МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования
«Юго-Западный государственный университет»
(ЮЗГУ)

Кафедра механики, мехатроники и робототехники

ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ В МЕХАТРОНИКЕ

Методические указания по выполнению лабораторных работ по дисциплине
«Объектно-ориентированное программирование в мехатронике»
по направлению 15.03.06 - «Мехатроника и робототехника»

Курск 2016

УДК 621

Составитель: П.А. Безмен

Рецензент
Кандидат технических наук, доцент кафедры механики,
мехатроники и робототехники
Е.Н. Политов

Объектно-ориентированное программирование в мехатронике: методические указания по выполнению лабораторных работ по дисциплине «Объектно-ориентированное программирование в мехатронике» по направлению 15.03.06 - «Мехатроника и робототехника» / Юго-Зап. гос. ун-т; сост.: П.А. Безмен; Курск, 2016. 80 с.: ил. 1, табл. 5. Библиогр.: с. 80.

Методические указания содержат основные требования по выполнению и защите лабораторных работ, включают индивидуальные задания и справочный материал.

Методические указания соответствуют требованиям программы, утверждённой учебно-методическим объединением (УМО).

Предназначены для студентов направления 15.03.06 – «Мехатроника и робототехника» всех форм обучения.

Текст печатается в авторской редакции

Подписано в печать . Формат 60x84 1/16.
Усл. печ. л. . Уч.-изд. л. . Тираж 100 экз. Заказ. Бесплатно.
Юго-Западный государственный университет.
305040, Курск, ул. 50 лет Октября, 94.

Содержание

Введение

Объектно-ориентированный язык – это язык программирования, на котором программа задается описанием поведения совокупности взаимосвязанных объектов. Объектно-ориентированное программирование имеет дело с объектами.

Объектно-ориентированные языки включают следующие основные черты: инкапсуляция данных, полиморфизм, наследование.

Объекты могут включать закрытые (private) данные и правила их обработки, доступные только объекту, защищенные (protected), доступные объекту и его наследникам, а также общие (public) данные и правила, которые доступны объектам и модулям в других частях программы. Важной чертой ООП является наследование, то есть возможность создавать иерархическую последовательность объектов от более общих к более специфическим объектам.

Язык C++ предоставляет гибкие и эффективные средства определения новых типов. Используя определения новых типов, отвечающих понятиям проблемной области (приложениям), программист может разбивать разрабатываемую программу на части, легко поддающиеся контролю. Новый тип создается пользователем для определения понятия, которому среди встроенных типов нет соответствия. Гораздо легче понять программу, в которой создаются типы, отвечающие понятиям проблемной области. Хорошо определенные типы делают программу ясной и короткой. Кроме того, компилятор может обнаружить недопустимое использование данных [1, 2, 3].

Такой метод построения программ называют абстракцией данных. Информация о типах содержится в объектах, тип которых определяется пользователем. Программирование с применением объектов называют объектно-ориентированным [4, 5].

Язык C++ является языком объектно-ориентированного программирования (ООП). Автор языка, Бьёрн Страуструп, создавал его с целью поддержки абстракции данных и объектно-

ориентированного программирования. Родоначальниками языков такого типа являются языки Smalltalk и Simula 67 [5].

Методические указания содержат два раздела.

В первом разделе приведены основные требования по выполнению и защите лабораторных работ по дисциплине «Объектно-ориентированное программирование в мехатронике».

Во втором разделе представлены задания для выполнения лабораторных работ.

1. Основные требования по выполнению и защите лабораторных работ

Лабораторные работы по предмету «Объектно-ориентированное программирование в мехатронике» выполняются на ЭВМ с применением интегрированных сред разработки Borland C++ Builder, Borland Developer Studio, Embarcadero CodeGear RAD Studio, Embarcadero RAD Studio или аналогичных.

Все отчеты по лабораторным работам выполняются на стандартных листах формата А4, скреплённых в тетрадь. Пример оформления титульного листа отчета по лабораторной работе приведен на рис. 1.

МИНОБРНАУКИ РОССИИ

Федеральное государственное образовательное учреждение высшего образования

«Юго-Западный государственный университет» (ЮЗГУ)

Кафедра механики, мехатроники и робототехники

Отчет по лабораторной работе №1

Вещественные типы данных

Вариант 11

Выполнил: ст. гр. МТ-41 Иванов А.С.

Проверил: к.т.н., доц. Петров П.А.

Курск 2016

Рис. 1. Пример оформления титульного листа

Отчет по лабораторной работе должен быть достаточно кратким, без лишних подробных пояснений и теоретических выводов, имеющихся в учебниках и других учебных пособиях, но содержащим:

- титульный лист;
- текст задания на выполнение лабораторной работы в соответствие с вариантом задания студента;
- листинг написанной студентом программы с комментариями;
- снимки экрана монитора с выведенными на него результатами работы написанной студентом программы;
- выводы о проделанной студентом лабораторной работе.

Отчет по лабораторной работе оформляется с применением ЭВМ в любом текстовом редакторе (Microsoft Word, Apache OpenOffice и др.), при этом желательно применение шрифтов 12 или 14 кегля. Отчет по лабораторной работе полностью печатается на принтере. После печати листы отчета скрепляются.

Изложение текстового материала отчета по лабораторной работе следует вести в безличной форме. Текст всего отчета должен быть выдержан в едином стиле.

Каждым студентом все отчеты по лабораторным работам должны выполняться и отдаваться преподавателю на проверку в сроки, предусмотренные графиком работы студентов в текущем семестре. После исправления всех ошибок, отмеченных преподавателем при проверке, результаты каждой лабораторной работы должны быть защищены студентом.

На защите студенту преподавателем задаются контрольные вопросы по соответствующему разделу курса. Если студент дал ответы на все заданные вопросы, и у преподавателя нет никаких дополнительных замечаний по отчету, то защита считается законченной.

2. Задания на выполнение лабораторных работ

2.1. Лабораторная работа №1 – Вещественные типы данных

Задание: написать с использованием интегрированной среды разработки консольное приложение, вычисляющее значение выражения при различных вещественных типах данных (float и double). Вычисления следует выполнять с использованием промежуточных переменных. Результат вычислений вывести на экран монитора, снабдив соответствующим заголовком. Сравнить и объяснить полученные результаты.

Таблица 1 – Варианты заданий

№ варианта	Задание
1	$\frac{(a+b)^2-(a^2+2ab)}{b^2}$, при a=1000, b=0.0001
2	$\frac{(a-b)^2-(a^2-2ab)}{b^2}$, при a=1000, b=0.0001
3	$\frac{(a+b)^3-(a^3+3a^2b)}{3ab^2+b^3}$, при a=1000, b=0.0001
4	$\frac{(a+b)^3-(a^3)}{3ab^2+b^3+3a^2b}$, при a=1000, b=0.0001
5	$\frac{(a-b)^3-(a^3-3a^2b)}{b^3-3ab^2}$, при a=1000, b=0.0001
6	$\frac{(a-b)^3-(a^3-3ab^2)}{b^3-3a^2b}$, при a=1000, b=0.0001
7	$\frac{(a-b)^3-(a^3)}{b^3-3ab^2-3a^2b}$, при a=1000, b=0.0001
8	$\frac{(a+b)^4-(a^4+4a^3b+6a^2b^2)}{4ab^3+b^4}$, при a=100, b=0.001
9	$\frac{(a+b)^4-(a^4+4a^3b)}{6a^2b^2+4ab^3+b^4}$, при a=100, b=0.001
10	$\frac{(a-b)^4-(a^4-4a^3b+6a^2b^2)}{b^4-4ab^3}$, при a=100, b=0.001
11	$\frac{(a-b)^4-(a^4-4a^3b)}{6a^2b^2-4ab^3+b^4}$, при a=100, b=0.001

Таблица 1 – Варианты заданий (продолжение)

№ варианта	Задание
12	$\frac{(a+b)^2-(a^2+2ab)}{b^2}$, при a=1000, b=0.0001
13	$\frac{(a-b)^2-(a^2-2ab)}{b^2}$, при a=1000, b=0.0001
14	$\frac{(a+b)^3-(a^3+3a^2b)}{3ab^2+b^3}$, при a=1000, b=0.0001
15	$\frac{(a+b)^3-(a^3)}{3ab^2+b^3+3a^2b}$, при a=1000, b=0.0001
16	$\frac{(a-b)^3-(a^3-3a^2b)}{b^3-3ab^2}$, при a=1000, b=0.0001
17	$\frac{(a-b)^3-(a^3-3ab^2)}{b^3-3a^2b}$, при a=1000, b=0.0001
18	$\frac{(a-b)^3-(a^3)}{b^3-3ab^2-3a^2b}$, при a=1000, b=0.0001
19	$\frac{(a+b)^4-(a^4+4a^3b+6a^2b^2)}{4ab^3+b^4}$, при a=100, b=0.001
20	$\frac{(a+b)^4-(a^4+4a^3b)}{6a^2b^2+4ab^3+b^4}$, при a=100, b=0.001
21	$\frac{(a-b)^4-(a^4-4a^3b+6a^2b^2)}{b^4-4ab^3}$, при a=100, b=0.001
22	$\frac{(a-b)^4-(a^4-4a^3b)}{6a^2b^2-4ab^3+b^4}$, при a=100, b=0.001
23	$\frac{(a+b)^3-(a^3+3a^2b)}{3ab^2+b^3}$, при a=1000, b=0.0001
24	$\frac{(a+b)^3-(a^3)}{3ab^2+b^3+3a^2b}$, при a=1000, b=0.0001
25	$\frac{(a-b)^3-(a^3-3a^2b)}{b^3-3ab^2}$, при a=1000, b=0.0001

Методические указания

Обмен данными посредством консольного ввода-вывода программа на языке Си реализует с помощью библиотеки функций ввода-вывода stdio – к проекту подключается заголовочный файл библиотеки stdio.h:

```
#include <stdio.h>
```

Функция, выполняющая вывод на экран монитора:

printf (<форматная строка>, <список аргументов>);

<форматная строка> - строка символов, заключенных в кавычки, которая показывает, как должны быть напечатаны аргументы.

Например:

printf (“Значение числа Пи равно %f\n”, pi);

Форматная строка может содержать:

- символы печатаемые текстуально;
- спецификации преобразования;
- управляющие символы (\n – управляющий символ перехода на новую строку; \a – звуковой сигнал; \t – горизонтальная табуляция и др.).

Каждому аргументу функции printf соответствует своя спецификация преобразования, указываемая в форматной строке:

%d, %i – десятичное целое число;
%f – число с плавающей запятой;
%e, %E – число с плавающей запятой в экспоненциальной форме;
%u – десятичное число в беззнаковой форме;
%c – символ;
%s – строка.

Функция, выполняющая ввод с клавиатуры:

scanf (<форматная строка>, <список аргументов>);

К аргументам функции scanf применяется операция взятия адреса (оператор &):

scanf(“%d%f ”, &x, &y);

Для ввода и вывода данных также можно использовать операции >> и << и стандартные потоки cin и cout (к проекту подключается заголовочный файл библиотеки iostream.h).

Пример использования стандартных потоков cin (для ввода посредством клавиатуры) и cout (для вывода на экран монитора):

```
#include <iostream.h>;
int n = 0;
    …
cout << “\nВведите количество элементов: ”;
cin >> n;
```

Для вычисления степени можно использовать функцию pow(x, y) из заголовочного файла библиотеки math.h.

При выполнении задания надо использовать вспомогательные переменные для хранения промежуточных результатов. Например: c = pow(a, 3); d = 3*a*a*b; e = 3*a*b*b; f = pow(b, 3).

2.2. Лабораторная работа №2 – Циклы

Задание: написать с применением интегрированной среды разработки консольное приложение, позволяющее с использованием оператора цикла найти сумму 10 членов ряда, общий член которого указан в конкретном варианте. Результат вычислений вывести на экран монитора, снабдив соответствующим заголовком.

Варианты заданий:

1) $a_n = \frac{n^3}{n!}$.

2) $a_n = \frac{n}{(n-1)^4}$.

3) $a_n = e \cdot 100^{-n}$.

4) $a_n = \frac{\lg(n!)}{n^2}$.

5) $a_n = \frac{(-1)^{n-1}}{n^n}$.

6) $a_n = \frac{1}{2^n} + \frac{1}{3^n}$.

7) $a_n = \frac{1}{((3n-2)(3n+1))}$.

8) $a_n = \frac{(2n-1)}{2^n}$.

9) $a_n = \frac{10^n}{n!}$.

10) $a_n = \frac{n!}{2^n}$.

11) $a_n = \frac{n!}{n^n}$.

12) $a_n = \frac{2^n n}{n^n}$.

13) $a_n = \frac{3^n n}{n!}$.

14) $a_n = \frac{n!}{3n^n}$.

15) $a_n = \dfrac{n^2}{2^{n^2}}$.

16) $a_n = \lg(n!)e^{-n\sqrt{n}}$.

17) $a_n = 10^{-n} \cdot n!$.

18) $a_n = \dfrac{n^n}{n!}$.

19) $a_n = \dfrac{n}{(n-1)^2}$.

20) $a_n = 100^{-n^2}$.

21) $a_n = \dfrac{\ln(n!)}{n^2}$.

22) $a_n = \dfrac{n^{\ln n}}{(\ln n)^n}$.

23) $a_n = \dfrac{n!}{n^{\sqrt{n}}}$.

24) $a_n = e^{-\sqrt{n}}$.

25) $a_n = n^2 e^{-\sqrt{n}}$.

Методические указания

Реализация повторяющихся процессов может быть достигнута использованием:

– цикла с предусловием:

while (выражение-условие) <оператор>;

В качестве <выражения-условия> чаще всего используется отношение или логическое выражение. Если оно истинно, т. е. не равно 0, то тело цикла (<оператор>) выполняется до тех пор, пока выражение-условие не станет ложным:

```
while (a != 0)
{
    cin >> a;
    s += a;
}
```

– цикла с постусловием:

do <оператор> while (выражение-условие);

Тело цикла (<оператор>) выполняется до тех пор, пока выражение-условие истинно:

```
do
{
    cin >> a;
    s += a;
}
while(a != 0);
```

– цикла с параметром:

for (выражение_1; выражение-условие; выражение_3) <оператор>;

Выражение_1 и выражение_3 могут состоять из нескольких выражений, разделенных запятыми. Выражение_1 – задает начальные условия для цикла (инициализация). Выражение-условие определяет условие выполнения цикла, если оно не равно 0, цикл выполняется, а затем вычисляется значение выражения_3. Выражение_3 – задает изменение параметра цикла или других переменных (коррекция). Цикл продолжается до тех пор, пока выражение-условие не станет равно 0. Любое выражение может отсутствовать, но разделяющие их символы « ; » должны быть обязательно.

2.3. Лабораторная работа №3 – Одномерные массивы

Задание: написать с применением интегрированной среды разработки консольное приложение, выполняющее в соответствии с номером варианта задания последовательность операций над одномерным массивом целочисленных элементов.

Варианты заданий:

1.
- Сформировать одномерный массив целых чисел, используя генератор случайных чисел. Длина массива задается пользователем.
- Вывести на экран монитора полученный массив.
- Удалить элемент с заданным значением (вводится пользователем).
- Добавить после каждого четного элемента массива элемент со значением 0.
- Вывести на экран монитора полученный массив.

2.
- Сформировать одномерный массив целых чисел, используя генератор случайных чисел. Длина массива задается пользователем.

- Вывести на экран монитора полученный массив.
- Удалить первый элемент равный 0.
- Добавить после каждого четного элемента массива элемент со значением *M*[*I*+1]+2, где *I* – индекс четного элемента в массиве *M*.
- Вывести на экран монитора полученный массив.

3.

- Сформировать одномерный массив целых чисел, используя генератор случайных чисел. Длина массива задается пользователем.
- Вывести на экран монитора полученный массив.
- Удалить все элементы равные 0.
- Добавить после первого четного элемента массива элемент со значением *M*[*I*+1]+2, где *I* – индекс элемента в массиве *M*.
- Вывести на экран монитора полученный массив.

4.

- Сформировать одномерный массив целых чисел, используя генератор случайных чисел. Длина массива задается пользователем.
- Вывести на экран монитора полученный массив.
- Удалить элементы, индексы которых кратны 3.
- Добавить после каждого отрицательного элемента массива элемент со значением | *M*[*I*+1]+1 |, где *I* – индекс отрицательного элемента в массиве *M*.
- Вывести на экран монитора полученный массив.

5.

- Сформировать одномерный массив целых чисел, используя генератор случайных чисел. Длина массива задается пользователем.
- Вывести на экран монитора полученный массив.
- Удалить элементы кратные 7.
- Добавить после каждого нечетного элемента массива элемент со значением 0.
- Вывести на экран монитора полученный массив.

6.

- Сформировать одномерный массив целых чисел, используя генератор случайных чисел. Длина массива задается пользователем.
- Вывести на экран монитора полученный массив.
- Удалить элемент с заданным номером (вводится пользователем).
- Добавить после первого четного элемента массива элемент со значением *M*[*I*+1]+2, где *I* – индекс первого четного элемента в массиве *M*.
- Вывести на экран монитора полученный массив.

7.

- Сформировать одномерный массив целых чисел, используя генератор случайных чисел. Длина массива задается пользователем.
- Вывести на экран монитора полученный массив.
- Удалить последний элемент равный 0.
- Добавить после элемента массива с заданным индексом (вводится пользователем) элемент со значением 100.
- Вывести на экран монитора полученный массив.

8.

- Сформировать одномерный массив целых чисел, используя генератор случайных чисел. Длина массива задается пользователем.
- Вывести на экран монитора полученный массив.
- Удалить все элементы с заданным значением (вводится пользователем).
- Добавить перед каждым четным элементом массива элемент со значением 0.
- Вывести на экран монитора полученный массив.

9.

- Сформировать одномерный массив целых чисел, используя генератор случайных чисел. Длина массива задается пользователем.
- Вывести на экран монитора полученный массив.
- Удалить первый элемент с заданным значением.
- Сдвинуть массив на *N* (вводится пользователем) элементов вправо.
- Вывести на экран монитора полученный массив.

10.

- Сформировать одномерный массив целых чисел, используя генератор случайных чисел. Длина массива задается пользователем.
- Вывести на экран монитора полученный массив.
- Удалить 5 первых элементы массива.
- Добавить в конец массива 3 новых элемента.
- Вывести на экран монитора полученный массив.

11.

- Сформировать одномерный массив целых чисел, используя генератор случайных чисел. Длина массива задается пользователем.
- Вывести на экран монитора полученный массив.
- Удалить 5 последних элементов массива.
- Добавить в начало массива 3 элемента со значением *M*[*I*+1]+2, где *I* – индекс одного из трех начальных элементов в массиве *M*.
- Вывести на экран монитора полученный массив.

12.

- Сформировать одномерный массив целых чисел, используя генератор случайных чисел. Длина массива задается пользователем.
- Вывести на экран монитора полученный массив.

- Поменять местами минимальный и максимальный элементы массива.
- Удалить из массива все элементы, превышающие его среднее значение более, чем на 10%.
- Вывести на экран монитора полученный массив.

13.

- Сформировать одномерный массив целых чисел, используя генератор случайных чисел. Длина массива задается пользователем.
- Вывести на экран монитора полученный массив.
- Удалить из массива все элементы, совпадающие с его минимальным значением.
- Добавить в начало массива 3 элемента со значением равным среднему арифметическому массива.
- Вывести на экран монитора полученный массив.

14.

- Сформировать одномерный массив целых чисел, используя генератор случайных чисел. Длина массива задается пользователем.
- Вывести на экран монитора полученный массив.
- Перевернуть массив.
- Добавить в начало массива 4 элемента со значением *M*[*I*+1]-1, где *I* – индекс одного из четырех начальных элементов в массиве *M*.
- Вывести на экран монитора полученный массив.

15.

- Сформировать одномерный массив целых чисел, используя генератор случайных чисел. Длина массива задается пользователем.
- Вывести на экран монитора полученный массив.
- Перевернуть массив.

– Добавить после каждого отрицательного элемента массива элемент со значением | *M*[*I*+1]+1 |, где *I* – индекс отрицательного элемента в массиве *M*.
– Вывести на экран монитора полученный массив.

16.

– Сформировать одномерный массив целых чисел, используя генератор случайных чисел. Длина массива задается пользователем.
– Вывести на экран монитора полученный массив.
– Удалить из массива все элементы, совпадающие с его минимальным значением.
– Добавить после каждого нечетного элемента массива элемент со значением 0.
– Вывести на экран монитора полученный массив.

17.

– Сформировать одномерный массив целых чисел, используя генератор случайных чисел. Длина массива задается пользователем.
– Вывести на экран монитора полученный массив.
– Удалить все элементы больше заданного значения (вводится пользователем).
– Добавить после каждого отрицательного элемента массива элемент со значением | *M*[*I*+1]+1 |, где *I* – индекс отрицательного элемента в массиве *M*.
– Вывести на экран монитора полученный массив.

18.

– Сформировать одномерный массив целых чисел, используя генератор случайных чисел. Длина массива задается пользователем.
– Вывести на экран монитора полученный массив.
– Удалить все элементы меньше заданного значения (вводится пользователем).

– Добавить после каждого четного элемента массива элемент со значением 0.
– Вывести на экран монитора полученный массив.

19.

– Сформировать одномерный массив целых чисел, используя генератор случайных чисел. Длина массива задается пользователем.
– Вывести на экран монитора полученный массив.
– Удалить первый элемент равный 0.
– Добавить после каждого четного элемента массива элемент со значением 0.
– Вывести на экран монитора полученный массив.

20.

– Сформировать одномерный массив целых чисел, используя генератор случайных чисел. Длина массива задается пользователем.
– Вывести на экран монитора полученный массив.
– Удалить элементы, значения которых меньше 10.
– Добавить после каждого отрицательного элемента массива элемент со значением | *M*[*I*+1]+5 |, где *I* – индекс отрицательного элемента в массиве *M*.
– Вывести на экран монитора полученный массив.

Методические указания

При выполнении работы используются статические массивы. Для организации статических массивов с псевдопеременными границами необходимо объявить массив достаточно большой длины, например, 100 элементов:

```
int L = 100;
int M[L];
```

Затем пользователь вводит реальную длину массива (не

больше L) и работает с массивом той длины, которую он сам указал. Неиспользуемые элементы (хотя память под них и будет выделена) не рассматриваются.

Генератор случайных чисел вызывается функцией (к проекту подключается заголовочный файл библиотеки stdlib.h) и возвращает случайное целое число в диапазоне от 0 до константы RAND_MAX, равной 32767:

```
int i = rand();
```

2.4. Лабораторная работа №4 – Сортировка массива

Задание: написать с применением интегрированной среды разработки консольное приложение, выполняющее в одномерном массиве, состоящем из *n* вещественных элементов, операции в соответствии с номером варианта задания.

Варианты заданий:

1.
- вычислить сумму отрицательных элементов массива;
- вычислить произведение элементов массива, расположенных между максимальным и минимальным элементами;
- упорядочить элементы массива по возрастанию.

2.
- вычислить сумму положительных элементов массива;
- вычислить произведение элементов массива, расположенных между максимальным по модулю и минимальным по модулю элементами;
- упорядочить элементы массива по убыванию.

3.
- вычислить произведение элементов массива с четными индексами;

– вычислить сумму элементов массива, расположенных между первым и последним нулевыми элементами;
– преобразовать массив таким образом, чтобы сначала располагались все положительные элементы, а потом – все отрицательные (элементы, равные 0, считать положительными).

4.

– вычислить сумму элементов массива с нечетными индексами;
– вычислить сумму элементов массива, расположенных между первым и последним отрицательными элементами;
– сжать массив, удалив из него все элементы, модуль которых не превышает 1. Освободившиеся в конце массива элементы заполнить нулями.

5.

– определить максимальный элемент массива;
– вычислить сумму элементов массива, расположенных до последнего положительного элемента;
– сжать массив, удалив из него все элементы, модуль которых находится в интервале $[a, b]$ (интервал задается пользователем). Освободившиеся в конце массива элементы заполнить нулями.

6.

– определить минимальный элемент массива;
– вычислить сумму элементов массива, расположенных между первым и последним положительными элементами;
– преобразовать массив таким образом, чтобы сначала располагались все элементы, равные нулю, а потом – все остальные.

7.

– определить индекс максимального элемента массива;
– вычислить произведение элементов массива, расположенных между первым и вторым нулевыми элементами;

– преобразовать массив таким образом, чтобы в первой его половине располагались элементы, стоявшие в нечетных позициях, а во второй половине – элементы, стоявшие в четных позициях.

8.

– определить индекс минимального элемента массива;
– вычислить сумму элементов массива, расположенных между первым и вторым отрицательными элементами;
– преобразовать массив таким образом, чтобы сначала располагались все элементы, модуль которых не превышает 1, а потом – все остальные.

9.

– определить максимальный по модулю элемент массива;
– вычислить сумму элементов массива, расположенных между первым и вторым положительными элементами;
– преобразовать массив таким образом, чтобы элементы, равные нулю, располагались после всех остальных.

10.

– определить минимальный по модулю элемент массива;
– вычислить сумму модулей элементов массива, расположенных после первого элемента, равного нулю;
– преобразовать массив таким образом, чтобы в первой его половине располагались элементы, стоявшие в четных позициях, а во второй половине – элементы, стоявшие в нечетных позициях.

11.

– определить индекс минимального по модулю элемента массива;
– вычислить сумму модулей элементов массива, расположенных после первого отрицательного элемента;
– сжать массив, удалив из него все элементы, величина которых находится в интервале [a, b] (интервал задается

пользователем). Освободившиеся в конце массива элементы заполнить нулями.

12.

- определить индекс максимального по модулю элемента массива;
- вычислить сумму элементов массива, расположенных после первого положительного элемента;
- преобразовать массив таким образом, чтобы сначала располагались все элементы, целая часть которых лежит в интервале [a, b] (интервал задается пользователем), а потом – все остальные.

13.

- определить количество элементов массива, лежащих в интервале [a, b] (интервал задается пользователем);
- вычислить сумму элементов массива, расположенных после максимального элемента;
- упорядочить элементы массива по убыванию модулей элементов.

14.

- определить количество элементов массива, равных 0;
- вычислить сумму элементов массива, расположенных после минимального элемента;
- упорядочить элементы массива по возрастанию модулей элементов.

15.

- определить количество элементов массива, больших C (C задается пользователем);
- вычислить произведение элементов массива, расположенных после максимального по модулю элемента;
- преобразовать массив таким образом, чтобы сначала располагались все отрицательные элементы, а потом – все

положительные (элементы, равные 0, считать положительными).

16.

- определить количество отрицательных элементов массива;
- вычислить сумму модулей элементов массива, расположенных после минимального по модулю элемента;
- заменить все отрицательные элементы массива их квадратами и упорядочить элементы массива по возрастанию.

17.

- определить количество положительных элементов массива;
- вычислить сумму элементов массива, расположенных после последнего элемента, равного нулю;
- преобразовать массив таким образом, чтобы сначала располагались все элементы, целая часть которых не превышает 1, а потом – все остальные.

18.

- определить количество элементов массива, меньших *C* (*C* задается пользователем);
- вычислить сумму целых частей элементов массива, расположенных после последнего отрицательного элемента;
- преобразовать массив таким образом, чтобы сначала располагались все элементы, отличающиеся от максимального не более чем на 20%, а потом – все остальные.

19.

- вычислить произведение отрицательных элементов массива;
- вычислить сумму положительных элементов массива, расположенных до максимального элемента.
- изменить порядок следования элементов в массиве на обратный.

20.

- вычислить произведение положительных элементов массива;

– вычислить сумму элементов массива, расположенных до минимального элемента;
– упорядочить по возрастанию отдельно элементы, стоящие на четных местах, и элементы, стоящие на нечетных местах.

Методические указания

При выполнении работы используются динамические одномерные массивы. Длина (количество элементов) массива и значения его элементов задаются пользователем. Тип элемента массива – float.

Пример выделения памяти под динамический массив mas с пятью элементами (с индексами от 0 до 4 включительно) типа float с использованием операции new:

```
float *mas = new float[5];
```

Пример освобождения памяти от динамического массива mas с использованием операции delete:

```
delete [] mas;
```

2.5. Лабораторная работа №5 – Структуры

Задание: написать с применением интегрированной среды разработки консольное приложение, реализующее работу базы данных на основе одномерного массива структур.

Варианты заданий:

1.
– Описать структуру с именем STUDENT, содержащую следующие поля:
    ■ фамилия и инициалы;
    ■ номер группы;

- успеваемость (массив из пяти элементов).

– Написать программу, выполняющую следующие действия:
  - ввод с клавиатуры данных в массив, состоящий из десяти структур типа STUDENT;
  - записи должны быть упорядочены по возрастанию номера группы;
  - вывод на дисплей фамилий и номеров групп для всех студентов, включенных в массив, если средний балл студента больше 4,0;
  - если таких студентов нет, вывести соответствующее сообщение.

2.

– Описать структуру с именем STUDENT, содержащую следующие поля:
  - фамилия и инициалы;
  - номер группы;
  - успеваемость (массив из пяти элементов).

– Написать программу, выполняющую следующие действия:
  - ввод с клавиатуры данных в массив, состоящий из десяти структур типа STUDENT;
  - записи должны быть упорядочены по возрастанию среднего балла;
  - вывод на дисплей фамилий и номеров групп для всех студентов, имеющих оценки 4 и 5;
  - если таких студентов нет, вывести соответствующее сообщение.

3.

– Описать структуру с именем STUDENT, содержащую следующие поля:
  - фамилия и инициалы;
  - номер группы;
  - успеваемость (массив из пяти элементов).

– Написать программу, выполняющую следующие действия:

- ввод с клавиатуры данных в массив, состоящий из десяти структур типа STUDENT;
- записи должны быть упорядочены по алфавиту;
- вывод на дисплей фамилий и номеров групп для всех студентов, имеющих хотя бы одну оценку 2;
- если таких студентов нет, вывести соответствующее сообщение.

4.

– Описать структуру с именем AEROFLOT, содержащую следующие поля:
  - название пункта назначения рейса;
  - номер рейса;
  - тип самолета.
– Написать программу, выполняющую следующие действия:
  - ввод с клавиатуры данных в массив, состоящий из семи элементов типа AEROFLOT;
  - записи должны быть упорядочены по возрастанию номера рейса;
  - вывод на экран номеров рейсов и типов самолетов, вылетающих в пункт назначения, название которого совпало с названием, введенным с клавиатуры;
  - если таких рейсов нет, выдать на дисплей соответствующее сообщение.

5.

– Описать структуру с именем AEROFLOT, содержащую следующие поля:
  - название пункта назначения рейса;
  - номер рейса;
  - тип самолета.
– Написать программу, выполняющую следующие действия:
  - ввод с клавиатуры данных в массив, состоящий из семи элементов тина AEROFLOT;
  - записи должны быть размещены в алфавитном порядке по названиям пунктов назначения;

- вывод на экран пунктов назначения и номеров рейсов, обслуживаемых самолетом, тип которого введен с клавиатуры;
- если таких рейсов нет, выдать на дисплей соответствующее сообщение.

6.

- Описать структуру с именем WORKER, содержащую следующие поля:
  - фамилия и инициалы работника;
  - название занимаемой должности;
  - год поступления на работу.
- Написать программу, выполняющую следующие действия:
  - ввод с клавиатуры данных в массив, состоящий из десяти структур типа WORKER;
  - записи должны быть размещены по алфавиту;
  - вывод на дисплей фамилий работников, чей стаж работы в организации превышает значение, введенное с клавиатуры;
  - если таких работников нет, вывести на дисплей соответствующее сообщение.

7.

- Описать структуру с именем TRAIN, содержащую следующие поля:
  - название пункта назначения;
  - номер поезда;
  - время отправления.
- Написать программу, выполняющую следующие действия:
  - ввод с клавиатуры данных в массив, состоящий из восьми элементов типа TRAIN;
  - записи должны быть размещены в алфавитном порядке по названиям пунктов назначения;
  - вывод на экран информации о поездах, отправляющихся после введённого с клавиатуры времени;

- если таких поездов нет, выдать на дисплей соответствующее сообщение.

8.

- Описать структуру с именем TRAIN, содержащую следующие поля:
  - название пункта назначения;
  - номер поезда;
  - время отправления.
- Написать программу, выполняющую следующие действия:
  - ввод с клавиатуры данных в массив, состоящий из шести элементов типа TRAIN;
  - записи должны быть упорядочены по времени отправления поезда;
  - вывод на экран информации о поездах, направляющихся в пункт, название которого введено с клавиатуры;
  - если таких поездов нет, выдать на дисплей соответствующее сообщение.

9.

- Описать структуру с именем TRAIN, содержащую следующие поля:
  - название пункта назначения;
  - номер поезда;
  - время отправления.
- Написать программу, выполняющую следующие действия:
  - ввод с клавиатуры данных в массив, состоящий из восьми элементов типа TRAIN;
  - записи должны быть упорядочены по номерам поездов;
  - вывод на экран информации о поезде, номер которого введен с клавиатуры;
  - если таких поездов нет, выдать на дисплей соответствующее сообщение.

10.

- Описать структуру с именем MARSH, содержащую следующие поля:
  - название начального пункта маршрута;
  - название конечного пункта маршрута;
  - номер маршрута.
- Написать программу, выполняющую следующие действия:
  - ввод с клавиатуры данных в массив, состоящий из восьми элементов типа MARSH;
  - записи должны быть упорядочены по номерам маршрутов;
  - вывод на экран информации о маршруте, номер которого введен с клавиатуры;
  - если таких маршрутов нет, выдать на дисплей соответствующее сообщение.

11.

- Описать структуру с именем MARSH, содержащую следующие поля:
  - название начального пункта маршрута;
  - название конечного пункта маршрута;
  - номер маршрута.
- Написать программу, выполняющую следующие действия:
  - ввод с клавиатуры данных в массив, состоящий из восьми элементов типа MARSH;
  - записи должны быть упорядочены по номерам маршрутов;
  - вывод на экран информации о маршрутах, которые начинаются или кончаются в пункте, название которого введено с клавиатуры;
  - если таких маршрутов нет, выдать па дисплей соответствующее сообщение.

12.

- Описать структуру с именем NOTE, содержащую следующие поля:
  - фамилия, имя;

- номер телефона;
- день рождения (массив из трех чисел).

– Написать программу, выполняющую следующие действия:
  - ввод с клавиатуры данных в массив, состоящий из восьми элементов типа NOTE;
  - записи должны быть упорядочены по датам дней рождения;
  - вывод па экран информации о человеке, номер телефона которого введен с клавиатуры;
  - если такого нет, выдать на дисплей соответствующее сообщение.

13.

– Описать структуру с именем NOTE, содержащую следующие поля:
  - фамилия, имя;
  - номер телефона;
  - день рождения (массив из трех чисел).
– Написать программу, выполняющую следующие действия:
  - ввод с клавиатуры данных в массив, состоящий из восьми элементов типа NOTE;
  - записи должны быть размещены по алфавиту;
  - вывод па экран информации о людях, чьи дни рождения приходятся на месяц, значение которого введено с клавиатуры;
  - если таких нет, выдать на дисплей соответствующее сообщение.

14.

– Описать структуру с именем NOTE, содержащую следующие поля:
  - фамилия, имя;
  - номер телефона;
  - день рождения (массив из трех чисел).
– Написать программу, выполняющую следующие действия:
  - ввод с клавиатуры данных в массив, состоящий из восьми элементов типа NOTE;

- записи должны быть упорядочены по трем первым цифрам номера телефона;
- вывод на экран информации о человеке, чья фамилия введена с клавиатуры;
- если такого нет, выдать на дисплей соответствующее сообщение.

15.

- Описать структуру с именем ZNAK, содержащую следующие поля:
  - фамилия, имя;
  - знак Зодиака;
  - день рождения (массив из трех чисел).
- Написать программу, выполняющую следующие действия:
  - ввод с клавиатуры данных в массив, состоящий из восьми элементов тина ZNAK;
  - записи должны быть упорядочены по датам дней рождения;
  - вывод на экран информации о человеке, чья фамилия введена с клавиатуры;
  - если такого нет, выдать на дисплей соответствующее сообщение.

16.

- Описать структуру с именем ZNAK, содержащую следующие поля:
  - фамилия, имя;
  - знак Зодиака;
  - день рождения (массив из трех чисел).
- Написать программу, выполняющую следующие действия:
  - ввод с клавиатуры данных в массив, состоящий из восьми элементов типа ZNAK;
  - записи должны быть упорядочены по датам дней рождения;
  - вывод на экран информации о людях, родившихся под знаком, наименование которого введено с клавиатуры;

- если таких нет, выдать на дисплей соответствующее сообщение.

17.

- Описать структуру с именем ZNAK, содержащую следующие поля:
  - фамилия, имя;
  - знак Зодиака;
  - день рождения (массив из трех чисел).
- Написать программу, выполняющую следующие действия:
  - ввод с клавиатуры данных в массив, состоящий из восьми элементов типа ZNAK;
  - записи должны быть упорядочены по знакам Зодиака;
  - вывод на экран информации о людях, родившихся в месяц, значение которого введено с клавиатуры;
  - если таких нет, выдать на дисплей соответствующее сообщение.

18.

- Описать структуру с именем PRICE, содержащую следующие поля:
  - название товара;
  - название магазина, в котором продается товар;
  - стоимость товара в рублях.
- Написать программу, выполняющую следующие действия:
  - ввод с клавиатуры данных в массив, состоящий из восьми элементов типа PRICE;
  - записи должны быть размещены в алфавитном порядке по названиям товаров;
  - вывод на экран информации о товаре, название которого введено с клавиатуры;
  - если таких товаров нет, выдать на дисплей соответствующее сообщение.

19.

- Описать структуру с именем PRICE, содержащую следующие поля:

- название товара;
- название магазина, в котором продается товар;
- стоимость товара в руб.

– Написать программу, выполняющую следующие действия:
  - ввод с клавиатуры данных в массив, состоящий из восьми элементов типа PRICE;
  - записи должны быть размещены в алфавитном порядке по названиям магазинов;
  - вывод на экран информации о товарах, продающихся в магазине, название которого введено с клавиатуры;
  - если такого магазина нет, выдать на дисплей соответствующее сообщение.

20.

– Описать структуру с именем ORDER, содержащую следующие поля:
  - расчетный счет плательщика;
  - расчетный счет получателя;
  - перечисляемая сумма в руб.
– Написать программу, выполняющую следующие действия:
  - ввод с клавиатуры данных в массив, состоящий из восьми элементов типа ORDER;
  - записи должны быть размещены в алфавитном порядке по расчетным счетам плательщиков;
  - вывод на экран информации о сумме, снятой с расчетного счета плательщика, введенного с клавиатуры;
  - если такого расчетного счета нет, выдать на дисплей соответствующее сообщение.

Методические указания

При выполнении работы могут использоваться как статические, так и динамические одномерные массивы структур.

Пример описания структуры, содержащей два поля:

```
struct MyStruct
{
        int A;
        char C[10];
};
```

Пример описания статического массива M, содержащего 10 элементов типа структуры MyStruct:

```
MyStruct M[10];
```

Пример использования динамического массива M, содержащего N элементов типа структуры MyStruct:

```
MyStruct *M;
    …
int N;
cin >> N;
    …
M = new MyStruct[N];
    …
delete [] M;
```

2.6. Лабораторная работа №6 – Строки и файлы

Задание: с помощью текстового редактора создать файл, содержащий текст, длина которого не превышает 1000 символов (длина строки текста не должна превышать 70 символов). Имя файла должно иметь расширение DAT. Слова в тексте разделены одним или несколькими пробелами. В конце каждого предложения текста ставится точка.

Написать с применением интегрированной среды разработки консольное приложение, которое в соответствии с номером варианта задания выполняет работу с созданным текстовым

файлом.

Варианты заданий:

1.
Написать программу, которая:
- выводит текст файла на экран монитора;
- определяет количество предложений в тексте.

2.
Написать программу, которая:
- выводит текст файла на экран монитора;
- определяет количество слов в тексте.

3.
Написать программу, которая:
- выводит текст файла на экран монитора;
- определяет количество слов в тексте, оканчивающихся на гласную букву.

4.
Написать программу, которая:
- выводит текст файла на экран монитора;
- выводит текст на экран монитора еще раз, располагая слова текста в обратном порядке и удаляя лишние пробелы.

5.
Написать программу, которая:
- выводит текст файла на экран монитора;
- определяет количество слов в тексте, у которых первый и последний символы совпадают.

6.
Написать программу, которая:
- выводит текст файла на экран монитора;
- определяет количество слов в тексте, начинающихся на

гласную букву.

7.

Написать программу, которая:

- выводит текст файла на экран монитора;
- определяет количество символов в самом длинном слове.

8.

Написать программу, которая:

- выводит текст файла на экран монитора;
- определяет количество символов в самом коротком слове.

9.

Написать программу, которая:

- выводит текст файла на экран монитора;
- определяет в каждом предложении текста количество символов, отличных от букв и пробела.

10.

Написать программу, которая:

- выводит текст файла на экран монитора;
- определяет количество предложений текста и количество слов в каждом предложении.

11.

Написать программу, которая:

- выводит текст файла на экран монитора;
- определяет количество букв 'a' в последнем слове текста.

12.

Написать программу, которая:

- выводит текст файла на экран монитора;
- определяет самую длинную последовательность цифр в тексте (считать, что любое количество пробелов между двумя цифрами не прерывает последовательности цифр).

13.

Написать программу, которая:

- выводит текст файла на экран монитора;
- определяет порядковый номер заданного слова в каждом предложении текста (заданное слово вводится пользователем).

14.

Написать программу, которая:

- выводит текст файла на экран монитора;
- выводит текст на экран монитора еще раз, выкидывая из него заданное слово (заданное слово вводится пользователем) и удаляя лишние пробелы.

15.

Написать программу, которая:

- выводит текст файла на экран монитора;
- выводит текст на экран монитора еще раз, меняя в нем местами заданные слова (заданные слова вводятся пользователем) и удаляя лишние пробелы.

16.

Написать программу, которая:

- выводит текст файла на экран монитора;
- выводит текст на экран монитора еще раз, заключая заданное слово (заданное слово вводится пользователем) в кавычки.

17.

Написать программу, которая:

- выводит текст файла на экран монитора;
- выводит текст на экран монитора еще раз, вставляя в каждое предложение в качестве последнего заданное слово (заданное слово вводится пользователем).

18.

Написать программу, которая:

- выводит текст файла на экран монитора;
- выводит текст на экран монитора еще раз, убирая лишние пробелы между словами и начиная каждое предложение с новой строки.

19.

Написать программу, которая:

- выводит текст файла на экран монитора;
- выводит текст на экран монитора еще раз, заменяя в заданном слове (заданное слово вводится пользователем) строчные буквы прописными.

20.

Написать программу, которая:

- выводит текст файла на экран монитора;
- определяет наибольшее количество подряд идущих пробелов в тексте.

Методические указания

Функции библиотеки для организации потокового чтения и записи файлов в стиле языка С находятся в библиотеке stdio (к проекту подключается заголовочный файл библиотеки stdio.h).

Прежде чем начать работать с потоком, его надо инициировать, т. е. открыть. При этом поток связывается со структурой предопределенного типа FILE, определение которой находится в библиотечном файле <stdio.h>. В структуре находится указатель на буфер, указатель на текущую позицию файла и т. п. При открытии потока, возвращается указатель на поток, т. е. на объект типа FILE:

```
#include <stdio.h>;
…
FILE *fp;
…
```

fp = fopen(“t.txt”, “r”);

где fopen(<имя_файла>, <режим_открытия>) - функция для инициализации файла.

Режимы открытия файла приведены в таблице 2.

Таблица 2 – Режимы открытия файла

Режим	Описание режима открытия файла
r	Файл открывается для чтения, если файл не существует, то выдается ошибка при исполнении программы.
w	Файл открывается для записи, если файл не существует, то он будет создан, если файл уже существует, то вся информация из него стирается.
a	Файл открывается для добавления, если файл не существует, то он будет создан, если существует, то информация из него не стирается и можно выполнять запись в конец файла.
r+	Файл открывается для чтения и записи, изменить размер файла нельзя, если файл не существует, то выдается ошибка при исполнении программы.
w+	Файл открывается для чтения и записи, если файл не существует, то он будет создан, если файл уже существует, то вся информация из него стирается.
a+	Файл открывается для чтения и записи, если файл не существует, то он будет создан, если существует, то информация из него не стирается, можно выполнять запись в конец файла.

Поток можно открыть в текстовом (t) или двоичном (b) режиме. По умолчанию используется текстовый режим. В явном виде режим указывается следующим образом:

– “r+b”или “rb” – двоичный (бинарный) режим;
– “r+t” или “rt” – текстовый режим.

В файле stdio.h определена константа EOF, которая сообщает об окончании файла (отрицательное целое число).

При открытии потока могут возникать следующие ошибки:

– файл, связанный с потоком не найден (при чтении из файла);
– диск заполнен (при записи);
– диск защищен от записи (при записи) и т. п.

В этих случаях указатель на поток приобретет значение NULL (0). Указатель на поток, отличный от аварийного, не равен 0.

После работы с файлом, его надо закрыть:

fclose(<указатель_на_поток>);

Для символьного ввода-вывода используются функции:

– int fgetc(FILE *fp), где fp – указатель на поток, из которого выполняется считывание. Функция возвращает очередной символ в форме int из потока fp. Если символ не может быть прочитан, то возвращается значение EOF.

– int fputc(int c, FILE *fp), где fp – указатель на поток, в который выполняется запись, c – переменная типа int, в которой содержится записываемый в поток символ. Функция возвращает записанный в поток fp символ в форме int. Если символ не может быть записан, то возвращается значение EOF.

Для построчного ввода-вывода используются следующие функции:

– char* fgets(char *s, int n, FILE *f), где char *s – адрес, по которому размещаются считанные байты, int n – количество считанных байтов, FILE *f – указатель на файл, из которого производится считывание.

Прием байтов заканчивается после передачи n-1 байтов или при получении управляющего символа ‘\n’. Управляющий символ тоже передается в принимающую строку. Строка в любом случае заканчивается ‘\0’. При успешном завершении работы функция возвращает указатель на прочитанную строку, при неуспешном – 0.

– int fputs(char *s, FILE *f), где char *s – адрес, из которого берутся записываемые в файл байты, FILE *f – указатель на файл, в который производится запись.

Символ конца строки (‘\0’) в файл не записывается. Функция возвращает EOF, если при записи в файл произошла ошибка, при успешной записи возвращает неотрицательное число.

Для блокового ввода-вывода используются функции:

– int fread(void *ptr, int size, int n, FILE *f), где void *ptr – указатель на область памяти, в которой размещаются считанные из файла данные, int size – размер одного считываемого элемента, int n – количество считываемых элементов, FILE *f – указатель на файл, из которого производится считывание.

В случае успешного считывания функция возвращает количество считанных элементов, иначе – EOF.

– int fwrite(void *ptr, int size, int n, FILE *f), где void *ptr – указатель на область памяти, в которой размещаются считанные из файла данные, int size – размер одного записываемого элемента, int n – количество записываемых элементов, FILE *f – указатель на файл, в который производится запись.

В случае успешной записи функция возвращает количество записанных элементов, иначе – EOF.

В некоторых случаях информацию удобно записывать в файл без преобразования, т. е. в символьном виде, пригодном для непосредственного отображения на экран. Для этого можно использовать функции форматированного ввода-вывода:

– int fprintf(FILE *f, const char *fmt, …) , где FILE *f – указатель на файл, в который производится запись, const char *fmt – форматная строка, …– список переменных, которые записываются в файл.

Функция возвращает число записанных символов.

– int fscanf(FILE *f, const char *fmt, par1, par2, …), где FILE *f – указатель на файл, из которого производится чтение, const char *fmt – форматная строка, par1, par2, … – список переменных, в которые заносится информация из файла.

Функция возвращает число переменных, которым присвоено значение.

Средства прямого доступа дают возможность перемещать указатель текущей позиции в потоке на нужный байт. Для этого используется функция fseek:

– int fseek(FILE *f, long offs, int orig), где FILE *f – указатель на файл, long offs – позиция смещения, int orig – начало отсчета.

Смещение задается выражением или переменной и может быть отрицательным, т. е. возможно перемещение как в прямом, так и в обратном направлениях.

Начало отсчета задается одной из определенных в файле <stdio.h> констант:

SEEK_SET = 0 – начало файла;

SEEK_CUR = 1 – текущая позиция;

SEEK_END = 2 – конец файла.

Функция возвращает 0, если перемещение в потоке выполнено успешно, иначе возвращает ненулевое значение.

Для работы с файловыми потоками в стиле языка C++ к проекту необходимо подключить заголовочные файлы библиотек:

– <ifstream.h> – для работы с входными потоками,
– <ofstream.h> – для работы с выходными потоками,
– <fstream.h> – для работы с двунаправленными потоками.

В библиотечных файлах потоки описаны как классы, т. е. представляют собой пользовательские типы данных (аналогично структурам данных). В описание класса, кроме данных, добавляются описания функций, обрабатывающих эти данные

(методы). Обращаться к методам можно также как и к полям данных – с помощью оператора «.» или «->».

Для создания файлового потока используются специальные методы – конструкторы, которые создают поток соответствующего класса, открывают файл с указанным именем и связывают файл с потоком:

- ifstream(const char *name, int mode = ios::in);
- ofstream(const char *name, int mode = ios::out | ios::trunc);
- fstream(const char *name, int mode = ios::in | ios::out);

Вторым параметром конструкторов является режим открытия файла (mode). Перечень доступных режимов открытия файлов, определенных в классе ios, представлен в таблице 3.

Таблица 3 - Режимы открытия файлов, определенные в классе ios

Режим	Описание режима открытия файла
ios::in	открыть файл для чтения
ios::out	открыть файл для записи
ios::ate	установить указатель на конец файла, читать нельзя, можно только записывать данные в конец файла
ios::app	открыть файл для добавления
ios::trunc	если файл существует, то создать новый
ios::binary	открыть в двоичном режиме
ios::nocreate	если файл не существует, выдать ошибку, новый файл не открывать
ios::noreplace	если файл существует, выдать ошибку, существующий файл не открывать

Открыть файл в программе можно с использованием либо конструкторов –

```
fstream f(“file.dat”, ios::in);
```

либо метода open, имеющего такие же параметры, как и в соответствующем конструкторе –

```
fstream f;
f.open(“file.dat”, ios::in);
```

После того как файловый поток открыт, работать с ним можно также как и со стандартными потоками cin и cout. При чтении данных из входного файла надо контролировать, был ли достигнут конец файла после очередной операции вывода. Это можно делать с помощью метода eof().

Если в процессе работы возникнет ошибочная ситуация, то потоковый объект принимает значение, равное 0.

Когда программа покидает область видимости потокового объекта, то он уничтожается, при этом перестает существовать связь между потоковым объектом и физическим файлом, а сам файл закрывается. Если файл требуется закрыть раньше, то используется метод close().

Пример чтения данных из файла и вывода их на экран монитора:

```
char CH;
f.open("file.dat", ios::in);

do
{
      f >> CH;
      if (f.eof()) break;
      cout << CH;
}
while(!f.eof());
f.close();
```

Строка в C++ – это массив символов, заканчивающийся нуль-символом – ‘\0’ (нуль-терминатором). По положению нуль-терминатора определяется фактическая длина строки. Количество элементов в таком массиве на 1 больше, чем изображение строки.

Присвоить значение строке с помощью оператора присваивания нельзя. Поместить строку в массив можно:

либо с помощью инициализации –

```
char s1[10] = "string1";
char s2[] = "string2";
```

либо при вводе –

```
char s3[10];
cin >> s3;
```

либо воспользовавшись специальной функцией, например, strcpy –

```
char s3[10];
cin >> s3;
char *s4 = new char[strlen(s3)+1];
strcpy(s4, s3);
```

Для работы со строками существует специальная библиотека string (к проекту подключается заголовочный файл библиотеки string.h). Некоторые из функций для работы со строками из библиотеки string приведены в таблице 4.

Таблица 4 – Некоторые функции для работы со строками

Прототип функции	Описание	Примечание
unsigned strlen(const char* s);	Вычисляет длину строки s.	–
int strcmp(const char* s1, const char* s2);	Сравнивает строки s1 и s2.	Если s1 < s2, то результат отрицательный, если s1 == s2, то результат равен 0, если s2 > s1 – результат положительный.
int strcnmp(const char* s1, const char* s2);	Сравнивает первые n символов строк s1 и s2.	Если s1 < s2, то результат отрицательный, если s1 == s2, то результат равен 0, если s2 > s1 – результат положительный.
char* strcpy(char* s1, const char* s2);	Копирует символы строки s1 в строку s2.	–
char* strncpy(char* s1, const char* s2, int n);	Копирует n символов строки s1 в строку s2.	Конец строки отбрасывается или дополняется пробелами.
char* strcat(char* s1, const char* s2);	Приписывает строку s2 к строке s1	–
char* strncat(char* s1, const char* s2);	Приписывает первые n символов строки s2 к строке s1	–
char* strdup(const char* s);	Выделяет память и переносит в нее копию строки s	–

Строки, при передаче в функцию, в качестве фактических параметров могут быть определены либо как одномерные массивы типа char[], либо как указатели типа char*. В отличие от обычных массивов в функции не указывается длина строки, т. к. в конце строки есть признак конца строки – ‘\0’ (нуль-символ).

2.7. Лабораторная работа №7 – Приложение ОС Windows с выводом в окно текста и фигур

Задание: Написать с применением интегрированной среды разработки приложение с графическим интерфейсом пользователя ОС Microsoft Windows, которое в соответствии с номером варианта задания выполняет вывод текста и фигур в окно.

Варианты заданий:

1. В окно бледно-зеленого цвета выведите три красных круга диаметром 200 пикселей, расположенные пирамидой (один круг в центре над двумя другими). Выведите поверх изображения свою фамилию шрифтом Times, размером 80 пикселей, перечеркнутым, синего цвета. Надпись расположите горизонтально.

2. В окно голубого цвета выведите три концентрических квадрата с размерами сторон 300, 200 и 100 пикселей. Внешний квадрат нарисуйте толстым зеленым пером, средний – фиолетовым, внутренний – красным. Квадраты должны быть прозрачными. Выведите поверх изображения свою фамилию шрифтом Arial, размером 80 пикселей, жирным, желтого цвета. Надпись расположите под углом 15° к горизонтали.

3. В окно желтого цвета выведите три соприкасающихся круга диаметром 200 пикселей каждый, расположенные горизонтально и залитые красным, зеленым и синим цветами. Выведите поверх изображения свою фамилию шрифтом Courier размером 80 пикселей, курсивом, черного цвета. Надпись расположите под углом 45° (снизу вверх).

4. В окно красного цвета выведите залитый синим цветом квадрат, ориентированный под углом 45° к осям координат. Такой наклонный квадрат рисуется с помощью функции Polygon() класса TCanvas. Выведите поверх изображения свою фамилию шрифтом Arial размером 80 пикселей, жирным, желтого цвета. Надпись расположите вертикально (снизу вверх).

5. В окно желтого цвета выведите три концентрические окружности с диаметрами 400, 300 и 200 пикселей. Внешнюю окружность нарисуйте толстым (6-8 пикселей) зеленым пером, среднюю – синим, а внутреннюю – коричневым. Образованные круги должны быть прозрачными. Выведите поверх изображения свою фамилию шрифтом Times размером 80 пикселей, курсивом, синего цвета. Надпись расположите вертикально (сверху вниз).

6. В окно серого цвета выведите круг розового цвета диаметром 300 пикселей, у которого правая средняя четверть (сектор размером 90°) покрашена в синий цвет. Выведите поверх изображения свою фамилию шрифтом Courier размером 80 пикселей, жирным, красного цвета. Надпись расположите под углом 45° (сверху вниз).

7. В окно голубого цвета выведите три концентрических круга с диаметрами 300, 200 и 100 пикселей. Закрасьте внешний круг зеленым цветом, средний – желтым, а внутренний – белым. Выведите поверх изображения свою фамилию шрифтом Times размером 80 пикселей, жирным, курсивом, темно-синего цвета. Надпись расположите горизонтально.

8. В окно желтого цвета выведите круг диаметром 400 пикселей, верхняя половина которого закрашена синим, а нижняя - красным цветом. Части окружности рисуются с помощью функций Pie() или Chord() класса TCanvas. Выведите поверх изображения свою фамилию вертикально снизу вверх шрифтом Courier размером 80 пикселей, жирным, перечеркнутым, белого цвета.

9. В окно черного цвета выведите три концентрических квадрата с размерами сторон 400, 300 и 200 пикселей. Внешний квадрат закрасьте желтым цветом, средний – синим, а внутренний – красным. Выведите поверх изображения свою фамилию шрифтом Times размером 80 пикселей, жирным, подчеркнутым, светло-серого цвета. Надпись расположите горизонтально, «вверх ногами».

10. В окно светло-зеленого цвета выведите круг диаметром 400 пикселей, залитый желтым цветом, верхняя четверть которого (до соответствующей хорды) закрашена темно-зеленым цветом. Фигуры, ограниченные частью окружности и хордой, рисуются с помощью функции Chord() класса TCanvas. Выведите поверх изображения свою фамилию шрифтом Arial размером 80 пикселей, жирным, курсивом, синего цвета. Надпись расположите под углом 30° к горизонтали.

11. В окно светло-коричневого цвета выведите толстую (6-8 пикселей) фиолетовую окружность диаметром 400 пикселей, левая половина которой заштрихована горизонтально, а правая - вертикально тем же фиолетовым цветом. Части окружности рисуются с помощью функций Pie( ) или Chord( ) класса TCanvas. Кисть для штриховки замкнутой фигуры создается конструктором TBrush с указанием стиля, значения которого можно найти в описании функции CreateHatchBrush( ). Выведите поверх изображения свою фамилию шрифтом Times размером 80 пикселей, курсивом, синего цвета. Надпись расположите вертикально сверху вниз.

12. В окно светло-голубого цвета выведите желтый круг диаметром 400 пикселей, с «выеденной» с правой стороны частью (как у надкусанного яблока). Выведите поверх изображения свою фамилию шрифтом Times размером 80 пикселей, подчеркнутым, жирным, курсивом, красного цвета. Надпись расположите под углом 75° (снизу вверх).

13. В окно коричневого цвета выведите три соприкасающихся круга диаметром 150 пикселей каждый, расположенные вертикально и залитые серым, красным и белым цветами. Выведите поверх изображения свою фамилию шрифтом Times размером 80 пикселей, подчеркнутым и перечеркнутым, черного цвета. Надпись расположите вертикально снизу вверх.

14. В окно серого цвета выведите три вложенных друг в друга квадрата с размерами сторон 300, 200 и 100 пикселей, соприкасающиеся левыми верхними углами. Внешний квадрат закрасьте фиолетовым цветом, средний - темно-синим, внутренний – светло-синим. Выведите поверх изображения свою фамилию шрифтом Courier размером 80 пикселей, жирным, желтого цвета. Надпись расположите под углом 45° (снизу вверх).

15. В окно синего цвета выведите круг диаметром 400 пикселей желтого цвета, у которого верхняя левая четверть (сектор размером 90°) прозрачна. Сектор рисуется с помощью функции Pie() класса TCanvas. Выведите поверх изображения свою фамилию шрифтом Arial размером 80 пикселей, перечеркнутым, жирным, красного цвета. Надпись расположите под углом 75° (снизу вверх).

16. В окно желтого цвета выведите два эллипса синего цвета с общим центром, расположенные один горизонтально, а другой вертикально. Выведите поверх изображения свою фамилию шрифтом Times размером 80 пикселей, узким (шириной 10 пикселей), жирным, красного цвета. Надпись расположите горизонтально.

17. В окно бледно-зеленого цвета выведите круг диаметром 400 пикселей, левая половина которого закрашена темно-красным, а правая - ярко-красным цветом. Полукруги рисуются с помощью функций Pie( ) или Chord( ) класса TCanvas. Выведите поверх изображения свою фамилию шрифтом Courier размером 80 пикселей, подчеркнутым, жирным, светло-серого цвета. Надпись расположите под углом 15° сверху вниз.

18. В окно светло-бирюзового цвета выведите круг диаметром 400 пикселей, разделенный пополам диагональю, проходящей вправо и вверх. Левую половину круга закрасьте красным цветом, правую – синим. Полукруги рисуются с помощью функций Pie( ) или Chord( ) класса TCanvas. Выведите поверх изображения свою фамилию шрифтом Times размером 80 пикселей, широким (шириной 80 пикселей), курсивом, черного цвета. Надпись расположите горизонтально.

19. В окно светло-серого цвета выведите прямоугольный треугольник синего цвета с длиной катетов 400 пикселей. Треугольник рисуется с помощью функции Polygon() класса TCanvas. Выведите поверх изображения свою фамилию шрифтом Arial размером 80 пикселей, курсивом, белого цвета. Надпись расположите под углом 75°.

20. В окно бледно-розового цвета выведите три концентрических эллипса с размерами 500x300, 400x200 и 200x100 пикселей. Внешний эллипс нарисуйте толстым (6-8 пикселей) черным пером, средний - синим, а внутренний - желтым. Эллипсы должны быть прозрачными. Выведите поверх изображения свою фамилию шрифтом Times размером 80 пикселей, перечеркнутым, узким (шириной 10 пикселей), красного цвета. Надпись расположите горизонтально.

21. В окно голубого цвета выведите изображение Солнца в виде желтого круга, эллиптической орбиты Земли вокруг него и самой Земли в виде коричневого кружка в верхней точке этой орбиты. Выведите поверх изображения горизонтально свою фамилию шрифтом Courier размером 20 пикселей, подчеркнутым, белого цвета. Надпись расположите горизонтально.

22. В окно синего цвета выведите большой желтый треугольник. Треугольник рисуется с помощью функции Polygon() класса TCanvas. Выведите поверх изображения свою фамилию шрифтом Arial размером 80 пикселей, узким (шириной 20 пикселей), черного цвета. Надпись расположите под углом 45° сверху вниз.

23. В окно синего цвета выведите изображение красного полукруглого солнца (как бы заходящего за горизонт) с расходящимися от него несколькими лучами в виде толстых красных линий. Линии рисуются функцией LineTo(), начальная точка перемещается функцией MoveTo(). Половины круга рисуются функциями Pie( ) или Chord( ) класса TCanvas. Выведите поверх изображения свою фамилию шрифтом Times размером 80 пикселей, жирным, курсивом, светло-серого цвета. Надпись расположите горизонтально вверх ногами.

24. В окно зеленого цвета выведите друг над другом два полукруга, соприкасающихся своими округлыми частями. Верхний полукруг залейте синим цветом, нижний - красным. Половины круга рисуются функциями Pie() или Chord() класса TCanvas. Выведите поверх изображения свою фамилию шрифтом Courier размером 80 пикселей, подчеркнутым и перечеркнутым, белого цвета. Надпись расположите горизонтально.

25. В окно светло-голубого цвета выведите желтое изображение солнца с расходящимися от него несколькими толстыми желтыми лучами. Линии рисуются функцией LineTo( ), начальная точка перемещается функцией MoveTo() класса TCanvas. Выведите поверх изображения свою фамилию шрифтом Arial размером 80 пикселей, перечеркнутым, жирным, курсивом, красного цвета. Надпись расположите под углом 30° снизу вверх.

26. В окно голубого цвета выведите удлиненный ромб, расположенный горизонтально и залитый темно-красным цветом. Ромб рисуется с помощью функции Polygon( ) класса TCanvas. Выведите поверх изображения свою фамилию шрифтом Times размером 150 пикселей, белого цвета. Надпись расположите вертикально (сверху вниз).

27. В окно бледно-зеленого цвета выведите фигуру наподобие песочных часов – два узких сектора, соединенные вершинами и нарисованные толстым белым пером. Секторы должны казаться

прозрачными. Секторы рисуются вызовом функции Pie() класса TCanvas. «Насыпьте» на дно песочных часов немного желтого песка (воспользуйтесь для этого функцией Chord( ) класса TCanvas). Выведите поверх изображения свою фамилию шрифтом Times размером 80 пикселей, узким (шириной 15 пикселей), красного цвета. Надпись расположите горизонтально вверх ногами.

28. В окно коричневого цвета выведите два соприкасающихся круга диаметром 200 пикселей каждый, расположенные по диагонали и залитые зеленым и синим цветами. Выведите поверх изображения свою фамилию шрифтом Courier небольшого размера (30 пикселей), но большой ширины (80 пикселей), жирным, красного цвета. Надпись расположите под углом 30° снизу вверх.

29. В окно светло-голубого цвета выведите три круга, расположенные на одной горизонтальной прямой так, что каждый следующий круг сдвинут относительно предыдущего на величину своего радиуса. Первый круг покрасьте в красный цвет, второй - в белый, а третий - в зеленый. Выведите поверх изображения свою фамилию шрифтом Times размером 80 пикселей, жирным, черного цвета. Надпись расположите под углом 5° сверху вниз.

30. В окно желтого цвета выведите неправильный выпуклый пятиугольник темно-синего цвета. Такая фигура рисуется с помощью функции Polygon() класса TCanvas. Выведите поверх изображения свою фамилию шрифтом Arial размером 10 пикселей, красного цвета. Надпись расположите горизонтально «вверх ногами».

Методические указания

Программа должна иметь одно окно (Form1).

Для организации формирования изображения на канве окна приложения (Form1->Canvas) необходимо на форме окна разместить объект TTimer, создав обработчик события OnTimer и установив интервал срабатывания события Timer1->Interval = 1. Обработчик события OnTimer должен выполнять вывод текста и

фигур в окно приложения.

Для поворота текста или графической фигуры относительно границ окна приложения воспользуйтесь функциями SetGraphicsMode() и SetWorldTransform().

Чтобы вывести на канву окна приложения текст, необходимо использовать функцию TextOutA().

Для задания цвета используйте функцию RGB(r, g, b), где r, g и b – компоненты, соответствующие цветовой модели RGB: канал красного цвета (r), канал зеленого цвета (g) и канал синего цвета (b).

При построении линий и контуров графических фигур используется объект «перо» (Form1->Canvas->Pen), при «заливке» цветом и узором замкнутых контуров фигур – объект «кисть» (Form1->Canvas->Brush).

Пример текста программы:

```
//---------------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop
#include <math.h>

#include "Unit1.h"
#pragma package(smart_init)
#pragma resource "*.dfm"

TForm1 *Form1;

int r = 0, g = 0, b = 0;
//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner) : TForm(Owner)
{
}
//---------------------------------------------------------------------------
void __fastcall TForm1::FormPaint(TObject *Sender)
{
Form1->Canvas->Pen->Style = psClear;
this->Canvas->Brush->Color = RGB(50, 200, 100);
```

```
Canvas->Ellipse(100, 100, 500, 500);

Canvas->Brush->Color = clYellow;
Canvas->Brush->Style = bsSolid;
Canvas->Chord(100, 100, 500, 500, 500, 100, 100,100);

Timer1Timer(Sender);
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Timer1Timer(TObject *Sender)
{
SetGraphicsMode(Canvas->Handle, GM_ADVANCED);
XFORM xform;
xform.eM11 = cos(0.2);
xform.eM12 = -sin(0.2);
xform.eM21 = sin(0.2);
xform.eM22 = cos(0.2);
xform.eDx = (float) 100;
xform.eDy = (float) 150;

SetWorldTransform(Canvas->Handle, &xform);

r = r + 10;
g = r + g;
b = (r * 40) + g + b;

if (r >= 255) r = 0;
if (g >= 255) g = 0;
if (b >= 255) b = 0;

Canvas->Brush->Color = RGB(255, 255, 255);
Canvas->Brush->Style = bsSolid;

Canvas->Font->Color = RGB(r, g, b);
Canvas->Font->Name = "Arial";
Canvas->Font->Style = Canvas->Font->Style << fsItalic << fsBold;
Canvas->Font->Size = 27;
```

```
Canvas->TextOutA(100, 200, "Student");

Form1->Canvas->Pen->Style = psDash;
Form1->Canvas->Pen->Color = clBlack;
Form1->Canvas->Pen->Width = 1;
Canvas->MoveTo(100, 100);
Canvas->LineTo(200, 200);
Canvas->LineTo(100, 200);
}
//---------------------------------------------------------------------------
```

2.8. Лабораторная работа №8 – Средства ввода/вывода в ОС Windows

Задание: Написать с применением интегрированной среды разработки приложение с графическим интерфейсом пользователя ОС Microsoft Windows, которое в соответствии с номером варианта задания выполняет файловый ввод/вывод данных, их преобразование и ввод/вывод информации в главное окно приложения, используя стандартные компоненты графического интерфейса пользователя.

Варианты заданий:

1. Запишите в файл два целых числа. Прочтите записанные в файл данные. Получите в программе сумму прочитанных чисел. Выведите в окно приложения исходные числа и результат вычисления с поясняющими надписями.

2. Запишите в файл два дробных числа в экспоненциальном формате. Прочтите записанные в файл данные. Получите в программе произведение прочитанных чисел и округлите его до целого. Выведите в окно приложения имя выбранного вами файла и результат вычислений.

3. Запишите в файл два дробных числа в формате с десятичной точкой. Прочтите записанные в файл данные. Получите в программе сумму и разность прочитанных чисел. Выведите в окно приложения исходные числа и результаты вычислений с поясняющими надписями.

4. Запишите в файл свою фамилию и год рождения. Прочтите записанные в файл данные. Выведите в окно приложения прочитанную фамилию и ваш возраст от 1900 г.

5. Запишите в файл три целых числа. Прочтите записанные в файл данные. Получите в программе среднее значение прочитанных чисел. Выведите в окно приложения исходные числа и результат вычисления с поясняющими надписями.

6. Запишите в файл два числа: целое и дробное в экспоненциальном формате. Прочтите записанные в файл данные. Получите в программе произведение прочитанных чисел и округлите его до целого. Выведите в окно приложения имя выбранного вами файла и результат вычисления.

7. Запишите в файл два числа: целое и дробное в формате с десятичной точкой. Прочтите записанные в файл данные. Получите в программе сумму и разность прочитанных чисел. Выведите в окно приложения исходные числа и результаты вычислений с поясняющими надписями.

8. Запишите в файл целое число и название учебной группы. Прочтите записанные в файл данные. Выведите в окно приложения прочитанное название группы и квадрат исходного числа.

9. Запишите в файл свою фамилию и возраст. Прочтите записанные в файл данные. Получите в программе число прожитых вами дней. Выведите в окно приложения результат вычисления и прочитанную фамилию.

10. Запишите в файл год своего рождения. Прочтите записанные в файл данные. Получите в программе число прожитых вами месяцев. Выведите в окно приложения свою фамилию и результат вычисления.

11. Запишите в файл два целых числа. Прочтите записанные в файл данные. Получите в программе геометрическое среднее прочитанных чисел. Выведите в окно приложения исходные числа и результат вычислений.

12. Запишите в файл два дробных числа в формате с десятичной точкой. Прочтите записанные в файл данные. Получите в программе частное прочитанных чисел и округлите его до целого. Выведите в окно приложения имя выбранного вами файла и результат вычисления.

13. Запишите в файл два числа: дробное в формате с фиксированной точкой и целое. Прочтите записанные в файл данные. Получите в программе частное прочитанных чисел и округлите его до целого. Выведите в окно приложения имя выбранного вами файла и результат вычисления с поясняющей надписью.

14. Запишите в файл два дробных числа: первое в формате с десятичной точкой, а второе в экспоненциальном формате. Прочтите записанные в файл данные. Получите в программе произведение прочитанных чисел. Выведите в окно приложения исходные числа и результат вычисления с поясняющими надписями.

15. Запишите в файл число в формате с десятичной точкой и свою фамилию. Прочтите записанные в файл данные. Получите в программе корень из прочитанного числа. Выведите в окно приложения результат вычисления и прочитанную фамилию, добавив к ней инициалы.

16. Запишите в файл дробное число в формате с десятичной точкой. Прочтите записанные в файл данные. Получите в программе целую и дробную части прочитанного числа. Выведите в окно приложения исходное число и результаты вычислений с поясняющими надписями.

17. Запишите в файл название учебного предмета и целое число. Прочтите записанные в файл данные. Получите в программе квадрат прочитанного числа. Выведите в окно приложения результат вычисления с поясняющей надписью и прочитанное название предмета.

18. Запишите в файл свое имя и дробное число в экспоненциальном формате. Прочтите записанные в файл данные. Получите в программе квадратный корень из прочитанного числа. Выведите в окно приложения прочитанное имя, исходное число и результат вычисления.

19. Запишите в файл число в экспоненциальном формате и свою фамилию. Прочтите записанные в файл данные. Получите в программе корень из прочитанного числа. Получите в программе целую и дробную части прочитанного числа. Выведите в окно приложения прочитанную фамилию и результаты вычислений.

20. Запишите в файл свою фамилию и год рождения. Прочтите записанные в файл данные. Выведите в окно приложения прочитанную фамилию и оставшееся вам «время жизни» до 3000 г.

21. Запишите в файл два числа: дробное в экспоненциальном формате и целое. Прочтите записанные в файл данные. Возведите первое число в степень второго. Выведите в окно приложения исходные числа и результат вычисления с поясняющими надписями.

22. Запишите в файл два дробных числа: первое в экспоненциальном формате, а второе - в формате с десятичной точкой. Прочтите записанные в файл данные. Получите в

программе произведение прочитанных чисел. Выведите в окно приложения результат вычисления и имя выбранного вами файла.

23. Запишите в файл число в экспоненциальном формате и свою фамилию. Прочтите записанные в файл данные. Получите в программе корень из прочитанного числа. Выведите в окно приложения результат вычисления и прочитанную фамилию, предварив ее инициалами.

24. Запишите в файл свою фамилию и год рождения. Прочтите записанные в файл данные. Получите в программе свой возраст. Выведите в окно приложения прочитанную фамилию и результат вычисления.

25. Запишите в файл два числа: дробное в формате с десятичной точкой и целое. Прочтите записанные в файл данные. Получите в программе произведение прочитанных чисел и округлите его до целого. Выведите в окно приложения имя выбранного вами файла и результат вычисления.

26. Запишите в файл два дробных числа: первое в формате с десятичной точкой, а второе в экспоненциальном формате. Прочтите записанные в файл данные. Получите в программе разность прочитанных чисел. Выведите в окно приложения результат вычисления в обоих форматах (экспоненциальном и с десятичной точкой).

27. Запишите в файл дробное число в формате с десятичной точкой и свою фамилию. Прочтите записанные в файл данные. Получите в программе куб прочитанного числа. Выведите в окно приложения прочитанную фамилию и результат вычисления.

28. Запишите в файл свой возраст. Прочтите записанные в файл данные. Получите в программе число прожитых вами месяцев. Выведите в окно приложения свою фамилию и результат вычислений.

29. Запишите в файл два числа: дробное в экспоненциальном формате и целое. Прочтите записанные в файл данные. Получите в программе сумму прочитанных чисел. Выведите в окно приложения исходные числа и результат вычисления с поясняющими надписями.

30. Запишите в файл два дробных числа: первое в экспоненциальном формате, а второе в формате с десятичной точкой. Прочтите записанные в файл данные. Получите в программе произведение прочитанных чисел. Выведите в окно приложения исходные числа и результат вычисления с поясняющими надписями.

Методические указания

Программа должна иметь одно окно (Form1).

В форме окна необходимо разместить объекты TSaveDialog и TOpenDialog для создания диалогового окна сохранения и открытия файлов соответственно. В форме окна также необходимо разместить объекты TButton: «Сохранить данные в файл…», «Открыть файл с данными и преобразовать их» Объектам TButton назначить соответствующие обработчики (методы).

Для организации ввода и вывода данных в окне приложения необходимо на его форме разместить интерфейсные элементы TEdit.

Для получения доступа для чтения/записи данных, отображаемых в окне ввода текста TEdit, необходимо обратиться к полю Text:

```
AnsiString S = Edit1->Text;
```

и

```
Edit1->Text = S;
```

AnsiString – класс динамической строки. При объявлении переменные типа AnsiString инициализируются пустыми строками:

```
AnsiString S = "";
```

Для AnsiString определены операции отношения ==, !=, >, <, >=, <=. Сравнение производится с учетом регистра. Сравниваются коды символов, начиная с первого, и если очередные символы не одинаковы, строка, содержащая символ с меньшим кодом считается меньше. Если все символы совпали, но одна строка длиннее и в ней имеются еще символы, то она считается больше, чем короткая.

Для AnsiString определены операции присваивания =, += и операция склеивания строк (конкатенации) +. Определена также операция индексации []. Индексы начинаются с 1. Например, если S1 = "Привет", то S1[1] вернет 'П', S1[2] вернет 'р' и т.д.

Основные методы класса AnsiString представлены в таблице 5 (в описаниях методов через S1 обозначена строка, метод которой используется).

Таблица 5 – Основные методы класса AnsiString

Метод класса	Синтаксис и описание
AnsiCompare	int __fastcall AnsiCompare(const AnsiString &rhs) Сравнивает данную строку S1 с rhs с учетом регистра. Сравнение зависит от текущих установок Windows и может отличаться от сравнения, осуществляемого операциями сравнения. Возвращает значение > 0 при S1 > rhs, значение < 0 при S1 < rhs и значение 0 при S1 == rhs
AnsiCompareIC	int __fastcall AnsiCompareIC(const AnsiString & rhs) Осуществляет сравнение, аналогичное AnsiCompare, но без учета регистра

Таблица 5 – Основные методы класса AnsiString (продолжение)

Метод класса	Синтаксис и описание
AnsiPos	int __fastcall AnsiPos(const AnsiString & subStr) Возвращает индекс первого символа первого вхождения subStr в S1. Индексы начинаются с 1. Если subStr не содержится в S1, возвращается 0. В отличие от Pos поддерживает многобайтные символы
c_str	char* __fastcall c_str() Возвращает указатель на строку с нулевым символом в конце, содержащую те же символы, что в AnsiString
CurrToStr	static AnsiString __fastcall CurrToStr(Currency value) Преобразует значение value типа Currency в строку
Delete	void __fastcall Delete(int index, int count) Удаляет из строки, начиная с позиции index, число символов, равное count
Insert	void __fastcall Insert(const AnsiString &str, int index) Вставляет в строку подстроку str, начиная с индекса index
IsDelimiter	bool __fastcall IsDelimiter(const AnsiString &delimiters, int index) Возвращает true, если символ с индексом index является одним из разделителей, указанных в строке delimiters. Работает и для многобайтных символов

Таблица 5 – Основные методы класса AnsiString (продолжение)

Метод класса	Синтаксис и описание
IsEmpty	bool __fastcall IsEmpty() Возвращает true, если строка пустая
Length	int __fastcall Length() Возвращает число символов в строке
LowerCase	AnsiString __fastcall LowerCase() Возвращает строку, в которой все символы приведены к нижнему регистру. Не влияет на исходную строку
Pos	int __fastcall Pos(const AnsiString &subStr) Возвращает индекс первого символа первого вхождения subStr в S1. Индексы начинаются с 1. Если subStr не содержится в S1, возвращается 0. В отличие от AnsiPos не поддерживает многобайтные символы
SetLength	void __fastcall SetLength(int newLength) Усекает строку до newLength символов. Если исходная строка короче, то она не увеличивается
SubString	AnsiString __fastcall SubString(int index, int count) Возвращает подстроку, начинающуюся с символа в позиции index и содержащую count символов
ToDouble	double __fastcall ToDouble() Преобразует строку в число с плавающей запятой. Если строка не соответствует формату числа с плавающей запятой, генерируется исключение EConvertError

Таблица 5 – Основные методы класса AnsiString (продолжение)

Метод класса	Синтаксис и описание
Tolnt	int __fastcall ToInt() Преобразует строку в целое число. Если строка не соответствует формату целого числа, генерируется исключение EConvertError
Trim	AnsiString __fastcall Trim() Возвращает строку, соответствующую исходной, но без пробельных символов до и после значащих символов
UpperCase	AnsiString __fastcall UpperCase() Возвращает строку, в которой все символы приведены к верхнему регистру. Не влияет на исходную строку

Для использования диалога сохранения файла выполните операции:

```
if (SaveDialog1->Execute())
{
char *STR = new char[256];
AnsiString FileName = SaveDialog1->FileName;
STR = FileName.c_str();

FILE *fp;
fp = fopen(STR, "w");
…
fclose(fp);
}
```

Для использования диалога открытия ранее сохраненного файла выполните операции:

```
if (OpenDialog1->Execute())
{
char *STR = new char[256];
AnsiString FileName = OpenDialog1->FileName;
STR = FileName.c_str();

FILE *fp;
fp = fopen(STR, "r");
…
fclose(fp);
}
```

Для записи строки, отображаемой в окне ввода текста TEdit, в текстовый файл необходимо осуществить последовательность операций:

```
char *STR1 = new char[256];
StrPCopy(STR1, Edit1->Text);
fputs(STR1, fp);
fputs("\n", fp);
```

Для чтения строки из текстового файла выполните последовательность операций:

```
char *STR1 = new char[256];
int n = 256;
fgets(STR1, n, fp);
Edit1->Text = StrPas(STR1);
```

Функция StrPCopy(char *Dest, AnsiString Source) копирует строку Source типа AnsiString в строку Dest, заканчивающуюся нуль-символом, и возвращает указатель на Dest. Данная функция не производит проверку длины. Поэтому, чтобы не возникало ошибок, необходимо, чтобы результирующая строка Dest имела

длину, равную Length(Source) + 1 символ.

Функция StrPas(char *Str) преобразовывает строку Str, заканчивающуюся нуль-символом, в строку типа AnsiString и возвращает указатель на строку AnsiString.

2.9. Лабораторная работа №9 – Преобразование растровых изображений

Задание: Написать с применением интегрированной среды разработки приложение с графическим интерфейсом пользователя ОС Microsoft Windows, которое в соответствии с номером варианта задания выполняет открытие (загрузку) и преобразование растровых изображений, их вывод в главное окно приложения, используя стандартные компоненты графического интерфейса пользователя.

Варианты заданий:

1. Преобразуйте цветное изображение в серое, сохранив градации яркости.

2. Преобразуйте цветное изображение в черно-белое.

3. Переведите изображение в красный цвет, сохранив градации яркости.

4. Преобразуйте цветное изображение в инверсное серое с сохранением градаций яркости.

5. Разделите изображение на 3 слоя по яркости, покрасив самый светлый слой в ярко-синий цвет, средний – в синий, а темный слой – в темно-синий цвет.

6. Преобразуйте цветное изображение в черно-белое инверсное.

7. Выделите в изображении темные области, оставив их цвет; светлые области закрасьте белым.

8. Выделите в изображении светлые области, оставив их цвет; темные области закрасьте фиолетовым цветом.

9. Переведите изображение в синий цвет, сохранив градации яркости.

10. Разделите изображение на 3 слоя по яркости, покрасив самый светлый слой в желтый цвет, средний – в коричневый, а темный слой – в темно-коричневый цвет.

11. Переведите изображение в зеленый цвет, сохранив градации яркости.

12. Преобразуйте цветное изображение в серое с 256 градациями яркости.

13. Закрасьте темно-красным темные области изображения.

14. Закрасьте желтым светлые области изображения.

15. Переведите изображение в фиолетовый цвет, сохранив градации яркости.

16. Разделите изображение на 2 слоя по яркости, покрасив светлый слой в красный цвет, а темный – в синий.

17. Переведите изображение в оттенки голубого цвета, сохранив градации яркости.

18. Преобразуйте цветное изображение в инверсное серое с сохранением градаций яркости.

19. Разделите изображение на 2 слоя по яркости и инвертируйте его, покрасив светлый слой в синий цвет, а темный – в светло-голубой.

20. Переведите изображение в желтый цвет, сохранив градации яркости.

21. Разделите изображение на 2 слоя по яркости, покрасив светлый слой в светло-серый цвет, а темный – в темно-серый.

22. Преобразуйте цветное изображение в монохромное синее.

23. Разделите изображение на 3 слоя по яркости, покрасив самый светлый слой в салатный цвет, средний – в зеленый, а темный слой – в темно-зеленый цвет.

24. Разделите изображение на 2 слоя по яркости, покрасив светлый слой в желтый цвет, а темный – в коричневый.

25. Преобразуйте цветное изображение в монохромное зеленое.

26. Выделите в изображении темные области, оставив их цвет; светлые области закрасьте бледно-желтым.

27. Преобразуйте цветное изображение в монохромное желто-коричневое.

28. Выделите в изображении темные области, оставив их цвет; светлые области закрасьте желтым.

29. Преобразуйте цветное изображение в монохромное красное.

30. Разделите изображение на 3 слоя по яркости, покрасив самый светлый слой в желтый цвет, средний – в красный, а темный – в зеленый.

Методические указания

Программа должна иметь одно окно (Form1). Окно должно быть масштабируемым (свойство Form1->BorderStyle = bsSizeable).

В форме окна необходимо разместить объект TOpenDialog для создания диалогового окна открытия файлов растровых изображений. В форме окна также необходимо разместить объект TMainMenu, и используя MenuDesigner создать пункт главного меню объекта TMainMenu с названием «Файл». В MenuDesigner в пункте главного меню «Файл» создать подменю, включающее пункты: «Открыть файл…», «Преобразовать изображение», «Выход». Пунктам подменю «Файл» назначить соответствующие обработчики (методы).

В форме окна разместить объект TImage и установить его свойства Image1->Align = alClient и Image1->Stretch = true.

Исходное изображение и изображение после преобразования необходимо вывести в объект TImage.

Для удаления изображения из объекта TImage используйте операции:

```
Image1->Picture->Graphic = NULL;
Image1->Repaint();
```

Для загрузки изображения из графического файла в объект TImage выполните операции:

```
if (OpenDialog1->Execute())
{
Image1->Picture->Graphic = NULL;
Image1->Picture->LoadFromFile(OpenDialog1->FileName);
Image1->Repaint();
}
```

Объект TImage поддерживает отображение растровых графических файлов Bitmap Picture (файлы с расширением bmp).

Чтобы получить доступ к содержимому определенного пиксела изображения, загруженного в объект TImage, необходимо воспользоваться операциями:

```
long int PIX = Image1->Canvas->Pixels[XR][YR];
```

и

```
Image1->Canvas->Pixels[XR][YR] = PIX;
```

где: XR и YR – координаты пиксела изображения, PIX – содержимое пиксела (цвет).

Для получения значения яркости каждого из цветовых каналов цвета пиксела (каналы R (красный), G (зеленый) и B (синий)) необходимо выполнить операции:

```
int R = GetRValue(Image1->Canvas->Pixels[XR][YR]);
int G = GetGValue(Image1->Canvas->Pixels[XR][YR]);
int B = GetBValue(Image1->Canvas->Pixels[XR][YR]);
```

Для формирования цвета определенного пиксела изображения из значений яркостей каждого из цветовых каналов воспользуйтесь операцией:

```
Image1->Canvas->Pixels[XR][YR] = (TColor) RGB(R, G, B);
```

Для конвертирования цветного изображения в изображение в оттенках серого выполните операции:

```
long int Gray = (R * 0.56) + (G * 0.33) + (B * 0.11);

Image1->Canvas->Pixels[XR][YR] = (TColor) RGB(Gray, Gray,
Gray);
```

2.10. Лабораторная работа №10 – Построение графика функции

Задание: Написать с применением интегрированной среды разработки приложение с графическим интерфейсом пользователя ОС Microsoft Windows, которое в соответствии с номером варианта задания выполняет построение графика функции *y*(*x*).

Варианты заданий:

1. $y(x)=x^2e^{4-x}$

2. $y(x)=2xe^{-\frac{x}{2}}$

3. $y(x)=|x|e^{-|x|}$

4. $y(x)=xe^{-x}$

5. $y(x)=xe^{-x^2}$

6. $y(x)=\frac{x}{2}e^{-x+1}$

7. $y(x)=x^2e^{-x}$

8. $y(x)=x^3e^{-\frac{x^2}{2}}$

9. $y(x)=xe^{-\frac{x^2}{2}}$

10. $y(x)=2(x-3)^2e^{x-2}$

11. $y(x)=x^3e^x$

12. $y(x)=(x+1)e^{-\frac{x^2}{4}}$

13. $y(x)=x^2e^x$

14. $y(x)=(x-2)e^{3-x}$

15. $y(x)=x^3e^{-x}$

16. $y(x)=\left(4x^2+2x\right)e^{2x-1}$

17. $y(x)=(x-4)e^{-2x}$

18. $y(x)=(2x-1)e^{2(1-x)}$

19. $y(x)=(x+2)e^{\frac{1}{x}}$

20. $y(x)=\left(x^2+x\right)e^x$

21. $y(x)=\frac{1}{3}(1-x)e^{3x+1}$

22. $y(x)=-(2x+1)e^{2(x+1)}$

23. $y(x)=(3-x)e^{x+2}$

24. $y(x)=(x+2)e^{x-1}$

25. $y(x)=(x+2)e^{\frac{1}{x}}$

26. $y(x)=-(x+2)e^{x+2}$

27. $y(x)=10(x-2)e^{-x}$

28. $y(x)=(x-1)e^{3x+1}$

29. $y(x)=(3-x)e^{x-2}$

30. $y(x)=2(x-6)^2 e^{\frac{1}{2}x-3}$

31. $y(x)=5(x-2)e^{x-1}$

32. $y(x)=-\frac{1}{2}\left(x^2+7x+14\right)e^{-\frac{1}{2}x}$

33. $y(x)=(x-2)e^{-\frac{1}{x}}$

34. $y(x)=\left(2x^2+x+1\right)e^{1-x}$

35. $y(x)=x^2 e^{-x^2}$

Методические указания

Программа должна иметь одно окно (Form1).

С помощью элементов интерфейса ОС Microsoft Windows (интерфейсные элементы TEdit) вводятся данные для построения функции (начальное значение аргумента *x* (тип float), конечное значение аргумента *x* (тип float), постоянный шаг (приращение) аргумента функции *x* (тип float)).

При вычислении значений функции *y*(*x*) (тип float) программой должен в прямоугольной системе координат строиться её график на канве (TCanvas) объекта TImage по точкам: X = *x* и Y = *y*(*x*) (значения *x* и *y*(*x*) округляются до ближайших целых значений (тип long int)). Точки графика функции соединяются прямыми линиями с помощью методов MoveTo() и LineTo() объекта TCanvas (Form1->Image1->Canvas->MoveTo() и Form1->Image1->Canvas->LineTo()). Начало координат графика функции совпадает с центром объекта Image1.

Рядом с объектом TImage должен находиться объект TStringGrid, имеющий три столбца: номер точки, значение координаты *x*, значение координаты *y*. Ячейки всех трех столбцов при вычислении значений функции *y*(*x*) заполняются соответствующими данными.

В окне должна находиться кнопка TButton, при нажатии которой происходит применение введенных данных для построения функции *y*(*x*), вычисление значений *x* и *y*, построение графика функции *y*(*x*), используя объект TImage, и заполнение ячеек объекта TStringGrid значениями вычисленных координат *x* и *y*.

График функции должен иметь цвет clRed (красный) и строиться в окне программы. Должно быть предусмотрено масштабирование графика функции коэффициентом масштабирования (тип float): масштаб задается через интерфейсный элемент TEdit. Коэффициент масштабирования должен быть равен или больше 1. Оси координат должны иметь цвет clBlack (черный).

К проекту необходимо подключить заголовочный файл

библиотеки math.h. Для округления координат *x* и *y* до ближайших целых значений надо воспользоваться функцией floor из библиотеки math:

double fx = floor(*x*);

где *x* – округляемая величина (тип double).

Список литературы

1. Финогенов К.Г. Лабораторный практикум "Основы программирования на языке C++ ". - М.: МИФИ, 2004.
2. Финогенов К.Г. Лабораторный практикум "Основы разработки приложений Windows". Кн. 1-2. М: - МИФИ, 2004 - 2005.
3. Финогенов К.Г. Основы объектно-ориентированного программирования. Лабораторный практикум. - М.: МИФИ, 2008.
4. Невская, Е.С. Искусство программирования: Учеб. пособие для вузов / Е.С. Невская, А.А. Чекулаева, М.И. Чердынцева – Москва: Вузов-ская книга, 2002. - 207 с.
5. Страуструп, Б. Язык программирования C++ для профессионалов / Б. Страуструп. – Москва: Интернет-Университет Информационных Технологий, 2006. – 568 с.