данных (начала и конца интервала аргумента) производится после вывода на экран соответствующего запроса.

7.1.11 Расчет величины шага выполняется аналогично п. 5.1.8.

7.1.12 Инициализация расчета векторов определяется по п. 5.1.9.

7.1.13 Программирование цикла расчета значений векторов функции выполняется оператором ***for***. В теле цикла вычисляется текущие значение вектора аргумента, на основе которого рассчитывается точка вектора функции, вывод результатов на консоль выполняется функцией ***printf***.

7.1.14 Вывод запроса на сохранение результатов в файле выполняется с помощью функций ***printf*** или ***puts***.

7.1.15 Символ ответа на запрос считывается функцией ***getch***.

7.1.16 Проверка разрешения записи результатов в файл производится внутри оператора ***if*** сравнением символа ответа со значениями ***'Y'*** или ***'y'***. При соблюдении условия выполняются операции записи данных в файл:

1) функцией ***printf*** выводится запрос имени файла;
2) функции ***scanf*** с описателем ***"%s"*** считывается строка имени;
3) открывается текстовый файл для записи функцией ***fopen***;
4) запись элементов массивов выполняется циклом ***for*** для индексной переменной от 0 до значения размера массива;
5) внутри цикла вывод текущих элементов массивов в строку текстового файла реализуется с помощью функции ***fprintf***;
6) закрывается файл вызовом функции ***fclose***.

7.1.17 Программирование задержки закрытия окна консоли.

7.1.18 Верификация и отладка первой программы.

7.1.19 Создание загрузочного файла первой программы.

7.1.20 Расчет заданной функции и формирование файла с данными.

7.1.21 Разработка алгоритма второй программы, включающей этапы:

1) вывод на экран запроса для имени файла исходных данных;
2) ввод имени файла с исходными данными;
3) открытие текстового файла для чтения исходных данных;
4) контроль ошибки открытия файла исходных данных. В случае наличия ошибки выдается сообщение на экран и завершается программа;
5) инициализация искомых переменных анализа данных;
6) организация цикла для выполнения операций по анализу данных:
   - чтения пары данных точки табличной функции из файла в вектора аргумента и функции;
   - вывод считанных значений на экран;
   - расчет анализируемых значений результатов;
   - увеличение счетчика считанных данных на единицу;
7) закрытие файла исходных данных;
8) уменьшение на единицу значения счетчика считанных точек;
9) расчет средних значений результатов анализа (если задан);

10) вывод на экран значений переменных анализа данных.

7.1.22 Создание в IDE проекта для второй программы на языке C++.

7.1.23 Создание или правка заготовки функции main программы.

7.1.24 Подключение стандартных модулей математических функций, консольных и файловых потоков ввода-вывода ко второй программе.

7.1.25 Задание константы максимального размера массива-вектора.

7.1.26 Объявление векторов аргумента и функции.

7.1.27 Описание переменных файлового потока класса ***ifstream*** и строковой переменной для хранения имени файла.

7.1.28 Объявление и инициализация переменных результатов анализа данных файла. В качестве начальных значений для переменных сумм и средних значений указываются нулевые значения, для переменных максимумов – минимально возможные значения ***-1e90*** (что означает 10^{-90}), а для минимумов – максимально возможные значения ***1e90*** (10^{90}). Также задается нулевая переменная счетчика считанных строк данных из файла.

7.1.29 Ввод имени файла производится по запросу с использованием потоков ***cout*** (вывод запроса) и ***cin*** (ввод имени).

7.1.30 Открытие файла для чтения выполняется встроенной в поток функцией ***open***, с параметром ***ios::in***.

7.1.31 Проверка ошибки открытия файлового потока выполняется оператором ***if***, в условии которого указывается оператор НЕ (**!**) перед переменной файлового потока. При наличии ошибки выводится сообщение и завершается программа с возвратом кода ошибки оператором ***return -1***.

7.1.32 Определяется цикл для чтения исходных данных из файла оператором ***while***. В его условии для продолжения чтения задается признак отсутствие конца файла, определяемое с помощью встроенной в файловый поток функции ***eof*** и логического оператора НЕ (***!***).

7.1.33 Внутри тела цикла выполняются следующие операции:

1) чтение из файла данных в элементы массивов аргумента и функции производится оператором извлечения (>>), указываемого после имени файлового потока;

2) выводятся считанные данные на экран потоком ***cout***;

3) накопление значений для определения переменных сумм и средних для анализа данных с помощью оператора присваивания;

4) определяется экстремумов (минимум или максимум) с помощью оператора ***if*****.** В условии сравнивается считанное значение элемента функции с переменной-экстремумом. Если условия поиска выполняется, то переопределяется экстремум;

5) в конце цикла инкрементируется счетчик считанных данных (++).

7.1.34 Закрывается файловый поток встроенной функцией ***close***.

7.1.35 Декрементируется (--) значение счетчика считанных данных.

7.1.36 Рассчитываются средние значения с помощью оператора присваивания делением их данных на число считанных элементов.

7.1.37 Выводятся полученные результаты на экран потоком ***cout***.

7.1.38 Программируется блокировка закрытия окна консоли.

7.1.39 Верификация и отладка второй программы.

7.1.40 Создание загрузочного файла второй программы.

7.1.41 Анализируется сформированный в п.7.1.20 файл с данными.

7.2 Содержание отчета

Отчеты по работе № 7 оформляются индивидуально на листах формата А4 в соответствии с требованиями ГОСТ 7.32 на бумажном или электронном носителе со следующим содержанием:

- титульный лист отчета;
- вариант индивидуального задания;
- схема алгоритма программы первой задачи;
- схема алгоритма программы второй задачи;
- текст и размер загрузочного файла первой программы;
- текст и размер загрузочного файла второй программы;
- текстовый файл с результатами расчета первой программы;
- результаты анализа данных файла второй программой.

Контрольные вопросы

1 Дать определение понятия «*файл*» в языках С и С++.

2 Дать определение понятия «*поток*» и перечислить стандартные потоки в языке С++.

3 Перечислить типы и виды файлов в языках С и С++.

4 Изобразить операции работы с файлами в схеме алгоритма программы.

5 Назвать группы операций, используемые при работе с файлами в программах языков С и С++.

6 Пояснить, как объявляется файловый тип данных в языке С.

7 Показать, каким образом выполняется открытие и закрытие доступа к файлу с контролем ошибки в языке С.

8 Объяснить реализацию записи данных в файл на языке С.

9 Запрограммировать чтение данных из файла на языке С.

10 Перечислить функции языка С, используемые для перемещения внутри данных файла.

11 Указать спецификаторы формата данных для функции ***fprintf*** и ***fscanf*** в языке С.

12 Назвать стандартные классы файловых потоков в языке С++.

13 Показать, каким образом выполняется открытие и закрытие файловых потоков с контролем ошибки в языке С++.

14 Объяснить реализацию записи данных в файловый поток на языке С++.

15 Запрограммировать чтение данных из файлового потока на языке С++.

16 Перечислить встроенные в файловый поток функции языка С++, используемые для перемещения внутри данных файла.

17 Составить алгоритм определения максимального и минимального

значений одномерного массива.

18 Записать алгоритм определения среднего и среднеквадратичного значений одномерного массива.

8 Лабораторная работа № 8. Разработка подпрограмм

Цель работы:

– приобретение практических навыков написания программ с использованием требований структурного программирования;

– создание и использование функций в языках С и С++.

Задание

Разработать программу на языке С++ для формирования матрицы-результата размером *M* x *N*, которая содержит три подпрограммы: ввода значений исходной матрицы с клавиатуры, расчета значений матрицы-результата и вывода значений элементов матрицы по строкам. Вычисление элементов матрицы-результата производится путем преобразования заданной матрицы исходных данных по указанному математическому выражению. Имена подпрограмм следует задать самостоятельно. Обмен данными между программой и подпрограммой должен быть реализован только через механизм формальных/фактических параметров.

8.1 Ход выполнения работы

8.1.1 Составление схемы алгоритма главной функции, включающей вызовы подпрограммы ввода исходной матрицы, расчета результата и вывода по строкам исходной и результирующей матриц на основе [5–7].

8.1.2 Создание в IDE консольного проекта программы на языке С++.

8.1.3 Формирование или редактирование заготовки функции ***main***.

8.1.4 Подключение директивой ***include*** стандартных модулей для работы с потоками ввода-вывода и функциями консольного ввода.

8.1.5 Задание двух целочисленных констант с размерами матрицы.

8.1.6 Описание компактного имени типа матрицы с помощью оператора ***typedef*** с указанием типа элементов и констант размеров.

8.1.7 Задание прототипов (заголовков) трех ***void***-функций перед функцией ***main***:

1) ввода матрицы с клавиатуры с одним параметром-матрицей;

2) расчета матрицы с двумя параметрами-матрицами (исходной и результата вычисления);

3) вывода значений матрицы на экран по строкам с одним параметром матрицей.

8.1.8 Объявление в теле функции ***main*** двух локальных переменных матриц (исходных данных и результата).

8.1.9 Вызов внутри ***main*** последовательно функций с параметрами-матрицами из п. 8.1.8:

1) ввода исходной матрицы с клавиатуры;

2) расчета матрицы результата;

3) вывода на экран значений исходной матрицы;

4) вывода на экран значений матрицы результатов.

8.1.10 Приостановка автоматического закрытия окна консоли.

8.1.11 Определение функции ввода исходных данных выполняется после тела функции main следующим образом:

1) формируется заголовок функции из ее прототипа п. 8.1.7;

2) организуется структура из двух вложенных циклов ***for*** для модификации номеров строки и столбца соответственно от 0 до константы;

3) в теле вложенного цикла выводится запрос на ввод элемента матрицы с указанными индексами строки и столбца потоком ***cout***;

4) выполняется чтения данных элемента с клавиатуры через ***cin***.

8.1.12 Определение функции расчета результата реализуется так:

1) создается заголовок функции расчета по прототипу из п. 8.1.7;

2) формируется два вложенных цикла ***for*** для модификации номеров строки и столбца соответственно от 0 до константы;

3) в теле последнего цикла оператором присваивания задается расчет текущего элемента результирующей матрицы по заданному выражению.

8.1.13 Определение функции консольного вывода матрицы производится в следующей последовательности:

1) записывается заголовок функции на основе прототипа п. 8.1.7;

2) задается последовательность из двух циклов for перебора индексов строк и столбцов выводимой матрицы;

3) в цикле перебора номеров столбцов выводится на экран текущий элемент строки с пробелом, используя манипулятор ширины вывода ***setw***;

4) в конце цикла перебора номеров строк манипулятором ***endl*** в потоке ***cout*** выполняется переход на новую строку экрана консоли.

8.1.14 Верификация и отладка программы средствами IDE.

8.1.15 Создание загрузочного файла программы.

8.1.16 Расчет с заданной исходной матрицей программы.

8.2 Содержание отчета

Отчеты по работе № 8 оформляются индивидуально на листах формата А4 в соответствии с требованиями ГОСТ 7.32 на бумажном или электронном носителе со следующим содержанием:

- титульный лист отчета;
- вариант индивидуального задания;
- схема алгоритма вызывающей (головной) программы;
- текста консольной программы;
- копия окна консоли с матрицами исходных данных и полученного результата расчета, а также размер загрузочного файла.

Контрольные вопросы

1 Дать определение понятий «структурное программирование» и «подпрограмма».

2 Перечислить основные требования технологии структурного программирования.

3 Назвать основные правила создания функций в языке С++.

4 Перечислить виды формальных параметров функций, использующихся в языке С++.

5 Пояснить правила использования в языке С++ в качестве формальных параметров функций массивов.

6 Описать преимущества и недостатки формальных параметров функций языка С++ с передачей данных по значению и по ссылке.

7 Дать определение понятия «*рекурсивная функция*».

8 Рассказать правила видимости имен глобальных и локальных переменных в функциях языка С++.

9 Объяснить назначение и правила описаний формальных параметров функций со значениями по умолчанию.

10 Пояснить правила перегрузки имен функций языка С++.

11 Объяснить назначение и правила использование указателя на функцию в языке С++.

12 Пояснить назначение директивы ***define*** в языке С++.

9 Лабораторная работа № 9. Разработка модулей

Цель работы:

– получение практических навыков написания программ со статическими модулями на языке программирования С++;

– программирование типа данных «*структура*» в языке С/С++.

Задание

Требуется написать статический модуль и консольную программу, использующую его ресурсы. При этом заголовочный файл должен содержать описание типа структуры, состав полей которой формируется согласно индивидуальному варианту задания.

Модуль также должен включать две подпрограммы:

1) функцию ввода значений полей переменной-структуры заданного типа с клавиатуры и передачи ее значения по ссылке в программу с помощью формального параметра;

2) функцию вывода данных в текстовый файл, путем добавления их к концу ранее созданного файла. Строка с именем файла и данные переменной-структуры передаются в подпрограмму с помощью формальных параметров по значению.

В функции ***main*** программы следует:

– описать одномерный массив заданного размера, элементами которого являются структуры в соответствии с разработанным типом;

– функцией ввода данных задать значения элементов массива;

– создать текстовый файл, имя которого задается с клавиатуры;

– используя функцию вывода данных записать введенную информацию по каждому элементу в созданный файл.

9.1 Ход выполнения работы

9.1.1 Создание в IDE консольного проекта программы на языке C++.

9.1.2 В состав проекта с помощью IDE на основе рекомендаций [5−7] следует включить новый статический модуль, включающий два файла:

1) заголовочный с расширением *h*, в котором следует описать заданный тип структуры и объявить прототипы (заголовки) двух функций: чтения полей структуры с клавиатуры и записи полей в текстовый файл;

2) файл реализации модуля (с расширением *cpp*), в котором описываются операторные части (тела) функций, прототипы которых определены в заголовочном файле.

9.1.3 В заголовочном файле модуля следует проверить наличие элементов блокировки повторного вызова модуля программой.

9.1.4 В начале *h*-файла выполняется директивой #***include*** подключение стандартных модулей языка C++ для работы с консольными и файловыми потоками ввода-вывода, которые будут автоматически подключаться к программе, использующей разрабатываемый модуль.

9.1.5 В заголовочном файле выполняется описание содержания заданного типа структуры, объявляемой с помощью слова ***struct***.

9.1.6 В *h*-файле после структуры определяются прототипы функций:

1) ***void***-функции консольного чтения полей структуры с параметром указателем на описанный в п. 9.1.5 тип структуры;

2) функции, возвращающий целочисленный код ошибки записи данных структуры в тестовый файл с двумя параметрами:

– строкой с именем текстового файла, в конец которого будет производиться добавление данных;

– переменную типа структуры, передающей данные по значению.

9.1.7 Следует проверить корректность подключения *h*-файла в файле реализации модуля.

9.1.8 Определение функции консольного ввода полей структуры с клавиатуры реализуется в *cpp*-файле модуля следующим образом:

1) копируется из *h*-файла прототип функции в качестве заголовка;

2) определяется составной оператор тела функции;

3) в теле функции выводится на экран запрос на ввод соответствующего поля с помощью потока ***cout***;

4) далее выполняется ввод с клавиатуры данных поля структуры с использованием потока ***cin***. При этом следует учитывать, что переменная-

параметр структуры является указателем, поэтому для доступа к данным полей следует использовать оператор **->** (стрелка).

9.1.9 Определение функции записи данных структуры в файл производится в *cpp*-файле модуля в следующей последовательности:

1) копируется из *h*-файла прототип функции в качестве заголовка;

2) определяется составной оператор для тела функции;

3) в теле функции открывается файловый поток для добавления (***ios::app***) класса ***ofstream*** с именем, соответствующим параметру функции;

4) с помощью оператора ***if*** определяется наличие ошибки открытия файла, в условии которого указывается оператор НЕ (***!***) перед переменной файлового потока. При наличии ошибки выводится сообщение и завершается функция с возвратом кода ошибки оператором ***return -1***;

5) запись данных структуры в файл выполняется передачей значений полей с комментариями в файловый поток оператором вставки;

6) по окончании вывода файл закрывается вызовом встроенного в поток метода ***fclose***;

7) в конце тела функции оператором ***return*** возвращается 0.

9.1.10 Подключение разработанного модуля к программе выполняется в главном файле консольного проекта директивой ***include*** с указанием имени заголовочного файла в кавычках.

9.1.11 Описание константы с заданным размером одномерного массива (вектора) структур выполняется с помощью директивы ***define***.

9.1.12 Объявление вектора структуры выполняется в области глобальных данных программы указанием слова ***struct***, имени типа структуры и имени переменной вектора с константой размера.

9.1.13 Для хранения имени файла внутри тела функции ***main*** объявляется локальная переменная типа символьного массива. Далее выводится на экран потоком ***cout*** запрос на имя файла, которое затем считывается с клавиатуры с помощью потока ***cin***.

9.1.14 Внутри главной функции организуется цикл ***for*** перебора номеров элементов вектора структур от 0 до константы размера. В теле цикла выполняются следующие операции:

1) чтение данных полей текущей структуры вектора с клавиатуры выполняется с помощью вызова функции из разработанного модуля. В качестве параметра в функцию передается имя текущего элемента вектора. Так как данные передаются из функции в программу, то перед именем переменной массива ставится оператор взятия адреса (***&***);

2) выполняется запись данных структуры элемента вектора в текстовый вызовом второй функции из разработанного модуля. В качестве входного параметра в функцию передается имя файла и имя текущего элемента структуры. При этом отслеживается возникновение ошибки записи данных с помощью оператора ***if***, внутри условия которого вызывается указанная функция;

3) если при выполнении подпункта 2 фиксируется ошибка, то сначала блокируется закрытие окна, а далее прекращается выполнение программы

оператором ***return*** с возвратом кода ошибки.

9.1.15 По завершению цикла программируется штатная приостановка закрытия программы до нажатия любой клавиши и возврат кода успешного завершения.

9.1.16 Верификация и отладка программы и модуля средствами IDE.

9.1.17 Создание загрузочного файла программы и объектного файла статического модуля.

9.1.18 Выполнение программы с вводом в качестве исходных данных параметров реальных устройств ПК с последующим сохранением в файле.

9.2 Содержание отчета

Отчеты по работе № 9 оформляются индивидуально на листах формата А4 в соответствии с требованиями ГОСТ 7.32 на бумажном или электронном носителе со следующим содержанием:

- титульный лист отчета;
- индивидуальное задание;
- текст заголовочного и *cpp*-файлов модуля;
- текст главной программы, использующей модуль;
- объемы загрузочного файла программы и объектного файла модуля;
- текстовый файл с заданными данными полей структур вектора.

Контрольные вопросы

1 Описать назначение и состав статического модуля в языках С и С++.

2 Перечислить типы модулей в языке С++.

3 Дать определение понятия «*тип структура*» в языке С++.

4 Дать определение понятия «*тип битовое поле*» в языке С++.

5 Дать определение понятия «*тип объединение*» в языке С++.

6 Объяснить способы доступа к ресурсам статического модуля в языке С++.

7 Пояснить особенности использования указателя на структуру в языке С++.

8 Описать назначение и состав динамически подключаемой библиотеки (DLL) в языке С++.

9 Описать способы использования в программах языка С++ ресурсов из DLL.

10 Лабораторная работа № 10. Основы объектно-ориентированного программирования

Цель работы:

– изучение принципов объектно-ориентированной технологии программирования;

– получение практических навыков создания программ в среде C++ на основе технологии объектно-ориентированного программирования (ООП).

Задание

Необходимо разработать модуль, содержащий часть генеалогического дерева заданного класса, и вызывающую модуль программу с использованием объектно-ориентированной технологии программирования. Модуль должен содержать описание базового класса устройства, от которого наследованием создаются два класса-потомка в соответствии с заданным вариантом. Классы должны содержать заданные поля и следующие встроенные функции члены класса (методы):

1) функции для ввода значений полей класса клавиатуры;
2) функции вывода данных полей в указанный текстовый файл;
3) полиморфные (виртуальные) функции вывода данных полей класса на экран консоли.

В вызывающей программе следует:

– создать две переменных объекта каждого из потомков класса;

– ввести данные полей объектов с клавиатуры с помощью встроенной функции-члена (метода) класса;

– вывести значения полей объектов на экран с помощью виртуального метода;

– ввести имя текстового файла с клавиатуры;

– записать данные полей объекта в созданный текстовый файл, используя разработанные методы классов.

10.1 Ход выполнения работы

10.1.1 Создание в загруженной IDE нового консольного проекта программы на языке C++ после изучения [7, 8].

10.1.2 В состав проекта с помощью IDE добавляется новый статический модуль, включающий файлы:

1) заголовочный (*h*-файл), в котором следует объявить заданные типы иерархической структуры классов;

2) файл реализации модуля (с расширением *cpp*), в котором описываются операторные части (тела) встроенных методов в классах, объявленных в заголовочном файле.

10.1.3 В *h*-файле модуля следует проверить наличие элементов блокировки повторного вызова модуля программой.

10.1.4 В начале заголовочного файла директивой #***include*** подключаются стандартные модули языка C++ для работы с консольными и файловыми потоками ввода-вывода, которые будут автоматически подключаться к программе, использующей разрабатываемый модуль.

10.1.5 В начале *h*-файле выполняется описание содержания заданного типа базового класса, объявляемой с помощью слова ***class***:

1) в секции класса ***protected*** объявляются заданные поля базового класса;

2) в секции ***public*** класса описываются заголовки встроенных в базовый класс методов:

– *void*-функции консольного ввода данных полей класса без параметров;

– функции записи данных полей класса в тестовый файл с одним параметром-строкой для имени файла, возвращающей целочисленный результат – код ошибки открытия файла для записи;

– полиморфной (виртуальной) *void*-функции без параметров вывода данных полей на экран консоли.

10.1.6 Определение функций членов базового класса (предка) выполняется в файле реализации модуля с первоначальной проверки корректности подключения заголовочного файла.

10.1.7 Описание метода консольного ввода данных базового класса включает следующие операции:

1) копируется из описания класса в *h*-файле объявление функции ввода в качестве заголовка. Перед именем метода указывается имя класса;

2) определяется составной оператор (***{ }***) для тела функции;

3) в теле последовательно программируется задание для всех полей:

– вывод на экран запроса с помощью потока ***cout*** и оператора <<;

– ввода данных потоком ***cin*** с оператором извлечения (>>).

10.1.8 Определение метода записи данных базового класса в файл выполняется в последовательности:

1) копируется описанный в базовом классе заголовок метода записи из *h*-файла, при этом указывается имя класса перед именем метода;

2) определяется составной оператор (*{ }*) для тела метода;

3) в теле метода открывается файловый поток для добавления (***ios::app***) класса ***ofstream*** с именем, соответствующим параметру метода;

4) с помощью оператора ***if*** определяется наличие ошибки открытия файла, в условии которого указывается оператор НЕ (***!***) перед переменной файлового потока. В случае наличия ошибки отображается сообщение и завершается функция оператором ***return*** с возвратом кода ошибки ***-1***;

5) запись данных полей класса в файл выполняется передачей значений полей с комментариями в файловый поток оператором **<<**;

6) в конце записи файловый поток закрывается методом ***fclose***;

7) в конце тела метода оператором ***return*** возвращается код 0.

10.1.9 Определение метода вывода данных базового класса на консоль

выполняется в следующей последовательности:

1) копируется описанный в базовом классе заголовок метода вывода из *h*-файла с указанием имени класса перед именем метода;

2) записывается составной оператор (***{ }***) для тела метода;

3) выводятся на консоль данные полей класса с соответствующими комментариями, передачей их в поток ***cout*** оператором **<<**.

10.1.10 Добавляется в заголовочный файл описание класса первого потомка со схемой наследования ***public***. При этом в раздел ***protected*** помещаются новые заданные поля данных, а раздел public объявляются три метода класса:

1) новой *void*-функции консольного ввода данных полей класса потомка бес параметров;

2) новой функции записи данных полей класса потомка в тестовый файл с одним параметром-строкой имени файла, возвращающей целочисленный результат – код ошибки открытия файла для записи;

3) полиморфной одноименной с предком *void*-функции без параметров для вывода данных полей класса на экран консоли.

10.1.11 В *cpp*-файле модуля выполняется определение метода чтения данных первого потомка в следующей последовательности:

1) в *h*-файле копируется из описания класса потомка объявление метода ввода для заголовка с указанием имени потомка перед методом;

2) определяется составной оператор (***{ }***) для тела функции;

3) первым в теле вызывается метод ввода данных базового класса, чтобы ввести данные унаследованных полей;

4) далее последовательно программируется ввод данных потомка:

- выводится запрос с помощью потока ***cout*** и оператора **<<**;
- вводятся данные потоком ***cin*** с оператором **>>**.

10.1.12 В *cpp*-файле модуля выполняется определение второго метода записи данных класса потомка в следующей последовательности:

1) из *h*-файла копируется как заголовок из описания класса потомка объявление функции записи данных с добавлением имени класса;

2) определяется составной оператор (*{ }*) для тела метода записи;

3) первым в теле вызывается метод записи данных предка с указанием в качестве параметра строки имени файла из заголовка;

4) далее открывается файловый поток для добавления (***ios::app***) класса ***ofstream*** с именем, соответствующим параметру заголовка;

5) оператором ***if*** определяется наличие ошибки открытия файла, в условии которого указывается оператор НЕ (***!***) перед переменной файлового потока. При обнаружении ошибки выводится сообщение и завершается функция оператором ***return*** с возвратом кода ошибки ***-1***;

6) записываются данные полей потомка в файл передачей значений полей с комментариями в файловый поток оператором **<<**;

7) после записи файловый поток закрывается методом ***fclose***;

8) в конце тела метода оператором ***return*** возвращается код 0.

10.1.13 В *cpp*-файле модуля выполняется определение второго метода

записи данных класса потомка в следующей последовательности:

1) из описания класса потомка в *h*-файле копируется как заголовок объявление функции консольного вывода данных с вставкой имени класса;

2) определяется составной оператор (***{ }***) для тела метода записи;

3) первым в теле вызывается одноименный метод вывода данных предка с предварительным указанием через ***::*** имени базового класса;

4) выводятся на консоль данные класса потомка с комментариями, передачей их в поток ***cout*** оператором <<.

10.1.14 В заголовочном файле модуля выполняется объявление второго класса потомка от базового с наследованием по схеме ***public*** аналогично п. 10.1.10.

10.1.15 В файле реализации модуля производится определение всех трех методов второго класса потомка аналогично пп. 10.1.11−10.1.13.

10.1.16 В файле программы выполняется подключение созданного модуля с библиотекой классов директивой #***include*** с указанием имени заголовочного файла в кавычках.

10.1.17 Формируется или корректируется заготовка функции ***main***.

10.1.18 В начале тела главной функции объявляются две переменных (объекта) по одной от каждого из классов потомков.

10.1.19 Для хранения имени файла внутри тела функции ***main*** объявляется локальная переменная типа символьного массива. Затем выводится на экран потоком ***cout*** запрос на ввод имени файла, которое считывается с клавиатуры с помощью потока ***cin***.

10.1.20 Ввод и обработка данных первого объекта выполняется вызовом для встроенных внутрь объекта первого предка методов с помощью конструкции составного имени с точкой в последовательности:

1) для первого объекта вызывается встроенный метод ввода данных;

2) выводятся значения полей объекта на экран третьим методом;

3) вызывается встроенная функция сохранения данных полей объекта в файл внутри условия оператора ***if*** с контролем неравенства возвращаемого значения 0 (наличие ошибки). Если имеется ошибка, то выполняется приостановка закрытия окна консоли и оператором ***return*** возвращается код ошибки.

10.1.21 Ввод и обработка данных второго объекта выполняется аналогично первому, описанному в п. 10.1.7.

10.1.22 Далее программируется приостановка закрытия программы до нажатия любой клавиши и возврат кода успешного завершения.

10.1.23 Выполняется верификация и отладка программы и модуля средствами IDE.

10.1.24 Создается загрузочный файл программы и объектный файл статического модуля с библиотекой классов.

10.1.25 Выполняется программа с вводом в качестве исходных данных параметров реальных устройств ПК с последующим сохранением их в текстовом файле.

10.2 Содержание отчета

Отчеты по работе № 10 оформляются индивидуально на листах формата А4 в соответствии с требованиями ГОСТ 7.32 на бумажном или электронном носителе со следующим содержанием:

- титульный лист отчета;
- вариант индивидуального задания;
- тест статического модуля с описанием библиотеки классов;
- текст главного файла программы (с функцией ***main***);
- размеры загрузочного файла программы и объектного файла модуля с библиотекой классов;
- текстовый файл с записанными данными программы.

Контрольные вопросы

1 Перечислить преимущества ООП перед структурным программированием.

2 Назвать основные свойства ООП.

3 Дать определение понятия «*инкапсуляция*» в ООП.

4 Объяснить механизм наследования в ООП.

5 Дать определение понятия «*полиморфизм*» в ООП.

6 Перечислить правила реализации в классах С++ встроенных полиморфных функций.

7 Описать формат объявления класса в языке С++.

8 Рассказать о назначении и описании в классах конструкторов.

9 Пояснить назначение и описания в классах деструкторов.

10 Объяснить назначение и формат в языке С++ множественного наследования.

11 Пояснить назначение и формат объявления дружественных функций в классах языка С++.

12 Перечислить правила переопределения операторов в классах языка С++.

13 Описать расширенный синтаксис структуры в языке С++.

14 Описать расширенный синтаксис объединение в языке С++.

15 Пояснить описание *inline*-функций в классах языка С++.

16 Пояснить назначение и формат абстрактных классов.

17 Описать назначение и формат чисто виртуальных методов в классах языка С++.

11 Лабораторная работа № 11. Основы разработки Windows-приложений

Цель работы:

– изучение интерфейса и состава IDE системы ускоренной разработки программ (RAD) для Windows-приложений на языке C++;

– получение практических навыков по визуальному программированию однооконных приложений Windows.

Задание

Заданием на лабораторную работу является разработка программы одно-оконного приложения Windows, предназначенного для расчета и построения графика функции на задаваемом интервале аргумента. Исходные данные должны вводиться с клавиатуры в текстовые поля, а результаты расчета выводиться в окне в виде таблицы и графика.

11.1 Ход выполнения работы

11.1.1 Создание в IDE нового проекта приложения Windows.

11.1.2 Настройка параметров окна приложения по [9, 10] включает:

1) задание с помощью мыши требуемых размеров окна;

2) установка в заголовке окна наименования лабораторной работы;

3) отключение возможности изменения размеров окна при активации приложения;

4) блокирование использования в заголовке окна кнопки максимизации.

11.1.3 Разработка интерфейса ввода данных пользователем включает:

1) установку на форму окна нужного числа компонентов однострочных текстовых редакторов, в которых будут набираться пользователем исходные данные. Необходимо записать в поля значения исходных данных по умолчанию;

2) помещение на форму текстовых надписей (меток) для пояснения назначения полей подпункта 1 и иных элементов окна. При необходимости можно скорректировать цвет и параметры шрифта отображаемых пояснений;

3) разместить на форме три стандартные кнопки, у которых с помощью мышки задать удобные размеры, а также поместить тестовые ндписи: «*Расчет*», «*О программе*» и «*Выход*»;

4) переименовать установленные кнопки и поля ввода, изменив заданный автоматически номер на обозначение вводимой переменной (для поля) или выполняемую функцию (для кнопки).

11.1.4 Установка компонента для отображения таблицы результатов расчета включает:

1) установку компонента таблицы строк текста на форму;

2) задание в свойствах таблицы состава из трех столбцов;

3) настройка в свойствах или на форме мышкой размеров самой таблицы, высоты строк и ширины ее столбцов.

11.1.5 Создание отображения графика функции выполняется так:

1) помещение на форму компонента построения графика (диаграммы) и задание требуемых размеров;

2) добавление в график новой серии данных в виде точек двухмерной функции;

3) задание в свойствах белого цвета фона графика;

4) отключение в свойствах отображения легенды графика, а также отключение или редактирование титульной надписи.

11.1.6 При необходимости в файле реализации модуля формы подключается директивой #***include*** стандартный модуль математики ***math***.

11.1.7 Описание переменных программы выполняется в файле реализации модуля формы. Объявляются следующие переменные:

1) два указателя на тип ***double*** для динамических одномерных массивов хранения результатов расчета функции;

2) начала и конца интервала аргумента, шага изменения аргумента типа ***double***;

3) целочисленная переменная числа расчетных точек функции;

4) описываются переменные типа ***double*** параметров функции (если они необходимы).

11.1.8 Определение заданного расчетного выражения выполняется в виде функции с одним параметров, возвращающей результат типа ***double***.

11.1.9 Программирование события нажатия кнопки для расчета функции выполняется в следующей последовательности:

1) выполняется двойной клик мышкой по изображению кнопки. В автоматически раскрывшемся редакторе кода отображается пустое тело функции-метода обработчика события, где и программируется остальное;

2) чтение исходных данных с текстовых полей с последующим преобразованием строковых значений в вещественный тип для переменных начала и конца интервала или целочисленное для числа точек;

3) создание двух динамических векторов независимой переменной (аргумента) и зависимой переменной (значений функции) оператором ***new***;

4) определение значения шага аргумента аналогично п. 5.1.8;

5) инициализация первой точки вектора аргумента и расчет первой точки вектора функции с использованием подпрограммы п. 11.1.8;

6) организуется цикл ***for*** для расчета остальных точек, в теле которого выполняются операции:

– определяется текущее значение вектора аргумента путем прибавления шага к предыдущему значению массива;

– рассчитывается текущее значение вектора функции с использованием подпрограммы п. 11.1.8;

7) в свойствах компонента задается число строк таблицы на единицу большим, чем число расчетных точек;

8) в первой строке таблицы отображаются подписи столбцов;

9) задается цикл ***for*** для вывода всех рассчитанных точек в поля таблицы, в теле которого выполняются операции:

– в первом столбце отображается номер точки с преобразованием целочисленного значения в строковый тип;

– во втором столбце выводится текущее значение вектора аргумента с преобразованием вещественных данных в строковый тип;

– в третьем столбце показывается значение текущей точки вектора функции с преобразованием вещественных данных в строковый тип;

10) очищается серия данных графика;

11) задается цикл ***for*** для вывода всех рассчитанных точек, в теле которого к серии данных графика добавляется текущая точка функции;

11.1.10 Кодирование события нажатия кнопки для завершения работы выполняется в следующей последовательности:

1) выполняется двойной клик мышкой по изображению кнопки;

2) в автоматически раскрывшемся редакторе кода отображается пустое тело функции-метода обработчика события двойного клика по кнопке. Внутри тела метода необходимо выполнить:

– уничтожить оператором ***delete*** две динамические переменные одномерных массивов с результатами расчетов;

– вызвать метод закрытия для объекта формы.

11.1.11 Кодирование события нажатия кнопки с информацией о программе выполняется двойным кликом мыши по ее изображению. В раскрывшемся теле метода обработчика события записывается функция отображения окна сообщения с данными об авторе программы (фамилия, имя, отчество и номер учебной группы).

11.1.12 Выполняется настройка приложения: задается пиктограмма, указывается версия и иная информация по программе.

11.1.13 Выполняется верификация и отладка программы.

11.1.14 Создается загрузочный файл программы.

11.1.15 Выполняется программа с заданными исходными данными.

11.2 Содержание отчета

Отчеты по работе № 11 оформляются индивидуально на листах формата А4 в соответствии с требованиями ГОСТ 7.32 на бумажном или электронном носителе со следующим содержанием:

- титульный лист отчета;
- текст варианта индивидуального задания;
- текст главного *cpp*-файла проекта (с функцией ***WinMain***);
- тексты заголовочного и *cpp*-файла модуля приложения;
- копия окна приложения после расчета функции.

Контрольные вопросы

1 Описать состав интерфейса окна Windows-приложения.

2 Перечислить основные элементы интерфейса RAD-системы для приложений Windows.

3 Описать состав проекта программы приложения Windows.

4 Пояснить, каким образом разрабатывается интерфейс для ввода данных Windows-приложения в среде RAD.

5 Объяснить особенности программирования вывода данных в текстовую таблицу в среде RAD.

6 Перечислить основные правила программирования вывода результатов в виде графика в среде RAD.

7 Пояснить основные свойства классов визуальных компонентов RAD.

8 Перечислить принципы формирования интерфейса приложения Windows.

12 Лабораторная работа № 12. Разработка интерфейса Windows-приложений

Цель работы:

– изучение методов разработки интерфейса многооконных Windows-приложений в среде визуального программирования;

– получение практических навыков по визуальному программированию многооконных Windows-приложений.

Задание

Заданием на лабораторную работу является разработка программы приложения Windows, предназначенной для расчета и построения графика функции по заданному выражению. Исходные данные должны вводиться как с клавиатуры, так и из текстового файла на главном окне. Результаты расчета должны выводиться на экран в виде таблицы и графика, которые должны располагаться на отдельных окнах и иметь возможность сохраняться в файлы. Приложение должно содержать главное меню, в котором должны быть разделы ввода исходных данных, расчета, просмотра результатов расчета, справочной информации. В главной форме приложения должна быть строка статуса со справочной информацией по выбранному элементу.

12.1 Ход выполнения работы

12.1.1 Создание в IDE нового проекта приложения Windows.

12.1.2 Настройка параметров главного окна приложения включает: задание требуемых размеров окна, установка в заголовке наименования лабораторной работы, отключение изменения размеров окна, блокирование использования в заголовке окна кнопки максимизации.

12.1.3 Проектирование интерфейса главной формы выполняется как:

1) установка на форму компонента главного меню, в котором с помощью специального редактора формируются подменю:

– «*Данные*», которое должно содержать исполняемые пункты: «Принять данные», «Сохранить в файл», «Загрузить из файла» и «Выход из программы»;

– «*Расчет*», которое само является выполняемым пунктом;

– «*Результаты*», включающее пункты «*Таблица*» и «*График*»;

– «*О программе*» − исполняемый пункт;

2) программирование компонента строки состояния включает:

– перенос на форму компонента из стандартной библиотеки;

– добавляется вручную заголовок метода отображения подсказки в раздел ***public*** описания класса главной формы в *h*-файле модуля;

– в *cpp*-файле модуля главной формы производится описание тела указанного метода, где передается в свойство текста компонента строки статуса значение подсказки;

– производится перепрограммирование нового обработчика события создания главной формы, в котором присваивается значение метода соответствующему свойству приложения;

3) размещение на главной форме полей текстовых редакторов для ввода исходных данных выполняется аналогично (1) из п. 11.1.3;

4) расположение текстовых надписей аналогично (2) из п. 11.1.3;

5) помещаются на форму два невизуальных компонента диалоговых окон ввода имени файла соответственно для сохранения и чтения данных.

12.1.4 Формирование интерфейса окна отображение таблицы результатов выполняется в следующей последовательности:

1) добавляется в проект новая форма, которая описывается отдельным статическим модулем;

2) настраиваются свойства формы окна аналогично п. 12.1.2;

3) устанавливается на форме компонент таблицы строк текста с последующей настройкой свойств аналогично п. 11.1.4;

4) размещаются на форме две стандартные кнопки с заданными тестовыми надписями: «*Закрыть окно*», «*Сохранить в файл*»;

5) помещается на форму невизуальный компонент диалогового окна ввода имени файла для сохранения.

12.1.5 Проектирование интерфейса окна отображения графика результатов выполняется в последовательности:

1) добавляется в проект новая форма окна графиком, описываемая собственным статическим модулем;

2) настраиваются свойства окна графика аналогично п. 12.1.2;

3) устанавливается на форме компонент диаграммы с последующей настройкой свойств аналогично п. 11.1.5;

4) размещаются на форме две стандартные кнопки с заданными тестовыми надписями: «*Закрыть окно*», «*Сохранить в файл*»;

5) помещается на форму невизуальный компонент диалогового окна ввода имени файла для сохранения графических данных.

12.1.6 В *cpp*-файле модуля главного окна выполняется директивой #***include*** подключение двух модулей окон отображения результатов.

12.1.7 Для работы с файлами исходных данных в файле реализации модуля главной формы выполняются подключения модуля ***fstream*** и при необходимости модуль математики ***math***.

12.1.8 Описание глобальных данных приложения производится в *cpp*-файле главной формы в следующем порядке:

1) определяются два указателя на тип ***double*** для динамических одномерных массивов хранения результатов расчета функции;

2) задается целочисленная переменная числа расчетных точек;

3) объявляются три переменные (начала и конца интервала аргумента, а также шага изменения аргумента) типа ***double***;

4) при необходимости описываются переменные типа ***double*** параметров функции;

5) заданное расчетное выражение объявляется в виде функции с одним параметров, возвращающей результат типа ***double***.

12.1.9 В *cpp*-файле модуля формы таблицы выполняется ее подключение к глобальным данным. Для этого копируются с главной формы объявления переменных, выполненные в (1) и (2) п. 12.1.8. Они в модуле таблицы объявляются внешними, используя слово ***extern***.

12.1.10 В файле реализации модуля графика выполняется описание внешних переменных указателей на векторы расчета и числа расчетных точек аналогично п. 12.1.9.

12.1.11 Кодирование пунктов подменю «*Данные*» главного меню:

1) производится двойной клик мышью по пункту «*Принять данные*», что раскроет тело обработчика соответствующего события. Внутри тела выполняется чтение исходных данных из текстовых полей формы с преобразованием типов аналогично (2) из п. 11.1.9;

2) выполняется двойной клик мышью по пункту «*Сохранить в файл*». В раскрывшемся теле метода обработчика события производится:

- внутри условия оператора ***if*** вызывается диалоговое окно ввода имени файла сохранения данных для контроля правильности выбора;
- при выборе имени в теле ***if*** объявляется и открывается для записи файловый поток класса ***ofstream*** с полученным из диалога именем;
- последовательно передаются в созданный поток значения переменных исходных данных вместе с символом новой строки файла;
- в конце закрывается файловый поток методом ***fclose***;

3) выполняется двойной клик мышью по пункту «*Загрузить из файла*». В раскрывшемся теле метода обработчика программируется:

- в условии оператора ***if*** вызывается диалоговое окно ввода имени для открытия файла данных, чтобы отследить правильности выбора;

– в теле ***if*** объявляется и открывает для чтения файловый поток класса ***ifstream*** с полученным из диалога именем;

– из созданного потока последовательно считываются значения переменных исходных данных в порядке, записанном в подпункте 2;

– выводятся полученные из файла данные в соответствующие текстовые поля формы после преобразования чисел в тестовые строки;

– в конце тела ***if*** закрывается файловый поток методом ***fclose***;

4) производится двойной клик по пункту «*Выход из программы*». В раскрывшемся теле метода программируется выход аналогично п. 11.1.10.

12.1.12 Программирование пункта меню «*Расчет*» выполняется так:

1) выполняется двойной клик мышкой по пункту. В автоматически раскрывшемся теле метода обработчика программируется остальное;

2) ***new*** создается два динамических вектора аргумента и функции;

3) определяется значения шага аргумента аналогично п. 5.1.8;

4) инициализируется первая точка вектора аргумента и находится первая точка вектора функции с использованием функции из (5) п. 12.1.8;

5) организуется цикл ***for*** расчета остальных точек, в теле которого:

– определяется текущее значение вектора аргумента путем прибавления шага к предыдущему значению массива;

– рассчитывается текущее значение вектора функции с использованием функции из (5) п. 12.1.8.

12.1.13 Программирование отображения результатов задается так:

1) выполняется двойной клик мышкой по пункту «*Таблица*». В автоматически раскрывшемся теле метода обработчика программируется вызов встроенного в форму таблицы метода ее модального отображения;

2) производится двойной клик мышкой по пункту «*График*». В раскрывшемся теле метода обработчика кодируется вызов метода модального отображения, встроенного в форму графика.

12.1.14 Кодирование выбора пункта «*О программе*» производится после двойного клика по нему мышкой. В раскрывшемся теле обработчика вызывается функция отображения окна сообщения с данными об авторе.

12.1.15 Программирование события отображения окна таблицы выполняется после перехода на его форму в следующим образом:

1) в IDE раскрывается тело обработчика отображения формы;

2) в начале тела в свойствах компонента таблицы задается число ее строк на единицу большим, чем число расчетных точек;

3) в первой строке таблицы отображаются подписи столбцов;

4) задается цикл ***for*** для вывода всех рассчитанных точек в поля таблицы, в теле которого выполняются операции:

– в первом столбце отображается номер точки с преобразованием целочисленного значения в строковый тип;

– во втором столбце выводится текущее значение вектора аргумента с преобразованием вещественных данных в строковый тип;

– в третьем столбце показывается значение текущей точки вектора функции с преобразованием вещественных данных в строковый тип.

12.1.16 Определение обработчиков событий для кнопок формы отображения таблицы результатов выполняется в последовательности:

1) подключается к *cpp*-файлу модуля формы модуль ***fstream***;

2) выполняется двойной клик по кнопке «*Сохранить в файл*», что вызовет раскрытие тела обработчика, в котором кодируется следующее:

– внутри условия оператора ***if*** вызывается диалоговое окно для ввода имени файла сохранения результатов с контролем выбора;

– при выборе имени в теле ***if*** объявляется и открывается для записи файловый поток класса ***ofstream*** с полученным из диалога именем;

– в цикле ***for*** последовательно передаются в созданный поток значения всех точек результатов из переменных векторов;

– закрывается файловый поток методом ***fclose***;

3) выполняется двойной клик по кнопке «*Закрыть окно*», что вызовет раскрытие тела обработчика, в котором вызывается встроенный в объект формы метод закрытия окна.

12.1.17 Кодирование события отображения окна графика выполняется после перехода на его форму следующим образом:

1) в IDE раскрывается тело обработчика отображения формы;

2) очищается серия данных графика;

3) задается цикл ***for*** для вывода всех рассчитанных точек, в теле которого к серии данных графика добавляется текущая точка функции.

12.1.18 Кодирование события отображения окна графика выполняется после перехода на его форму следующим образом:

1) выполняется двойной клик по кнопке «*Сохранить в файл*», что вызовет раскрытие тела обработчика, в котором кодируется следующее:

– внутри условия оператора ***if*** вызывается диалоговое окно для ввода имени файла сохранения графика с контролем правильности выбора;

– в теле ***if*** вызывается встроенный в объект графика метод записи данных в графический файл, куда передается имя из окна диалога;

2) выполняется двойной клик по кнопке «*Закрыть окно*», что вызовет раскрытие тела обработчика, в котором вызывается встроенный в объект формы метод закрытия окна.

12.1.19 Выполняется настройка приложения: задается пиктограмма, указывается версия и иная информация по программе.

12.1.20 Выполняется верификация и отладка программы.

12.1.21 Создается загрузочный файл программы.

12.1.22 Выполняется программа с заданными исходными данными.

12.2 Содержание отчета

Отчеты по работе № 12 оформляются индивидуально на листах формата А4 в соответствии с требованиями ГОСТ 7.32 на бумажном или электронном носителе со следующим содержанием:

- титульный лист отчета;
- текст варианта индивидуального задания;
- тексты *h*- и *cpp*-файлов всех модулей приложения;
- главный *cpp*-файл проекта (с функцией ***WinMain***);
- копии экранов каждой формы приложения с результатами.

Контрольные вопросы

1 Объяснить состав проекта многооконного приложения.
2 Пояснить правила работы с диалоговыми окнами файлов.
3 Объяснить, каким образом выполнен обмен данными между формами приложения.
4 Пояснить, каким образом преобразуются данные в программе.
5 Перечислить способы программирования команд управления в Windows приложении.
6 Пояснить программирование главного меню приложения.
7 Объяснить способ отображения подсказок в строке статуса.
8 Пояснить программирование печати данных в Windows.

13 Лабораторная работа № 13. Программирование графики приложений

Цель работы:

– изучение компонентов обработки графических данных в среде визуального программирования языка C++;

– получение практических навыков визуального программированию графических элементов для Windows-приложений.

Задание

Заданием на лабораторную работу является разработка программы приложения Windows, предназначенной для загрузки рисунка из графического файла на форму, добавления в рисунок заданного числа графических элементов случайно выбранного цвета (тип фигуры № 1) цвета и размеров, а также в случайной позиции, следует предусмотреть добавления текстовой надписи в изображение. Цвет фигуры № 2 задается с помощью специального диалогового окна, а фона рисунка задается согласно варианту задания. Фигура № 2 применяется в качестве фонового рисунка. В центре окна выводится текстовая надпись из поля редактора текста. Необходимо предусмотреть возможность изменения размеров окна приложения с изменением размера рисунка.

В программе должна быть возможность сохранить результаты рисования в графический файл.

13.1 Ход выполнения работы

13.1.1 Создание в IDE нового проекта приложения Windows.

13.1.2 Настройка параметров главного окна приложения включает: задание требуемых размеров окна, установка в заголовке наименования лабораторной работы.

13.1.3 Создание интерфейса приложения выполняется в следующем порядке:

1) ставятся на форму компоненты четырех стандартных кнопок:
   - для очистки экрана (заливка всего поля данных цветом);
   - для вставки заданного числа фигур № 1 в рисунок;
   - для отображения фонового рисунка из фигур № 2;
   - для вставки текстовой надписи в центр рисунка;

2) устанавливается на форму компонент панели кнопок инструментов, где создается следующий набор инструментальных кнопок:
   - загрузка изображения из *bmp*-файла в поле компонента данных;
   - сохранение изображения из поля данных в *bmp*-файл;
   - выбор цвета линий рисования фигур;
   - настройка параметров шрифта текстовых надписей;
   - выход из программы;

3) располагаются два кнопочных переключателя для дискретного изменения следующих целочисленных данных:
   - числа выводимых фигур № 1;
   - числа фигур № 2, выводимых по горизонтали фонового узора;

4) размещаются три текстовых поля для реализации ввода данных:
   - число фигур № 1 (данные первого переключателя);
   - число фигур № 2 (данные второго переключателя);
   - отображаемая в центре надпись;

5) размещается компонент данных формируемого изображения;

6) задается компонент с набором пиктограмм для панели кнопок (2);

7) размещаются два невизуальных компонента диалоговых окон выбора имени графического файла при вводе и выводе данных;

8) устанавливается компонент стандартного диалога выбора цвета;

9) помещается компонент задания параметров шрифта.

13.1.4 Выполняется описание глобальных данных приложения в верхней части cpp-файла модуля формы:

1) директивой #***include*** подключаются стандартные модули для генератора случайных чисел (***stdlib*** и ***time*** или ***cstdlib*** и ***ctime***);

2) двух целочисленных значений размеров формы по *X* и *Y*;

3) переменная класса цвета;

4) указатель на гарнитуру шрифта текста рисунка.

13.1.5 После переменных определяется *void*-функция, определяющая текущие размеры рисунка и передающая в переменные (1) из п. 13.1.4.

13.1.6 Определяются обработчики нажатия стандартных кнопок:

1) выполняется двойной клик по кнопке очистки рисунка, в раскрывшемся теле обработчика события программируется следующее:

– вызывается функция размеров окна;

– по полученным данным в поле рисунка строится равномерно залитый заданным цветом прямоугольник;

2) производится двойной клик по кнопке рисования фигуры № 1, в раскрывшемся теле обработчика события выполняется следующее:

– описываются локальные целочисленные переменные: числа фигур № 1 (получающее значение от первого переключателя), две переменных расположения центра и одна размера фигуры;

– вызывается функция размеров окна;

– организуется цикл ***for*** для вывода заданного числа фигур, внутри цикла которого выполняются остальные операции;

– по полученным данным генерируются случайные координаты центра расположения фигуры и ее размера;

– устанавливается случайное значение цвета от 0 до 65535;

– вызывается функция (одна или две) для построения заданной фигуры по найденным ранее случайным координатам;

3) выполняется двойной клик по кнопке заполнения фона фигурой № 2, далее в теле обработчика события программируется следующее:

– описываются локальные целочисленные переменные: два числа фигур по горизонтали (данные от второго переключателя) и вертикали, две переменных размера фигуры по координатам *X* и *Y*;

– вызывается функция размеров окна;

– определяются по данным окна размеры фигур и число фигур по вертикали;

– устанавливается заданный цвет контура фигур;

– организуется два вложенных цикла ***for*** для вывода заданного числа фигур по горизонтали и вертикали;

– внутри циклов по полученным данным вызывается функция для построения заданной фигуры № 2;

4) производится двойной клик по кнопке вывода надписи, в раскрывшемся теле обработчика события кодируется следующее:

– оператором ***if*** проверяется содержимое поля текстовой надписи и если оно не нулевое, то выполняются операции, описанные ниже;

– устанавливается значения параметров шрифта из соответствующей глобальной переменной;

– вызывается функция определения размеров окна;

– в середину окна производится вывод текстовой надписи.

13.1.7 Программирование нажатия инструментальных кнопок:

1) производится двойной клик по кнопке загрузки изображения, в раскрывшемся теле обработчика события программируется следующее:

– в операторе ***if*** вызывается диалог открытия графического файла;

– если выполнен выбор имени файла, то с помощью встроенного метода графические данные загружаются в рисунок;

2) выполняется двойной клик по кнопке сохранения изображения, в раскрывшемся теле обработчика события кодируется следующее:

– вызывается в операторе ***if*** диалог записи графического файла;

– если выполнен выбор имени файла, то с помощью встроенного метода графические данные передаются из рисунка в указанный файл;

3) осуществляется двойной клик по кнопке выбора цвета, в раскрывшемся теле обработчика события задается следующее:

– в операторе ***if*** вызывается диалог выбора цвета;

– если выполнен выбор цвета, то выбранный код цвета передается в соответствующую глобальную переменную;

4) исполняется двойной клик по кнопке настройки шрифта, в раскрывшемся теле обработчика события кодируется следующее:

– вызывается в операторе ***if*** диалог выбора параметров шрифта;

– если параметры шрифта выбраны, то они присваиваются соответствующему глобальному указателю шрифта;

5) производится двойной клик по кнопке закрытия приложения, в раскрывшемся теле обработчика события программируется следующее:

– в условии оператора ***if*** вызывается окно диалога с ответом на вопрос «*Закрыть приложение?*»;

– если получен положительный ответ, то вызывается встроенный метод закрытия формы (приложения).

13.1.8 Обработка изменения размеров окна выполняется в соответствующем обработчике события, раскрываемого с помощью IDE. Внутри тела обработчика выполняется следующее:

1) устанавливается автоматическое масштабирование в свойствах компонента рисунка, в поле которого выполняется отображение данных;

2) оператором ***if*** ограничивается минимальная ширина рисунка, при которой вывод графических данных имеет смысл;

3) ограничивается минимальная высота рисунка оператором ***if***;

4) выполняется коррекция ширины рисунка на основе ширины окна без определенной целочисленной константы ширины;

5) производится пересчет высоты рисунка на основе разности высоты окна с определенной целочисленной константой высоты.

13.1.9 Выполняется настройка приложения: задается пиктограмма, указывается версия и иная информация по программе.

13.1.10 Производится верификация и отладка программы.

13.1.11 Формируется загрузочный файл программы.

13.1.12 Выполняется программа по генерации фона и заданных фигур с сохранением результатов в растровый файл.

13.2 Содержание отчета

Отчеты по работе № 13 оформляются индивидуально на листах формата А4 в соответствии с требованиями ГОСТ 7.32 на бумажном или электронном носителе со следующим содержанием:

- титульный лист отчета;
- текст индивидуального задания;
- главный *cpp*-файл проекта приложения;
- тексты *h*- и *cpp*-файлов модуля приложения;
- графический файл сохраненного изображения фона;
- копии экрана приложения не менее чем с пятью случайно расположенными фигурами № 1.

Контрольные вопросы

1 Пояснить способы программирования графических данных.
2 Перечислить основные графические функции рисования.
3 Указать способы загрузки графических данных из внешнего файла.
4 Пояснить согласование изменений размера окна с его данными.
5 Рассказать о программировании случайных чисел в языке C++.
6 Перечислить компоненты среды программирования для работы с графическими данными.
7 Описать способы создания пиктограмм для Windows-приложений.

Список литературы

1 **Горнец, Н. Н.** ЭВМ и периферийные устройства. Устройства ввода-вывода: учебник / Н. Н. Горнец, А. Г. Рощин. − 2-е изд., стер. − Москва: Академия, 2016. – 224 с.

2 **Гуриков, С. Р.** Информатика: учебник / С. Р. Гуриков. – Москва: Форум, 2014. – 464 с.

3 **Дорогов, В. Г.** Основы программирования на языке С: учебное пособие / В. Г. Дорогов, Е. Г. Дорогова, Л. Г. Гагарина. – Москва: ФОРУМ, 2011. – 224 с.

4 Информатика. Базовый курс: учебное пособие / Под ред. С. В. Симоновича. – Санкт-Петербург: Питер, 2012. – 640 с.

5 Информатика. Основы программирования на языках С и С++: методические рекомендации для студентов специальности 1-53 01 05 «Автоматизированные электроприводы»: в 4 ч. / Сост. В. Н. Абабурко. – Могилев: Белорус.-Рос. ун-т, 2010. – Ч. 1. – 40 с.

6 Информатика. Основы программирования на языках С и С++: методические рекомендации для студентов специальности 1-53 01 05 «Автоматизи-

рованные электроприводы»: в 3 ч. / Сост. В. Н. Абабурко. – Могилев: Белорус.-Рос. ун-т, 2010. – Ч. 2. – 47 с.

7 Информатика. Основы программирования на языках С и С++: методические рекомендации для студентов специальности 1-53 01 05 «Автоматизированные электроприводы»: в 3 ч. / Сост. В. Н. Абабурко. – Могилев: Белорус.-Рос. ун-т, 2010. – Ч. 3. – 48 с.

8 Информатика. Основы программирования на языках С и С++: методические рекомендации для студентов специальности 1-53 01 05 «Автоматизированные электроприводы»: в 4 ч. / Сост. В. Н. Абабурко. – Могилев: Белорус.-Рос. ун-т, 2011. – Ч. 4. – 46 с.

9 Информатика. Основы программирования в среде С++Builder: методические рекомендации для студентов специальности 1-53 01 05 «Автоматизированные электроприводы»: в 2 ч. / Сост. В. Н. Абабурко. – Могилев: Белорус.-Рос. ун-т, 2011. – Ч. 1. – 49 с.

10 Информатика. Основы программирования в среде С++Builder: методические рекомендации для студентов специальности 1-53 01 05 «Автоматизированные электроприводы»: в 2 ч. / Сост. В. Н. Абабурко. – Могилев: Белорус.-Рос. ун-т, 2011. – Ч. 2. – 50 с.