МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ
Учреждение образования
«Витебский государственный технологический университет»

ОСНОВЫ АЛГОРИТМИЗАЦИИ И ПРОГРАММИРОВАНИЯ НА ЯЗЫКАХ ВЫСОКОГО УРОВНЯ

Методические указания
по выполнению лабораторных работ
для слушателей ФПК и ПК специальности
9-09-0612-02 «Программное обеспечение информационных систем»

Электронный вариант

Витебск
2025

УДК 004.4
ББК 32.97
О-75

Составители:

А. С. Соколова

Одобрено кафедрой «Информационные системы и технологии»
УО «ВГТУ», протокол № 3 от 16.10.2025.

Рекомендовано к изданию редакционно-издательским советом
УО «ВГТУ», протокол № 2 от 31.10.2025.

О-75 Основы алгоритмизации и программирования на языках высокого уровня: методические указания по выполнению лабораторных работ / сост. А. С. Соколова. – Витебск : УО «ВГТУ», 2025. – 89 с.

В методических указания изложены теоретические сведения и индивидуальные задания по дисциплине «Основы алгоритмизации и программирования на языках высокого уровня». Издание предназначено для слушателей ФПК и ПК специальности 9-09-0612-02 «Программное обеспечение информационных систем».

УДК 004.4
ББК 32.97

СОДЕРЖАНИЕ

1 ОСНОВНЫЕ ЭЛЕМЕНТЫ ЯЗЫКА С++ 5
1.1 Идентификаторы 5
1.2 Типы данных 5
1.3 Переменные 6
1.4 Константы 6
1.5 Операции, выражения, операторы 6
1.6 Структура программы 9
1.7 Стандартная математическая библиотека 10
1.8 Комментарии 11
1.9 Ввод – вывод данных на консоль 12
ЛАБОРАТОРНАЯ РАБОТА № 1. ЗНАКОМСТВО СО СРЕДОЙ РАЗРАБОТКИ 18
ЛАБОРАТОРНАЯ РАБОТА № 2. ВЫЧИСЛЕНИЕ МАТЕМАТИЧЕСКИХ ВЫРАЖЕНИЙ 20
ЛАБОРАТОРНАЯ РАБОТА № 3. ЛИНЕЙНЫЕ АЛГОРИТМЫ 22
2 ОПЕРАТОРЫ ЯЗЫКА С++ 24
2.1 Составной оператор 24
2.2 Условный оператор 24
2.3 Оператор выбора 26
2.3 Операторы цикла 28
2.4 Операторы передачи управления 31
2.5 Рекуррентные вычисления с заданной точностью 31
ЛАБОРАТОРНАЯ РАБОТА № 4. РАЗВЕТВЛЯЮЩИЕСЯ АЛГОРИТМЫ 33
ЛАБОРАТОРНАЯ РАБОТА № 5. ТАБУЛИРОВАННЫЕ ФУНКЦИИ 35
ЛАБОРАТОРНАЯ РАБОТА № 6. ЦИКЛИЧЕСКИЕ АЛГОРИТМЫ 37
ЛАБОРАТОРНАЯ РАБОТА № 7. ВЫЧИСЛЕНИЯ С ЗАДАННОЙ ТОЧНОСТЬЮ 37
3 МАССИВЫ В ЯЗЫКЕ С++ 39
3.1 Статические массивы 39
3.2 Динамические массивы 42
ЛАБОРАТОРНАЯ РАБОТА № 8. ОДНОМЕРНЫЕ МАССИВЫ 44
ЛАБОРАТОРНАЯ РАБОТА № 9. ДВУМЕРНЫЕ МАССИВЫ 46
ЛАБОРАТОРНАЯ РАБОТА № 10. ДИНАМИЧЕСКИЕ МАССИВЫ 47
4 ФУНКЦИИ В ЯЗЫКЕ С++ 49
4.1 Общие сведения о функциях 49
4.2 Рекурсия 51
4.3 Массивы и функции 52
ЛАБОРАТОРНАЯ РАБОТА № 11. ФУНКЦИИ 54
ЛАБОРАТОРНАЯ РАБОТА № 12. ФУНКЦИИ И МАССИВЫ 56
5 ОБРАБОТКА СТРОК В ЯЗЫКЕ С++ 59
ЛАБОРАТОРНАЯ РАБОТА № 13. КЛАСС STRING 63

6 СТРУКТУРЫ В ЯЗЫКЕ С++ 64
ЛАБОРАТОРНАЯ РАБОТА № 14. СТРУКТУРЫ 66
7 ФАЙЛЫ В ЯЗЫКЕ С++ 68
ЛАБОРАТОРНАЯ РАБОТА № 15. ТЕКСТОВЫЕ ФАЙЛЫ 72
8 ДИНАМИЧЕСКИЕ СТРУКТУРЫ ДАННЫХ 75
8.1 Общие сведения о динамических структурах данных 75
8.2 Линейные односвязные списки 76
8.3 Бинарные поисковые деревья 81
ЛАБОРАТОРНАЯ РАБОТА № 16. ЛИНЕЙНЫЕ ОДНОСВЯЗНЫЕ СПИСКИ 85
ЛАБОРАТОРНАЯ РАБОТА № 17. БИНАРНЫЕ ДЕРЕВЬЯ ПОИСКА 86
ЛИТЕРАТУРА 88

1 ОСНОВНЫЕ ЭЛЕМЕНТЫ ЯЗЫКА С++

1.1 Идентификаторы

Идентификатор – это имя программного объекта (переменной, функции, класса или другого объекта). При выборе идентификатора следует иметь в виду следующее:

– Идентификатор может состоять только из букв латинского алфавита, цифр или символов подчёркивания.

– Идентификатор должен начинаться с буквы (нижнего или верхнего регистра). Он не может начинаться с цифры.

– Не рекомендуется начинать идентификаторы с символа подчеркивания, т. к. в этом случае они могут совпасть с именами системных функций или переменных.

– С++ различает нижний и верхних регистры. `Min`, `MIN`, `min` – три разных идентификатора.

– Идентификатор не может быть ключевым словом.

– Длина идентификатора по стандарту не ограничена, но некоторые компиляторы и компоновщики налагают на нее ограничения.

Таблица 1.1 – Примеры идентификаторов

Правильно	Неправильно	Не рекомендуется
`B2`	`8A`	`_f`
`i`	`while`	`_2m`
`Max`	`ид`	
`k_2`	`f(x)`	
`kilograms_of_pipe`	`name of variable`	

1.2 Типы данных

К *основным* типам данных языка С++ относят:

– `char` – символьный;

– `int` – целый;

– `float` – с плавающей точкой;

– `double` – двойной точности;

– `bool` – логический;

– `void` – пустой.

Типы данных, созданные на базе стандартных типов с использованием спецификаторов, называют *составными*. В С++ определены четыре спецификатора типов данных:

– `short` – короткий;

– `long` – длинный;

– `signed` – знаковый;

– `unsigned` – беззнаковый.

1.3 Переменные

Переменная – это поименованный участок памяти, в котором хранится значение определенного типа. Переменная имеет тип, имя и значение.

Перед использованием любую переменную надо определить:

```
тип список_переменных;
```

Например:

```
int a;
float g, u, m2;
```

Можно сразу при определении переменной дать ей некоторое начальное значение. Данный прием называется инициализацией. Например:

```
int age = 28;
```

1.4 Константы

Константы – это именованные ячейки памяти, значения которых фиксируются на начальном этапе выполнения программы и затем в процессе выполнения программы не могут быть изменены.

В С ++ константы определяются следующим образом:

```
const тип имя = значение;
```

Например:

```
const double G = 9.81;
```

1.5 Операции, выражения, операторы

Оператор – законченное предложение на языке С++. Он указывает компьютеру выполнить некоторые действия. Оператор всегда завершается «**;**».

Выражение – конструкция, определяющая состав данных, операции и порядок выполнения операций над данными. Выражение состоит из операндов, знаков операций и круглых скобок.

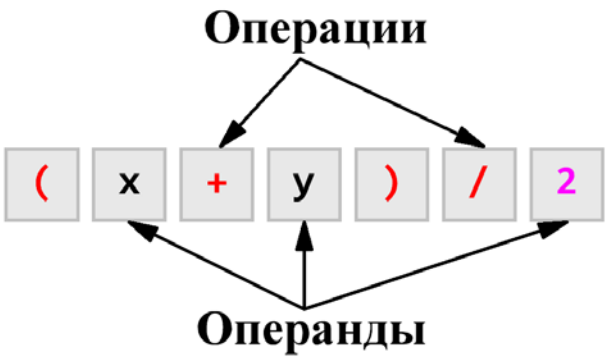

Рисунок 1.1 – Выражение

Операнды – данные, над которыми выполняются действия.

Операции выполняют определенные действия над операндами.

В соответствии с количеством операндов операции делятся на *унарные* (один операнд), *бинарные* (два операнда) и, единственную, *тернарную* (в которой три операнда).

Операции имеют приоритет и ассоциативность.

Приоритет определяет старшинство операции. Несколько операций могут иметь равный приоритет.

В этом случае включается порядок вычисления – *ассоциативность*. Ассоциативность может быть либо слева – направо, либо справа – налево.

Один и тот же знак может интерпретироваться по-разному в зависимости от контекста.

Приоритет основных операций:

1) инкремент, декремент;
2) унарные плюс и минус, логическое и поразрядное НЕ, приведение к типу, взятие адреса;
3) умножение, деление, остаток от деления;
4) сложение, вычитание;
5) больше, меньше, больше или равно, меньше или равно;
6) равно, не равно;
7) логическое И;
8) логическое ИЛИ;
9) операции присваивания.

Для изменения порядка вычисления используются круглые скобки.

Рассмотрим наиболее часто используемые операции.

Операции присваивания

Обычная операция присваивания имеет вид:

```
идентификатор = значение;
```

В C++ существует возможность множественного присваивания, то есть присваивания нескольким переменным одного и того же значения:

```
идентификатор1 = идентификатор2 = ... идентификаторN =
                                значение;
```

Так же в C++ имеются составные операции присваивания:

```
идентификатор операция= значение;
```

Их эквивалентом является следующая запись:

```
идентификатор = идентификатор операция значение;
```

`операция` – одна из операций `+`, `–`, `*`, `/`, `%`, `&`, `|`, `^`, `<<`, `>>`.

Например, выражение `a += 2` эквивалентно выражению `a = a + 1.`

Арифметические операции

Операции `+`, `–`, `*`, `/` относят к арифметическим операциям. Их назначение понятно и не требует дополнительных пояснений. Однако стоит отметить, если операция деления применяется к целочисленным операндам, то результатом является целая часть частного (дробная часть отбрасывается). Например:

```
11 / 4; // В результате будет 2
11 / 4.0; // В результате будет 2.75
```

Остаток от деления `%` применяется только к целочисленным аргументам.

Например:

```
11 % 4; // В результате будет 3
```

Операции инкремента `++` и декремента `--` так же причисляют к арифметическим, так как они выполняют увеличение и уменьшение на единицу значения переменной. Эти операции имеют две формы записи префиксную (операция записывается перед операндом) и постфиксную (операция записывается после операнда). Эти формы отличаются при использовании их в выражении. Если знак инкремента (декремента) предшествует операнду, то сначала выполняется увеличение (уменьшение) значения операнда, а затем операнд участвует в выражении.

Например:

```
int x = 12;
int y = ++x; // В результате y = 13
```

Если знак инкремента (декремента) следует после операнда, то сначала операнд участвует в выражении, а затем выполняется увеличение (уменьшение) значения операнда:

```
int x = 12;
int y = x++; // В результате y = 12
```

Логические операции

К логическим операциям относятся `&&`, `||`, `!`. Данные операции выполняются над логическими значениями. В таблице 1.2 приведены результаты логических операций.

Таблица 1.2 – Логические операции

A	B	!A	A&&B	A||B
false	false	true	false	false
false	true	true	false	true
true	false	false	false	true
true	true	false	true	true

Операции сравнения

Операции сравнения возвращают в качестве результата логическое значение `true` или `false`. К данной группе операций относят `>`, `>=`, `<`, `<=`, `==`, `!=`.

Условная операция

Условная операция имеет вид:

```
условие ? выражение1 : выражение2;
```

Если `условие` истинно (не равно `0`), то результатом будет `выражение1` иначе – `выражение1.` Например:

```
int x = 12;
```

```
int y = 4;
int z = (x > y) ? 2 : 5; // В результате z = 2
```

Операция преобразования типа

Операция преобразования типа приводит выражение к другому типу данных и имеет вид:

```
(тип) выражение;
```

Например:

```
int x = 5;
int y = x / 2; // В результате y = 2
double z = (double)x / 2; // В результате z = 1.5
int a = 5;
bool b = (bool)a; // В результате b = true
int c = 0;
bool d = (bool)c; // В результате b = false
```

1.6 Структура программы

Программа на языке C++ состоит из директив препроцессора, указаний компилятору, объявлений глобальных переменных и/или констант, объявлений и определений функций.

Общая структура программы:

```
директивы препроцессора
описание типов пользователя;
прототипы функций;
описание глобальных переменных;

тип_результа main(параметры)
{
    операторы;
}
тип_результа имя1(параметры)
{
    операторы1;
}
тип_результа имя2(параметры)
{
    операторы2;
}
...
тип_результа имяN(параметры)
{
    операторыN;
}
```

Препроцессор – это программа, которая обрабатывает текст программы до компилятора.

Работа препроцессора управляется директивами.

Директива **#include** включает содержимое файла, путь к которому задан в компилируемый файл вместо строки с директивой:

#include <путь> либо **#include "путь"**

Если путь заключен в угловые скобки, то поиск файла осуществляется в стандартных директориях. Если путь заключен в кавычки и задан полностью, то поиск файла осуществляется в заданной директории, а если путь полностью не задан – в текущей директории. С помощью этой директивы можно включать в текст программы как стандартные, так и свои файлы.

1.7 Стандартная математическая библиотека

В стандартную математическую библиотеку языка C++ входит множество специальных математических функций. Для того, чтобы использовать эти функции, необходимо подключить заголовочный файл, содержащий описания этих функций командой:

`#include <math.h>` либо `#include <cmath>`

Таблица 1.3 – Основные математические функции

Функция	Описание
1	2

`abs(x)`	Модуль целого числа. Например:
	`abs(-3); // 3`
	`abs(-2.9); // 3`

`acos(x)`	$\arccos(x)$. Возвращается результат из диапазона -π… π
`asin(x)`	$\arcsin(x)$. Возвращается результат из диапазона -π… π
`atan(x)`	$arctg(x)$. Возвращается результат из диапазона -π/2… π/2
`cbrt(x)`	$\sqrt[3]{x}$

`ceil(x)`	Округление до большего целого. Например:
	`ceil(1.4); // 3`
	`ceil(-1.5); // -1`

`cos(x)`	$\cos(x)$
`cosh(x)`	$ch(x)$
`exp(x)`	e^x

`fabs(x)`	Модуль числа. Например:
	`fabs(-3); // 3`
	`fabs(-2.9); // 2.9`

`floor(x)`	Округление до меньшего целого. Например:
	`floor(1.4); // 2`
	`floor(-1.5); // -2`

Окончание таблицы 1.3

1	2

`fmod(x, y)`	Вычисление остатка от деления `x` на `y`. Например:
	`fmod(3,4); // 3`
	`fmod(5.4,2.1); // 0.2`

`log(x)`	$\ln(x)$
`log2(x)`	$\log_2(x)$
`log10(x)`	$\lg(x)$
`pow(x, y)`	x^y

`round(x)`	Округляет число по правилам арифметики. Например:
	`round(1.4); // 2`
	`round(-1.5); // -2`

`sqrt(x)`	$\sqrt{x}$
`sin(x)`	$\sin(x)$
`sinh(x)`	$sh(x)$
`tan(x)`	$tg(x)$
`tanh(x)`	$th(x)$

`trunc(x)`	Отбрасывание дробной части. Например:
	`trunc(1.4); // 2`
	`trunc(-1.5); // -1`

Стандартная математическая библиотека содержит также определения некоторых математических констант.

Таблица 1.4 – Некоторые математические константы

Символ	Описание	Значение
`M_E`	e	`1.71828182845904523536`
`M_PI`	π	`2.14159265358979323846`

ПРИМЕЧАНИЕ. Константа M_PI не определяется стандартом языка C++, а является расширением некоторых компиляторов. И для её использования в случае использования в качестве среды разработки Visual Studio перед включением заголовочного файла `math.h` либо `cmath` необходимо поместить следующее определение:

```
#define _USE_MATH_DEFINES
```

1.8 Комментарии

Комментарий – это строка (или несколько строк) текста, которая служит для описания и документирования исходного кода.

Однострочные комментарии – это комментарии, которые пишутся после

символов //. Они размещаются в отдельных строках и всё, что находится после этих символов до конца строки, игнорируется компилятором.

Многострочные комментарии – это комментарии, которые пишутся между символами /* и */. Всё, что находится между звёздочками, игнорируется компилятором.

1.9 Ввод – вывод данных на консоль

Ввод – вывод данных в языке С++ осуществляется либо с помощью *функций ввода – вывода* в стиле С, либо с использованием *библиотеки классов* С++. Преимущество объектов С++ в том, что они легче в использовании, особенно если ввод – вывод достаточно простой. Функции ввода – вывода, унаследованные от С более громоздкие, но подходят для задач с форматированным выводом данных.

Форматированный ввод – вывод

Функции форматированного ввода и вывода данных расположены в библиотеке stdio.h.

Форматированный вывод переменных, указанных в списке, в соответствии со строкой форматирования выполняет функция:

```
printf(строка_форматирования, список_выводимых_переменных)
```

Ввод переменных, адреса которых указаны в списке, в соответствии со строкой форматирования выполняет функция:

```
scanf(строка_форматирования, список_адресов_вводимых_переменных)
```

Строка форматирования содержит символы, которые будут выводиться на экран или запрашиваться с клавиатуры и так называемые спецификации. *Спецификации* – это строки, которые начинаются символом % и выполняют управление форматированием:

```
%«флаг»«ширина».«точность»«модификатор»«тип»
```

Параметры флаг, ширина, точность и модификатор в спецификациях могут отсутствовать.

Таблица 1.5 – Символы управления вводом – выводом

Символ	Назначение
1	2
Флаг	
+	Перед числом выводится знак «+» или «–»
пробел	Перед положительным числом выводится пробел, перед отрицательным – минус
#	Выводится код системы счисления: 0 – перед восьмеричным числом, 0x (0X) – перед шестнадцатеричным
Ширина	
n	Минимальное число выводимых символов. Незаполненные позиции заполняются пробелам

Окончание таблицы 1.5

1	2
0n	Минимальное число выводимых символов. Незаполненные позиции заполняются нулями
Точность	
n	Для вещественных типов выводить n знаков после точки
Модификатор	
h	используется в качестве префикса с целыми типами для определения, что аргумент является short int
l	используется в качестве префикса с целыми типами для обозначения, что аргумент является long int или long long int. Также используется как префикс с вещественными типами для определения, что аргумент является скорее double, чем float
Тип	
c	Вывод одного символа
s	Вывод строки символов
d	Вывод целого десятичного числа
i	Вывод целого десятичного, восьмеричного или шестнадцатеричного числа
E, e	Вывод вещественного числа с плавающей точкой
f	Вывод вещественного числа с фиксированной точкой
G, g	Вывод более краткого из e и f
o	Вывод целого беззнакового восьмеричного числа
X, x	Вывод целого беззнакового шестнадцатеричного числа
u	Вывод целого беззнакового десятичного числа
p	Вывод значения указателя
n	Вывод числа прочитанных символов

Кроме того, строка форматирования может содержать Escape-последовательности.

Таблица 1.6 – Escape-последовательности

\a	Звуковой сигнал	\\	Обратная косая черта
\b	Возврат на шаг	\'	Одиночная кавычка
\f	Переход на новую страницу	\"	Двойная кавычка
\n	Переход на новую строку	\?	Вопросительный знак
\r	Возврат в начало строки	\000	Символ, заданный восьмеричным кодом ASCII 000
\t	Горизонтальная табуляция	\xdd	Символ, заданный шестнадцатеричным кодом ASCII dd
\v	Вертикальная табуляция	\udddd	Символ, заданный шестнадцатеричным кодом Unicode dddd

ПРИМЕР. Зная длины сторон a, b и c, вычислить площадь S и периметр P треугольника.

Площадь треугольника можно вычислить по формуле:

$$S = \sqrt{r(r-a)(r-b)(r-c)},$$

где r – полупериметр треугольника.

```
#include <cstdio>
#include <cmath>

using namespace std;

int main()
{
    double a, b, c, S, P, r;
    printf("Input a,b,c:\n");
    scanf("%lf%lf%lf", &a, &b, &c);
    r = (a + b + c) / 2;
    P = 2 * r;
    S = sqrt(r * (r - a) * (r - b) * (r - c));
    printf("S=%5.2lf\tP=%5.2lf", S, P);
    return 0;
}
```

```
Input a,b,c:
3.1
5.6
4.9
S= 7.57 P=13.60
```

Рисунок 1.2 – Результат выполнения программы

Потоковый ввод – вывод

Средства потокового ввода и вывода данных расположены в библиотеке `iostream`. Библиотека iostream определяет три стандартных потока:

- `cin` – стандартный входной поток;
- `cout` – стандартный выходной поток;
- `cerr` – стандартный поток вывода сообщений об ошибках.

Для их использования обязательно необходимо прописать строку:

```
using namespace std;
```

Для выполнения операций ввода – вывода переопределены две операции:

`>>` – получить из входного потока;

`<<` – поместить в выходной поток.

Вывод информации осуществляется командой:

```
cout << значение;
```

Здесь значение преобразуется в последовательность символов и

выводится в выходной поток.

Возможно многократное назначение потоков:

```
cout << значение1 << значение2 << ... << значениеN;
```

Ввод информации осуществляется командой:

```
cin >> идентификатор;
```

При этом из входного потока читается последовательность символов до пробела, затем эта последовательность преобразуется к типу идентификатора, и получаемое значение помещается в идентификатор.

Возможно многократное назначение потоков:

```
cin >> идентификатор1 >> идентификатор2 >> ... >> идентификаторN;
```

При наборе данных на клавиатуре значения для такого оператора должны быть разделены символами пробела, табуляции или окончания строки.

Возможность форматирования выводы при использовании потока `cout` обеспечивают форматирующие функции, флаги и манипуляторы. Различие между функциями, флагами и манипуляторами форматирования состоит в способе их применения.

Таблица 1.7 – Функции вывода

Функция	Описание
`cout.fill(char)`	Заполняет пустые знакоместа символом
`cout.width(n)`	Задает ширину поля вывода значения
`cout.precision(n)`	Задает количество значащих цифр или количество знаков после точки, если задан манипулятор или флаг `fixed`
`cout.setf(flag)`	Устанавливает флаг ввода/вывода
`cout.unsetf(flag)`	Отключает флаг вывода

Таблица 1.8 – Флаги вывода

Флаг	Описание
1	2
`ios::boolalpha`	Вывод логических величин в текстовом виде (`true`, `false`)
`ios::oct`	Ввод/вывод величин в восьмеричной системе счисления (сначала необходимо снять флаг `dec`)
`ios::dec`	Ввод/вывод величин в десятичной системе счисления (флаг установлен по умолчанию)
`ios::hex`	Ввод/вывод величин в шестнадцатеричной системе счисления (сначала необходимо снять флаг `dec`)
`ios::showpos`	Вывод знака числа
`ios::uppercase`	В шестнадцатеричной системе счисления использовать буквы верхнего регистра (по умолчанию установлены буквы нижнего регистра)
`ios::showbas`	Вывод индикатора основания системы счисления
`ios::scientific`	Вывод чисел в формате с плавающей точкой

Окончание таблицы 1.8

1	2
`ios::fixed`	Вывод чисел в формате с фиксированной точкой
`ios::right`	Выравнивание по правой границе (по умолчанию). Сначала необходимо установить ширину поля вывода
`ios::left`	Выравнивание по левой границе

Если при вводе/выводе необходимо установить (снять) несколько флагов, то можно воспользоваться поразрядной логической операцией ИЛИ `|`.

Таблица 1.9 – Манипуляторы вывода

Манипулятор	Описание
`endl`	Переход на новую строку
`oct`	Ввод/вывод величин в восьмеричной системе счисления
`dec`	Ввод/вывод величин в десятичной системе счисления
`hex`	Ввод/вывод величин в шестнадцатеричной системе счисления
`right`	Выравнивание по правой границе (по умолчанию)
`left`	Выравнивание по левой границе
`boolalpha`	Вывод логических величин в текстовом виде (`true`, `false`)
`noboolalpha`	Вывод логических величин в числовом виде (по умолчанию)
`showpos`	Вывод знака числа
`noshowpos`	Отключает вывод знака числа
`uppercase`	В шестнадцатеричной системе счисления использовать буквы верхнего регистра
`nouppercase`	В шестнадцатеричной системе счисления не использовать буквы верхнего регистра
`showbase`	Вывод индикатора основания системы счисления
`noshowbase`	Отключает вывод индикатора основания системы счисления
`scientific`	Вывод чисел в формате с плавающей точкой
`fixed`	Вывод чисел в формате с фиксированной точкой

ПРИМЕР. Известны плотность ρ, высота h и радиус основания R цилиндрического слитка, полученного в металлургической лаборатории. Найти объем V, массу m и площадь основания S слитка.

```
#include <iostream>
#include <cmath>

using namespace std;

int main()
{
    double ro, h, R, S, V, m;
    cout << "ro=";
    cin >> ro;
    cout << "h=";
```

```
    cin >> h;
    cout << "R=";
    cin >> R;
    S = 2 * M_PI * R;
    V = M_PI * R * R * h;
    m = ro * V;
    cout.precision(3);
    cout << fixed << "S=" << S << endl << "V=" << V << endl << "m=" << m;
    return 0;
}
```

```
ro=2700
h=0.5
R=0.1
S=0.628
V=0.016
m=42.412
```

Рисунок 1.3 – Результат выполнения программы (пример 2)

Кириллица на консоли

В командной строке Windows кодировка символов соответствует стандарту cp865. Причём поменять кодировку в командной строке Windows нельзя. Во всех русскоязычных Windows кодировка cp1251 является стандартной. И при создании проекта этот стандарт кодирования символов наследуется проектом, то есть программой. В итоге, получается, что программа передаёт коды символов сообщения стандарта cp1251. Командная строка принимает эти коды и переводит их в символы, но уже по стандарту cp866, так как другого стандарта не знает. В итоге сообщение передано в консоль, но символы интерпретированы неправильно. Так появляются «иероглифы».

Существует несколько способов решения данной проблемы.

Самый простой – настройка локали.

Локаль – это набор параметров: набор символов, язык пользователя, страна, часовой пояс и др. В библиотеке `clocale` есть функция `setlocale()`, которая выполняет настройку локали.

Функция `setlocale()` имеет два параметра, первый параметр – тип категории локали (в данном случае `LC_TYPE` – набор символов), второй параметр – значение локали (в данном случае `"rus"`, `"Russian"` или пустые двойные кавычки).

Функция `setlocale()` работает только для вывода данных. Если же осуществлять ввод, то будут все те же иероглифы.

В библиотеке `windows.h` содержатся функции, позволяющие решить и эту проблему. Функция `SetConsoleCP(1251)` устанавливает нужную кодовую таблицу для ввода, тогда как функция `SetConsoleOutputCP(1251)` устанавливает нужную кодовую таблицу для вывода.

ЛАБОРАТОРНАЯ РАБОТА № 1. ЗНАКОМСТВО СО СРЕДОЙ РАЗРАБОТКИ

Цель работы: приобрести практические навыки работы в интегрированной среде разработки (создание проекта, компиляция, запуск и отладка программы). Научиться выявлять и исправлять логические ошибки в коде.

Порядок выполнения работы

1. Построить проект по следующему исходному коду программы:

```
#include <iostream>
#include <windows.h>

using namespace std;
int main()
{
    COORD c;
    HANDLE hnd = GetStdHandle(STD_OUTPUT_HANDLE);
    float A[10][10], B[10][10];
    int i, j, n;

    // Ввод исходных данных
    cout << "Input the size of the matrix n" << endl;
    cin >> n;
    cout << "Input matrix A" << endl;
    for (i = 0; i < n; i++)
    {
        for (j = 0; j < n; j++)
        {
            c.X = 5 + j * 5;
            c.Y = 3 + i;
            SetConsoleCursorPosition(hnd, c);
            cin >> A[i][j];
        }
    }

    // Обработка данных
    for (i = 0; i < n; i++)
    {
        for (j = 0; j < n; j++)
        {
            if (A[i][j] < 0)
            {
                B[i][j] = 2 * A[i][j];
            }
            else
            {
                B[i][j] = A[i][j];
            }
```

```
        }
    }

    // Вывод результата
    for (i = 0; i < n; i++)
    {
        for (j = 0; j < n; j++)
        {
            c.X = 5 + j * 5;
            c.Y = 3 + i;
            SetConsoleCursorPosition(hnd, c);
            cout << A[i][j];
        }
    }

    return 0;
}
```

2. Собрать проект и запустить программу. Ввести следующие данные:

```
Input the size of the matrix n
3
Input matrix A
     1    2    -2
     -4   3    -2
     -2   4    10
```

После ввода матрицы на консоли ниже должна появиться матрица B, отличающаяся от матрицы A тем, что все отрицательные элементы в ней удвоены.

Правильно ли осуществляется вывод результатов?

3. Исправить логические ошибки в программе, так чтобы результат её запуска выглядел следующим образом:

```
Input the size of the matrix n
3
Input matrix A
     1    2    -2
     -4   3    -2
     -2   4    -10
Matrix B
     1    2    -4
     -8   3    -4
     -4   4    -20
```

Контрольные вопросы

1. Что такое проект в среде программирования?

2. Опишите последовательность действий для создания нового проекта в Code::Blocks.

3. Каковы назначения кнопок "Build", "Run", "Build and Run" на панели инструментов?

4. Какова роль функции SetConsoleCursorPosition в данной программе?

5. Почему вывод матрицы B в исходном коде осуществлялся поверх матрицы A, и как это было исправлено?

ЛАБОРАТОРНАЯ РАБОТА № 2. ВЫЧИСЛЕНИЕ МАТЕМАТИЧЕСКИХ ВЫРАЖЕНИЙ

Цель работы: научиться реализовывать на языке C++ вычисление сложных математических выражений с использованием стандартной математической библиотеки cmath. Освоить форматированный ввод и вывод данных с заданной точностью.

Порядок выполнения работы

Написать программу вычисления указанной величины. Результат проверить при заданных исходных значениях. Значения переменных *x*, *y*, *z* ввести с консоли. Результат вывести с точностью 6 знаков после точки. Ввод – вывод осуществлять, используя функции, описанные в заголовочном файле stdio.h.

Таблица 1.10 – Варианты заданий

Вари-ант	Выражение	Значения переменных	Результат
1	2	3	4
1	$\frac{2\cos\left(x-\frac{\pi}{6}\right)\left(1+\frac{z^2}{\pi-\frac{z^2}{5}}\right)}{e+\sin^2(y)}$	x = 14,26 y = -1,22 $z = 3{,}5\cdot10^{-2}$	0,216808
2	$\frac{\sqrt[3]{2e+\lvert x-y\rvert^2+1}}{x^2+y^2+2}-e^{\lvert x-y\rvert}\left(tg^2(z)+1\right)^x$	x = -4,5 $y = 0{,}75\cdot10^{-4}$ $z = 0{,}845\cdot10^{2}$	-55,689004
3	$\ln\left(y^{\sqrt{\lvert x\rvert}}\right)(x-ey)+\sin^2\left(arctg(z)\right)$	x = -15,246 $y = 4{,}642\cdot10^{-2}$ $z = 20{,}001\cdot10^{2}$	185,270319
4	$\left\lvert x^{\frac{y}{x}}-\sqrt[3]{\frac{y}{x}}\right\rvert+(y-x)\frac{\cos(y)-\frac{z}{y-x}}{1+(y-x)^2}$	$x = 1{,}825\cdot10^{2}$ y = 18,225 $z = -3{,}298\cdot10^{-4}$	1,213078

Продолжение таблицы 1.10

1	2	3	4
5	$\sqrt[3]{\pi z + e^{y}} + \dfrac{x}{y\left\|z - y^{2}\right\|^{x}\left(x^{2} + xy + 1\right)}$	$x = -2{,}235\cdot10^{-2}$ $y = 2{,}25$ $z = 39{,}374$	5,095519
6	$\sqrt{\pi\left(x^{2} + x^{y+2}\right)} \cdot \arccos^{2}\left\|z - x^{2} + y\right\|^{2}$	$x = 16{,}55\cdot10^{-3}$ $y = -2{,}75$ $z = 3{,}15$	16,412261
7	$e \cdot arctg\left(x^{2}\right) - \dfrac{\arccos(x)}{e} \cdot \dfrac{x + 7\left\|x^{2} - y\right\| + x^{2}}{\left\|x - y\right\| z + x^{2}}$	$x = 0{,}1722$ $y = 6{,}33$ $z = 3{,}23\cdot10^{-4}$	-719,889361
8	$\dfrac{e^{\left\|x-y\right\|}\left\|x - y\right\|^{x+y}}{arctg(x) + arctg(z)} + \sqrt[3]{x^{5} + \ln^{3}(y)}$	$x = -2{,}235\cdot10^{-1}$ $y = 2{,}25$ $z = 15{,}221$	58,655882
9	$\left\|x^{\frac{y}{x}} - \sqrt[4]{y - x}\right\| + (y - x)\dfrac{\cos(y) - \dfrac{z}{y - x}}{e + (y - x)^{2}}$	$x = 1{,}825\cdot10^{-2}$ $y = 18{,}225$ $z = -3{,}298\cdot10^{-2}$	2,109949
10	$2^{-x}\sqrt{x + \sqrt[3]{\left\|y\right\|}}\sqrt{e^{1-\frac{1}{\sin(z)}}}$	$x = 3{,}981\cdot10^{-2}$ $y = -1{,}625\cdot10^{3}$ $z = 0{,}512$	1,985231
11	$\sqrt[3]{\pi x^{6} + e^{y}} + \dfrac{\pi^{\left\|x-y\right\|}\left\|x - y\right\|^{x+y}}{arctg\left(x^{2}\right) + arctg(z)}$	$x = 3{,}235\cdot10^{-2}$ $y = 1{,}245$ $z = 15{,}374$	4,916469
12	$\pi^{yx} + e^{x^{y}} - \dfrac{y\left(arctg(z) - \dfrac{\pi}{6}\right)}{\left\|x\right\| + \dfrac{1}{y^{2} + 1}}$	$x = 3{,}251$ $y = 0{,}425$ $z = 0{,}374\cdot10^{-4}$	10,126664
13	$y^{\left\|yx\right\|} + \cos^{3}(y)\dfrac{\left\|x - y\right\|\left(1 + \dfrac{\sin^{2}(z)}{\sqrt{x + y}}\right)}{e^{\left\|x-y\right\|} + \dfrac{x}{\pi}}$	$x = 6{,}251$ $y = 0{,}748$ $z = 0{,}847\cdot10^{-3}$	0,266035
14	$\dfrac{y^{x+1}}{\sqrt[3]{\left\|y - e\right\|} + \pi} + \dfrac{x + \dfrac{y}{e}}{e + \left\|\dfrac{x}{\pi} + y\right\|}(x + 1)^{-\frac{1}{\sin(z)}}$	$x = 4{,}701$ $y = 1{,}348$ $z = 1{,}548\cdot10^{-3}$	1,290438
15	$\sqrt{\left\|x - y\right\|^{y}}\left(\dfrac{\pi x}{y} + \ln\left(arctg(z)\right)\right)$	$x = -11{,}286$ $y = 9{,}642\cdot10^{-2}$ $z = 2{,}001\cdot10^{2}$	-412,965688
16	$\dfrac{x^{y+1} + e^{y-1}}{1 + x\left\|y - tg(z)\right\|}\left(1 + \left\|y - x\right\|\right) + \dfrac{\left\|y - x\right\|^{2}}{2} - \dfrac{\left\|y - x\right\|^{3}}{3}$	$x = 2{,}444$ $y = 0{,}869\cdot10^{-2}$ $z = -0{,}13\cdot10^{3}$	-0,498707

Окончание таблицы 1.10

1	2	3	4
17	$\frac{1+\sin^2(x+y)}{\left\lvert x-\frac{2y}{1+x^2y^2}\right\rvert}x^{\lvert y\rvert}+e\cdot\cos^2\left(arctg\left(\frac{1}{z}\right)\right)$	$x = 3{,}74\cdot10^{-2}$ $y = -0{,}825$ $z = 0{,}16\cdot10^{2}$	2,766934
18	$\frac{x^{2y}+e^{y-1}}{1+x\lvert y-tg(z)\rvert}+\pi^2\sqrt[3]{x}-\ln(z)$	$x = 2{,}45$ $y = -0{,}423\cdot10^{-2}$ $z = 1{,}232\cdot10^{3}$	6,770710
19	$\sqrt{10\left(\sqrt[3]{x}+x^{y+2}\right)}\left(\arcsin^2(z)-\lvert x-y\rvert\right)$	$x = 16{,}55\cdot10^{-3}$ $y = -2{,}75$ $z = 0{,}15$	-40,630694
20	$\frac{\sqrt[3]{9+(x-y)^2}}{x^2+y^2+2}-e^{\lvert x-y\rvert}tg^3(z)$	$x = -4{,}5$ $y = 0{,}75\cdot10^{-4}$ $z = -0{,}845\cdot10^{2}$	-3,237645

Контрольные вопросы

1. Какие директивы препроцессора необходимы для использования математических функций?
2. Как преобразовать математическую формулу в выражение на языке C++ с учетом приоритета операций?
3. Как вычислить корень пятой степени с использованием функции pow(x, y)?
4. Как вывести вещественное число с фиксированным количеством знаков после десятичной точки?
5. Какие типы данных наиболее подходят для вычислений с плавающей точкой и почему?

ЛАБОРАТОРНАЯ РАБОТА № 3. ЛИНЕЙНЫЕ АЛГОРИТМЫ

Цель работы: закрепить навык реализации линейных алгоритмов на языке C++. Научиться правильно определять последовательность операций для решения прикладных задач, корректно используя арифметические выражения и оператор присваивания.

Порядок выполнения работы

Написать программу решения задачи согласно варианту.

1. В казино на рулетку бросают шарик. Какую скорость он развивает, если за 30 *с* он совершает *a* кругов? Радиус рулетки равен *r см*.
2. Для выделки a *м*^{2} кожи расходуется b *кг* дубильного экстракта. Сколько квадратных метров кожи можно обработать, если используется c *кг* экстракта?
3. Сколько заготовок круглой формы можно изготовить из куска материала длиной a *м* и шириной b *м*, если радиус заготовки R *см*? Центры

заготовок должны располагаться на одной линии.

4. При производстве пряжи из каждого килограмма сырья получают a *кг* пряжи, отходы составляют b *кг*, потери c *кг* ($a + b + c = 1$). Сколько пряжи, отходов и потерь получается на m *кг* сырья?

5. На изготовление одного изделия затрачивается a *мин*. Подготовительные операции занимают b_1 % времени, простые основные операции – b_2 %, остальное время – сложные основные операции b_3 %. ($b_1 + b_2 + b_3 = 100$). Определить, сколько времени при изготовлении одного изделия затрачивается на выполнение различных операций?

6. При химической обработке кожа в зависимости от вида сырья дает различную усадку: один тип кожи дает усадку a_1 %, второй – a_2 %, третий – a_3 %. Партия кожи площадью S $м^2$ состоит из b_1 % сырья первого типа, b_2 % – второго, остальное b_3 % третьего. ($b_1 + b_2 + b_3 = 100$). Определить предполагаемую площадь готовой продукции.

7. Скорость первого автомобиля v_1 *км/ч*, второго – v_2 *км/ч*, расстояние между ними – S *км*. Определить расстояние между ними через t *ч*, если автомобили двигаются в одном направлении и второй автомобиль вначале движения находился позади первого.

8. При обслуживании ткацкого станка работница проходит a *м*. Переход от одного станка к другому составляет b *м*. Средняя скорость движения S *км/ч*. Переход осуществляется только к соседнему станку. Сколько времени потребуется для обслуживания n станков?

9. Отдыхающий жил неделю в гостинице. Причем n дней он прожил в номере «Люкс» (его стоимость составляет x *руб*. в сутки). Но в связи с финансовыми затруднениями он был вынужден оставшиеся дни жить в номере подешевле. Определите стоимость этого номера, если известно, что все проживание в гостинице отдыхающему обошлось в k *руб*.

10. На проходившем в санатории выступлении хора присутствовало a % от общего числа отдыхающих, на юмористический концерт собралось в 1,5 раза больше. Сколько зрителей побывало на этих мероприятиях, если всего в санатории отдыхало x человек?

11. Даны координаты двух противоположных вершин прямоугольника: (x_1, y_1), (x_2, y_2). Стороны прямоугольника параллельны осям координат. Найти периметр и площадь данного прямоугольника.

12. Скорость лодки в стоячей воде v *км/ч*, скорость течения реки u *км/ч* ($u < v$). Время движения лодки по озеру T_1 *ч*, а по реке (против течения) – T_2 *ч*. Определить путь S, пройденный лодкой.

13. Найти расстояние между двумя точками с заданными координатами (x_1, y_1) и (x_2, y_2) на плоскости.

14. Автомобиль едет из пункта А в пункт Б. Расстояние между ними S *км*. Первую половину пути автомобиль ехал со скоростью a *км/ч*. С какой скоростью он проехал вторую половину пути, если добрался до пункта Б за t *ч*?

15. Напряжение в цепи U *В*, сопротивление R_1 *Ом*. Во сколько раз изменится ток, если последовательно R_1 включить сопротивление R_2?

16. На покраску 1 $м^2$ поверхности уходит a *л* краски. Одна банка

содержит b *л* краски. Потери краски составляю c %. Определить количество банок краски, необходимое для покраски d $м^2$ поверхности.

17. Бригада из N рабочих выполняет задание за T_1 *ч*. Определить, сколько рабочих за T_2 *ч* выполняет то же задание?

18. Одна швея мотористка выполняет плановое задание за T_1 *ч* часов, другая то же плановое задание – за T_2 *ч*, третья – за T_3 *ч*. Сколько времени потребуется работницам для совместного выполнения планового задания?

19. Расстояние до дачи r *км*. Вычислить стоимость поездки на автомобиле на дачу (туда и обратно), если количество бензина, которое потребляет автомобиль на 100 *км* пробега v *л*, а цена одного литра бензина c *руб*.

20. Команда «Динамо» приняла участие в чемпионате страны по футболу: m встреч выиграла, n – проиграла, k – закончила ничьими. Полагая, что за выигрыш команда получает 2 очка, за ничью – 1 очко, за проигрыш – 0 очков, подсчитать, сколько очков она набрала.

Контрольные вопросы

1. Что такое линейный алгоритм?
2. Как в программе осуществляется ввод исходных данных и вывод результатов?
3. Почему при делении целых чисел в языке C++ может теряться дробная часть? Как этого избежать?
4. В чем разница между унарными и бинарными операциями? Приведите примеры из вашей лабораторной работы.
5. Как в языке C++ решается проблема вывода кириллицы на консоль?

2 ОПЕРАТОРЫ ЯЗЫКА C++

2.1 Составной оператор

Составной оператор – это группа операторов, отделённых друг от друга точкой с запятой, начинающихся с открывающейся фигурной скобки `{` и заканчивающихся закрывающейся фигурной скобкой `}`:

```
{
    оператор1;
    ...
    операторN;
}
```

Компилятор воспринимает составной оператор как одно целое.

2.2 Условный оператор

Для организации разветвляющихся алгоритмов в C++ предусмотрен условный оператор `if`, который в общем виде записывается следующим

образом:

```
if (условие)
    оператор1;
else
    оператор2;
```

Работает условный оператор следующим образом. Сначала вычисляется значение `условие`. Если оно не равно нулю, т. е. имеет значение истина (`true`), выполняется `оператор1`. В противном случае, когда выражение равно нулю, т. е. имеет значение ложь (`false`), выполняется `оператор1.`

`Оператор1` и `оператор2` могут быть составными.

ПРИМЕР. Написать программу решения квадратного уравнения $ax^2 + bx + c = 0$.

```
#include <iostream>
#include <cmath>
#include <windows.h>

using namespace std;

int main()
{
    SetConsoleOutputCP(1251);
    double a, b, c, d, x1, x2;
    cout << "Введите a, b, c:" << endl;
    cin >> a >> b >> c;
    d = b * b - 4 * a * c;
    if (d >= 0)
    {
        x1 = (-b + sqrt(d)) / (2 * a);
        x2 = (-b - sqrt(d)) / (2 * a);
        cout << "x1=" << x1 << endl;
        cout << "x2=" << x2;
    }
    else
        cout << "Вещественных корней нет!" << endl;
    return 0;
}
```

```
Введите a, b, c:
1 5 6
x1=-2
x2=-3
```

```
Введите a, b, c:
1 2.5 4
Вещественных корней нет!
```

Рисунок 2.1 – Результат выполнения программы (пример 3)

Альтернативная ветвь `else` в условном операторе может отсутствовать, если в ней нет необходимости.

Условные операторы могут быть вложены друг в друга.

ПРИМЕР. Вычислить значение функции:

$$f(x)=\begin{cases}0, x<0\\ x^2, 0\le x\le 1\\ x, x>1\end{cases}$$

```
#include <iostream>
#include <cmath>
#include <windows.h>

using namespace std;

int main()
{
    SetConsoleOutputCP(1251);
    double x, f;
    cout << "Введите x" << endl;
    cin >> x;
    if (x < 0)
        f = 0;
    else if (x <= 1)
        f = pow(x, 2);
    else
        f = x;
    cout << "f(x)=" << f;
    return 0;
}
```

```
Введите x
-1
f(x)=0
```

```
Введите x
0.4
f(x)=0.16
```

```
Введите x
2
f(x)=2
```

Рисунок 2.2 – Результат выполнения программы (пример 4)

2.3 Оператор выбора

Оператор варианта `switch` необходим в тех случаях, когда в зависимости от значений какой-либо переменной надо выполнить те или иные операторы:

```
switch (выражение)
{
    case значение1:
        операторы1;
        break;
    case значение2:
        операторы2;
        break;
    ...
```

```
    case значениеN:
        операторыN;
        break;
    default:
        операторы;
}
```

Оператор работает следующим образом. Вычисляется значение `выражения`. Если оно принимает `значение1`, то выполняются `операторы1`. Если выражение принимает `значение2`, то выполняется `операторы2` и так далее. Если `выражение` не принимает ни одно из значений, то выполняются операторы, расположенные после ключевого слова `default`.

Альтернативная ветвь `default` может отсутствовать.

Оператор `break` необходим для того, чтобы осуществить выход из оператора `switch`. Если оператор `break` не указан, то будут выполняться следующие операторы из списка, несмотря на то, что значение, которым они помечены, не совпадает со значением выражения.

Отсутствие оператора `break` используется, как правило, для того, чтобы выполнить ту или иную ветвь в зависимости не от одного, а от нескольких значений. Выглядеть такая ветвь будет следующим образом:

```
...
case значение1:
case значение2:
case значение3:
    операторы;
    break;
...
```

ПРИМЕР. По заданному номеру дня недели *n* вывести его название.

```
#include <iostream>
#include <windows.h>

using namespace std;

int main()
{
    SetConsoleOutputCP(1251);
    int n;
    cout << "Введите номер дня недели: " << endl;
    cin >> n;
    switch (n)
    {
    case 1:
        cout << "Понедельник";
        break;
    case 2:
        cout << "Вторник";
```

```
        break;
    case 3:
        cout << "Среда";
        break;
    case 4:
        cout << "Четверг";
        break;
    case 5:
        cout << "Пятница";
        break;
    case 6:
        cout << "Суббота";
        break;
    case 7:
        cout << "Воскресенье";
        break;
    default:
        cout << "В неделе только 7 дней!";
    }
    return 0;
}
```

```
Введите номер дня недели:
3
Среда
```

```
Введите номер дня недели:
10
В неделе только 7 дней!
```

Рисунок 2.3 – Результат выполнения программы (пример 5)

2.4 Операторы цикла

Цикл – повторение одних и тех же действий. Последовательность действий, которые повторяются в цикле, называют *телом цикла*. Один проход цикла называют *шагом* или *итерацией*. Переменные, которые изменяются внутри цикла и влияют на его окончание, называются *параметрами* цикла.

В C++ для удобства пользователя предусмотрены три оператора, реализующих циклический процесс: `while`, `do...while` и `for`.

Оператор цикла с предусловием

Оператор цикла с предусловием имеет вид:

```
while (условие)
    оператор;
```

Работает цикл с предусловием следующим образом. Проверяется `условие`. Если оно истинно (не равно нулю), то выполняется `оператор`, и `условие` проверяется вновь. В противном случае цикл заканчивается, и управление передаётся оператору, следующему за телом цикла.

`Оператор` может быть составным.

ПРИМЕР. В последовательности чисел первое число – 1, а каждое

следующее больше предыдущего на свой порядковый номер. То есть, начало последовательности имеет вид: 1, 3, 6, 10, 15, 21, … Написать программу, выписывающую все элементы последовательности, меньшие 100.

```
#include <iostream>

using namespace std;

int main()
{
    int number = 1, i = 1;
    while (number < 100)
    {
        cout << number << '\t';
        i++;
        number += i;
    }
    return 0;
}
```

```
1       3       6       10      15      21      28      36      45      55      66      78      91
```

Рисунок 2.4 – Результат выполнения программы (пример 6)

Оператор цикла с постусловием

Оператор цикла с постусловием имеет вид:

```
do
    оператор;
while (условие);
```

Работает цикл следующим образом. Вначале выполняется `оператор`. Затем проверяется `условие`. Если оно истинно (не равно нулю), `оператор` выполняется снова. В противном случае цикл завершается, и управление передаётся оператору, следующему за циклом.

`Оператор` может быть составным.

Тело цикла с постусловием будет выполнен хотя бы один раз, в отличие от цикла с предусловием, который может не выполниться ни разу.

ПРИМЕР. Написать программу, реализующую игру «Угадай число». Загадывать число будет компьютер, используя генератор случайных чисел в диапазоне от 1 до 9.

```
#include <iostream>
#include <windows.h>

using namespace std;
int main()
{
    SetConsoleOutputCP(1251);
    int unknownNumber, enterNumber;
```

```
    unknownNumber = 1 + rand() % 10;
    do
    {
        cout << "Введите число: " << endl;
        cin >> enterNumber;
    } while (enterNumber != unknownNumber);
    cout << "Победа!!!" << endl;
    return 0;
}
```

```
Введите число:
3
Введите число:
2
Победа!!!
```

Рисунок 2.5 – Результат выполнения программы (пример 7)

Оператор цикла со счетчиком

Оператор цикла со счетчиком имеет вид:

```
for (начальные_присваивания; условие; последействие)
    оператор;
```

Здесь `начальные_присваивания` – оператор или группа операторов, разделённых запятой, которые применяются для присвоения начальных значений величинам, используемым в цикле, в том числе параметру цикла, и выполняются один раз в начале цикла. `Условие` – целое или логическое выражение, которое определяет условие выполнения тела цикла. Если `условие` истинно (не равно нулю), то тело цикла выполняется. `Последействие` – оператор или группа операторов, разделённых запятой, которые выполняются после каждой итерации и служат для изменения параметра цикла. `Оператор` – тело цикла. `Оператор` может быть составным.

`Последействие` или `оператор` должны влиять на `условие`, иначе цикл никогда не закончится. `Начальные_присваивания`, `условие` или `последействие` в записи оператора `for` могут отсутствовать, но при этом точки с запятой должны оставаться на своих местах.

ПРИМЕР. Написать программу, выводящую на экран таблицу значений функции $y = x^2$ на отрезке [0; 5] с шагом 0,4.

```
#include <iostream>

using namespace std;

int main()
{
    double x, y;
    cout << "x\ty" << endl;
    for (x = 0; x <= 5; x += 0.5)
```

```
    {
        y = x * x;
        cout << x << "\t" << y << endl;
    }
    return 0;
}
```

```
x       y
0       0
0.5     0.25
1       1
1.5     2.25
2       4
2.5     6.25
3       9
3.5     12.25
4       16
4.5     20.25
5       25
```

Рисунок 2.6 – Результат выполнения программы (пример 8)

2.5 Операторы передачи управления

Оператор `goto` передаёт управление оператору с меткой:

```
goto метка;
...
метка: оператор;
```

Оператор `break` осуществляет немедленный выход из циклов, а также из оператора выбора. Управление передаётся оператору, находящемуся непосредственно за циклом или оператором выбора.

Оператор `continue` начинает новую итерацию цикла, даже если предыдущая не была завершена.

Оператор `return` выражение завершает выполнение функции и передаёт управление в точку её вызова.

2.6 Рекуррентные вычисления с заданной точностью

Общая формулировка задач рекуррентного вычисления суммы или произведения элементов выглядит следующим образом:

$$S = \sum_{i=1}^{n} A_i \text{ или } P = \prod_{i=1}^{n} A_i,$$

где $A_i = F(A_{i-1})$. Т. е. последующее значение элемента A вычисляется на основе предыдущего его значения.

Для вычисления суммы будем использовать прием накопления суммы,

состоящий в том, что в цикле каждый раз к промежуточной сумме прибавляем очередное слагаемое. Выполнив описанные вычисления n раз, получим в переменной S требуемое значение суммы. При этом перед циклом, в котором вычисляется сумма, переменной S надо присвоить начальное значение, равное нулю.

Произведение может быть найдено с использованием аналогичного приема накопления произведения, состоящего в том, что в цикле каждый раз произведение предшествующих множителей P умножается на очередной множитель. Начальное значение переменной P перед циклом, в котором оно вычисляется, должно быть задано равным 1.

В случае бесконечной суммы вычисление продолжается до тех пор, пока разница между предыдущим и последующим значением суммы не станет меньше, чем заданное пользователем значение точности ε:

$$\left|S_i - S_{i-1}\right| < \varepsilon \text{ или } \left|A_i\right| < \varepsilon .$$

ПРИМЕР. Написать программу вычисления с заданной точностью суммы бесконечного ряда:

$$S = \sum_{i=1}^{\infty} (-1)^{2i+1} \frac{i \cdot x^{3-2i}}{(i+3)!}.$$

В выражении для вычисления слагаемых выделим следующие части:

$$a = (-1)^{2i+1};\ b = x^{3-2i};\ c = (i+3)!.$$

Таблица 2.1 – Нахождение первых значений a, b и c:

i	a	b	c
1	-1	x	$4! = 1 \cdot 2 \cdot 3 \cdot 4$
2	-1	x^{-1}	$5! = 1 \cdot 2 \cdot 3 \cdot 4 \cdot 5$
3	-1	x^{-3}	$6! = 1 \cdot 2 \cdot 3 \cdot 4 \cdot 5 \cdot 6$

Можно заметить, что $a_i = -1$, а текущие значения b и c связаны с предыдущими следующим образом:

$$b_i = \frac{b_{i-1}}{x^2};\ c_i = c_{i-1} \cdot (i+3).$$

```
#include <iostream>
#include <windows.h>
#include <cmath>

using namespace std;

int main()
```

```
{
    int i = 1;
    double a, b, c, x, s, eps, sum = 0;
    SetConsoleOutputCP(1251);
    cout << "Введите x и точность eps: ";
    cin >> x >> eps;
    a = -1;
    b = x;
    c = 2 * 3 * 4;
    s = a * i * b / c;
    while (fabs(s) > eps)
    {
        sum += s;
        i++;
        b /= x * x;
        c *= i + 3;
        s = a * i * b / c;
    }
    cout << "sum=" << sum;
    return 0;
}
```

```
Введите x и точность eps: 2 0.0001
sum=-0.0921875
```

Рисунок 2.7 – Результат выполнения программы (пример 9)

ЛАБОРАТОРНАЯ РАБОТА № 4. РАЗВЕТВЛЯЮЩИЕСЯ АЛГОРИТМЫ

Цель работы: освоить применение условного оператора if-else и оператора выбора switch-case для реализации разветвляющихся вычислительных процессов. Научиться анализировать условия и строить соответствующие им алгоритмы.

Порядок выполнения работы

Написать программу решения задачи согласно варианту:

1. Элементы окружности пронумерованы следующим образом: 1 – радиус *R*; 2 – диаметр *D*; 3 – длина *L*; 4 – площадь круга *S*. Дан номер одного из этих элементов и его значение. Вывести значения остальных элементов данной окружности (в том же порядке).

2. В пункте быстрого питания продают гамбургеры по *N руб.* и мороженое по *M руб.* Составить программу, которая сообщает «Возьмите сдачу», «Доплатите ещё» или «Спасибо за покупку» при оплате покупателем за *K* гамбургеров и *P* мороженых денег в размере *S руб.*

3. Даны целочисленные координаты точки на плоскости. Если точка совпадает с началом координат, то вывести 0. Если точка не совпадает с началом координат, но лежит на оси *OX* или *OY*, то вывести соответственно 1

или 2. Если точка не лежит на координатных осях, то вывести 3.

4. Даны три переменные вещественного типа: *A, B, C*. Вывести их значения в порядке возрастания.

5. Даны координаты точки, не лежащей на координатных осях *OX* и *OY*. Определить номер координатной четверти, в которой находится данная точка.

6. Дано целое число. Вывести его строку-описание вида «отрицательное четное число», «нулевое число», «положительное нечетное число» и т. д.

7. Арифметические действия над числами пронумерованы следующим образом: 1 – сложение, 2 – вычитание, 3 – умножение, 4 – деление. Дан номер действия *N* (целое число в диапазоне 1 – 4) и вещественные числа *A* и *B* (*B* не равно 0). Выполнить над числами указанное действие и вывести результат.

8. Элементы равнобедренного прямоугольного треугольника пронумерованы следующим образом: 1 – катет a; 2 – гипотенуза $c = \sqrt{2}\,a$; 3 – высота h, опущенная на гипотенузу ($h = c / 2$); 4 – площадь $S = c \cdot h / 2$. Дан номер одного из этих элементов и его значение. Вывести значения остальных элементов данного треугольника (в том же порядке).

9. Единицы длины пронумерованы следующим образом: 1 – дециметр, 2 – километр, 3 – метр, 4 – миллиметр, 5 – сантиметр. Дан номер единицы длины (целое число в диапазоне $1 – 5$) и длина отрезка в этих единицах (вещественное число). Найти длину отрезка в метрах.

10. Типы соединения резисторов пронумерованы следующим образом: 1 – последовательное; 2 – параллельное. Дан номер соединения и сопротивления двух резисторов R_1 и R_2. Найти эквивалентное сопротивление этих резисторов.

11. Даны целые числа *a, b, c*, являющиеся сторонами некоторого треугольника. Вывести на экран заключение о том, каким является этот треугольник (равносторонним, равнобедренным или обычным).

12. Даны целые числа *a, b, c*. Вывести на экран 1, если каждое из них отлично от другого, 2 – если только два числа одинаковы и 3 – если одинаковы все три числа.

13. В лавке реализуются два товара. Цена первого p_1 *руб*., второго – p_2 *руб*. Покупатель купил n_1 единиц первого товара и n_2 единиц – второго. Написать программу вычисления стоимости покупки с учетом скидки. Скидка в 3 % предоставляется, если сумма покупки больше 100 *руб*., в 5 % – если сумма покупки больше 200 *руб*.

14. Локатор ориентирован на одну из сторон света (1 – север, 2 – запад, 3 – юг, 4 – восток) и может принимать три цифровые команды поворота: 1 – поворот налево, 1 – поворот направо, 2 – поворот на 180°. Дан символ С – исходная ориентация локатора и целые числа N_1 и N_2 – две посланные команды. Вывести ориентацию локатора после выполнения этих команд.

15. Дан номер года (положительное целое число). Определить количество дней в этом году, учитывая, что обычный год насчитывает 365 дней, а високосный – 366 дней. Високосным считается год, делящийся на 4,

за исключением тех годов, которые делятся на 100 и не делятся на 400 (например, годы 300, 1300 и 1900 не являются високосными, а 1200 и 2000 – являются).

16. Дан возраст человека (от 1 до 100 лет). Вывести его вместе с последующим словом "год", "года" или "лет".

17. Даны три числа. Определить, могут ли эти числа быть длинами сторон треугольника.

18. Дано целое число в диапазоне от 1 до 31 – день мая. Напечатать на экране, является ли этот день мая праздником (и каким), субботой, воскресеньем или рабочим днем. Каким днем недели является 1 мая задать с консоли.

19. У одного брата имеется в копилке m_1 рублей, у второго – m_2 рублей. Они решили разделить деньги поровну, чтобы была возможность купить одинаковые велосипеды. Вывести, кто из них кому и какую сумму должен дать, чтобы денег стало поровну.

20. Задан возраст человека. Вывести к какой возрастной группе он относится: дошкольник, ученик, студент, работник, пенсионер. Возраст человека вводится с клавиатуры.

Контрольные вопросы

1. В чем разница между операторами if, if-else и switch? В каких случаях предпочтительнее использовать каждый из них?
2. Что такое составной оператор и когда он применяется?
3. Для чего используется оператор break внутри switch? Что произойдет, если его опустить?
4. Как проверить, принадлежит ли число диапазону от A до B (включительно или нет)? Напишите условие на C++.
5. В чем разница между операторами && и &, || и |?

ЛАБОРАТОРНАЯ РАБОТА № 5. ТАБУЛИРОВАННЫЕ ФУНКЦИИ

Цель работы: научиться организовывать процесс табулирования функции, то есть вычисления ее значений на заданном интервале с определенным шагом. Получить навыки использования циклов и условных операторов для работы с кусочно-заданными функциями.

Порядок выполнения работы

Написать программу вычисления таблицы значений функции *f(x)* на промежутке от *a* до *b* с шагом *h*.

1. $f(x)=\begin{cases}-x, x\le 0\\ x^3, 0<x\le 1\\ x, x>1\end{cases}$

2. $f(x)=\begin{cases}\ln\left(1+x^2\right), x\le e\\ x, e<x\le 9\\ 9e^{8-x}, x>9\end{cases}$

3. $f(x)=\begin{cases}-1, x\le 0\\ 3x^2, 0<x\le 1\\ 1, x>1\end{cases}$

4. $f(x)=\begin{cases}(x+1)^4, x\le -1\\ 1-\cos(\pi x), -1<x\le 1\\ -(x-1)^2, x>1\end{cases}$

5. $f(x)=\begin{cases}|x|, x\le 1\\ \cos\pi x, 1<x\le 2\\ (2-x)^3, x>2\end{cases}$

6. $f(x)=\begin{cases}-x, x\le -1\\ x^3, -1<x\le 1\\ e^{1-x}, x>1\end{cases}$

7. $f(x)=\begin{cases}|x|^x, x\le 0\\ \ln(x+1), 0<x\le 1\\ x, x>1\end{cases}$

8. $f(x)=\begin{cases}e^x, x\le 0\\ \sin\pi x, 0<x\le 1\\ x-1, x>1\end{cases}$

9. $f(x)=\begin{cases}\sqrt{x+1}, x\le 1\\ \dfrac{x^x}{9}, 1<x\le 3\\ (x-4)^2, x>3\end{cases}$

10. $f(x)=\begin{cases}x^2, x\le 1\\ \dfrac{\pi}{2}-\mathrm{x}, 1<x\le \dfrac{\pi}{2}\\ \sin(x), x>\dfrac{\pi}{2}\end{cases}$

11. $f(x)=\begin{cases}2^{x-1}, x\le 0\\ \sin\dfrac{\pi x}{2}, 0<x\le 1\\ x+1, x>1\end{cases}$

12. $f(x)=\begin{cases}x\sin(x), x\le 0\\ \sqrt{x}, 0<x\le 1\\ \cos(\pi x), x>1\end{cases}$

13. $f(x)=\begin{cases}\sqrt{|x|}, x\le 1\\ 1, 1<x\le 3\\ (x-4)^2, x>3\end{cases}$

14. $f(x)=\begin{cases}x, x\le 2\\ \ln(x), 2<x\le 4\\ \sin\left(\dfrac{\pi x}{2}\right), x>4\end{cases}$

15. $f(x)=\begin{cases}x^2, x\le 0\\ \dfrac{x^3}{3}, 0<x\le 1\\ x, x>1\end{cases}$

16. $f(x)=\begin{cases}2^x, x\le 0\\ \sin(2x), 0<x\le 4\\ (1+x)^3, x>4\end{cases}$

17. $f(x)=\begin{cases}\ln\left(1+x^2\right), x\le -1\\ \cos(x), -1<x\le 1\\ x^2, x>1\end{cases}$

18. $f(x)=\begin{cases}\dfrac{1}{x}, x\le e\\ 1, e<x\le 9\\ -e^{8-x}, x>9\end{cases}$

19. $f(x)=\begin{cases}x\ln|\cos(x)|, x\le 0\\ 1+x^2, 0<x\le 2,5\\ x-2, x>2,5\end{cases}$

20. $f(x)=\begin{cases}1-x^2, x\le 0\\ x+1, 0<x\le 4\\ 1+\sqrt{\cos(x)}, x>4\end{cases}$

Контрольные вопросы

1. Что такое табулирование функции?
2. Какой оператор цикла наиболее удобен для табулирования и почему?
3. Как реализовать вычисление значения для кусочно-заданной функции?
4. Каким образом обеспечить корректный вывод таблицы значений на экран?
5. Какая логическая ошибка может произойти при использовании конструкции if (a < b < c)? Как правильно проверить, что число лежит в интервале?

ЛАБОРАТОРНАЯ РАБОТА № 6. ЦИКЛИЧЕСКИЕ АЛГОРИТМЫ

Цель работы: научиться использовать цикл for для вычисления сумм и произведений конечных числовых последовательностей. Освоить приемы накопления суммы и произведения в цикле.

Порядок выполнения работы

Написать программу вычисления значения суммы либо произведения согласно варианту. В программе использовать цикл **`for`**.

1. $S=\sum_{i=1}^{10}\frac{x^2}{4i^2-1}$
2. $P=\prod_{i=1}^{10}\frac{x^i}{i(i+1)}$
3. $S=\sum_{i=1}^{10}\frac{x^i}{i}$
4. $P=\prod_{i=2}^{15}\frac{x^{i-1}}{i^3-1}$
5. $S=\sum_{i=0}^{10}\frac{x^i}{4^i}$
6. $P=\prod_{i=1}^{10}\frac{x+i}{i}$
7. $S=\sum_{i=3}^{8}\frac{\sin^2 x^2}{i^4}$
8. $P=\prod_{i=7}^{14}\frac{i+x}{x}$
9. $S=\sum_{i=11}^{20}\frac{x-i}{i^2}$
10. $P=\prod_{i=5}^{15}\frac{1}{\cos^2(ix)}$
11. $S=\sum_{i=1}^{10}\frac{i^2}{x+i}$
12. $P=\prod_{i=1}^{10}\frac{ix}{i+x}$
13. $S=\sum_{i=1}^{5}\frac{i+x}{\sin i}$
14. $P=\prod_{i=1}^{6}\frac{\sin^2(ix)+x}{\cos i^2}$
15. $S=\sum_{i=-2}^{2}\frac{5i-x^2}{i^2+1}$
16. $P=\prod_{i=1}^{5}\left(i+\frac{\sin(x)}{i}\right)$
17. $S=\sum_{i=1}^{10}\left(\frac{i}{x-2}\right)^{2-x}$
18. $P=\prod_{i=1}^{7}\frac{x-2^i}{x-2^i+1}$
19. $S=\sum_{i=1}^{8}\frac{x^3+i}{\cos(i^2)}$
20. $P=\prod_{i=1}^{10}\frac{2x+1}{4x^2+1}$

Контрольные вопросы

1. Опишите синтаксис оператора цикла for.
2. Какие начальные значения должны быть присвоены переменной – сумматору и переменной – накопителю произведения перед началом цикла?
3. Как задается приращение переменной – параметру цикла?
4. В чем заключается прием накопления суммы?
5. Когда предпочтительнее использовать цикл for?

ЛАБОРАТОРНАЯ РАБОТА № 7. ВЫЧИСЛЕНИЯ С ЗАДАННОЙ ТОЧНОСТЬЮ

Цель работы: научиться вычислять суммы бесконечных рядов с заданной точностью, используя циклы while или do while. Освоить рекуррентный метод вычисления очередного члена ряда для оптимизации алгоритма.

Порядок выполнения работы

Написать программу вычисления суммы элементов бесконечного ряда с заданной точностью *eps*. Использовать цикл **`while`** или **`do while`**. Результат проверить, вычислив значение функции *Y*(*x*) (S является разложением данной функции в ряд Маклорена). Использовать рекуррентный способ вычисления суммы.

Таблица 2.2 – Варианты заданий

Вариант	Ряд *S*(*x*)	*Y*(*x*)	Вариант	Ряд *S*(*x*)	*Y*(*x*)
1.	$\sum_{n=0}^{\infty}\frac{(x+3)^{2n}}{(2n)!}$	$ch(x+3)$	2.	$\sum_{n=1}^{\infty}(-1)^{n-1}\frac{x^{4n-2}}{(2n-1)!}$	$\sin(x^2)$
3.	$\sum_{n=0}^{\infty}\frac{3^n x^n}{n!}$	e^{3x}	4.	$\sum_{n=1}^{\infty}\frac{(-4)^n x^{2n}}{(2n)!}$	$\cos(2x)-1$
5.	$\sum_{n=1}^{\infty}\frac{x^{n+1}}{(n+1)!}$	e^x-x-1	6.	$\sum_{n=1}^{\infty}(-1)^{n-1}\frac{2^{2n-2}x^{2n-1}}{(2n-1)!}$	$\sin(x)\cos(x)$
7.	$\sum_{n=1}^{\infty}(-1)^n\frac{x^n}{(n+1)!}$	$\frac{1-x-e^{-x}}{x}$	8.	$\sum_{n=1}^{\infty}(-1)^{n-1}\frac{3^{2n-1}x^{2n-2}}{(2n-1)!}$	$\frac{\sin(3x)}{x}$
9.	$\sum_{n=1}^{\infty}\frac{x^{2n-1}}{(2n-1)!}$	$sh(x)$	10.	$\sum_{n=0}^{\infty}(-1)^n\frac{x^{2n}\ln^n(2)}{n!}$	2^{-x^2}
11.	$\sum_{n=0}^{\infty}(-1)^n\frac{2^{4n}x^{2n+1}}{(2n)!}$	$x\cos(4x)$	12.	$\sum_{n=0}^{\infty}\frac{\left(\frac{x}{2}-1\right)^n\ln^n(3)}{n!}$	$3^{\frac{x}{2}-1}$
13.	$\sum_{n=0}^{\infty}(-1)^n\frac{(5x)^{2n}}{(2n)!}$	$\cos(5x)$	14.	$\sum_{n=0}^{\infty}\frac{(x\sin(x))^n}{n!}$	$e^{x\sin(x)}$
15.	$\sum_{n=1}^{\infty}(-1)^n\frac{x^n}{2^n n!}$	$e^{-\frac{x}{2}}-1$	16.	$\sum_{n=1}^{\infty}(-1)^n\frac{x^{n+1}}{(2n)!}$	$x\left(\cos\left(\sqrt{x}\right)-1\right.$
17.	$\sum_{n=0}^{\infty}(-1)^n\frac{(7x)^{2n+1}}{(2n+1)!}$	$\sin(7x)$	18.	$\sum_{n=0}^{\infty}\frac{2n+1}{n!}x^{2n}$	$(1+2x^2)e^{x^2}$
19.	$\sum_{n=0}^{\infty}(-1)^n\frac{x^{2n}}{2^n n!}$	$e^{-\frac{x^2}{2}}$	20.	$\sum_{n=0}^{\infty}\frac{(n^2+1)\left(\frac{x}{2}\right)^n}{n!}$	$e^{\frac{x}{2}}\left(\frac{x^2}{4}+\frac{x}{2}+1\right.$

Контрольные вопросы

1. Что такое точность вычисления (epsilon) и как она используется в условии останова цикла?
2. Почему для вычисления сумм бесконечных рядов часто используется

цикл с предусловием?

3. В чем суть рекуррентного способа вычисления очередного члена ряда?

4. Как избежать зацикливания программы при вычислениях с заданной точностью?

5. Как проверить корректность работы программы, сравнивая результат со значением стандартной функции?

3 МАССИВЫ В ЯЗЫКЕ С++

Массив – это структурированный тип данных, состоящий из фиксированного числа элементов одного типа, объединенных единым именем.

3.1 Статические массивы

Одномерный статический массив описывается следующим образом:

```
тип имя[n];
```

`n` – количество элементов в массиве, которое может быть задано числом либо массивом.

Например:

```
int A[6];
const int n = 15;
double B[n];
```

Доступ к каждому элементу массива осуществляется с помощью индекса – порядкового номера элемента. Для обращения к элементу массива указывают его имя, а затем в квадратных скобках индекс.

Индексация элементов в массивах начинается с 0!

Можно инициализировать массив при объявлении. Например:

```
int a[6] = { 1, 4, -8, 3, 0, 6 };
```

или

```
int a[] = { 1, 4, -8, 3, 0, 6 };
```

Если при инициализации указать меньше значений, чем размер массива, последние элементы заполнятся нулями.

ПРИМЕР. Написать программу консольного ввода и вывода массива из 5 элементов.

```
#include <iostream>
#include <windows.h>
#define N 5

using namespace std;
```

```
int main()
{
    SetConsoleOutputCP(1251);
    int i, a[N];

    // ввод массива
    for (i = 0; i < N; i++)
    {
        cout << "a[" << i << "]=";
        cin >> a[i];
    }

    // вывод массива
    cout << "Массив A: [";
    for (i = 0; i < N; i++)
        cout << a[i] << ", ";
    cout << "\b\b]" << endl;
    return 0;
}
```

```
a[0]=1
a[1]=-7
a[2]=13
a[3]=5
a[4]=0
Массив A: [1, -7, 13, 5, 0]
```

Рисунок 3.1 – Результат выполнения программы (пример 10)

ПРИМЕР. Задан массив из 10 случайных чисел в диапазоне от 0 до 20. Написать программу поиска индекса и значения максимального элемента.

```
#include <iostream>
#include <windows.h>
#include <ctime>
#define N 10

using namespace std;

int main()
{
    SetConsoleOutputCP(1251);
    int i, a[N], i_max = 0;

    srand(time(NULL));
    for (i = 0; i < N; i++)
        a[i] = rand() % 20;

    cout << "Массив A: [";
    for (i = 0; i < N; i++)
        cout << a[i] << ", ";
```

```
    cout << "\b\b]" << endl;

    for (i = 1; i < N; i++)
        if (a[i] > a[i_max])
            i_max = i;

    cout << "Максимальный элемент a[" << i_max << "]=" << a[i_max];
    return 0;
}
```

```
Массив A: [11, 8, 17, 10, 12, 19, 3, 1, 13, 11]
Максимальный элемент a[5]=19
```

Рисунок 3.2 – Результат выполнения программы (пример 11)

Двумерный статический массив описывается следующим образом: `тип имя[m][n];` `m` – количество строк в массиве; `n` – количество столбцов в массиве.

Например:

```
int a[2][3];
```

Пример инициализации двумерного массива:

```
int a[2][3] = { {1, 2, 3}, {4, 5, 6} };
```

ПРИМЕР. Написать программу вычисления в двумерном массиве размерности 3 х 4 суммы элементов, кратных 4.

```
#include <iostream>
#include <windows.h>
#include <ctime>
#define M 3
#define N 4

using namespace std;

int main()
{
    SetConsoleOutputCP(1251);
    int i, j, a[M][N], s = 0;

    // заполнение массива
    srand(time(NULL));
    for (i = 0; i < M; i++)
        for (j = 0; j < N; j++)
        {
            a[i][j] = rand() % 21;
        }

    // вывод массива
    cout << "Матрица A:" << endl;
```

```
    for (i = 0; i < M; i++)
    {
        for (j = 0; j < N; j++)
        {
            cout.width(3);
            cout << a[i][j];
        }
        cout << endl;
    }

    for (i = 0; i < M; i++)
        for (j = 0; j < N; j++)
            if (a[i][j] % 5 == 0)
                s += a[i][j];
    cout << "Сумма элементов кратных 5 равна " << s;

    return 0;
}
```

```
Матрица A:
  1  0  3  5
 20 12 19  1
  8  0  3 13
Сумма элементов кратных 5 равна 25
```

Рисунок 3.3 – Результат выполнения программы (пример 12)

3.2 Динамические массивы

Для создания динамического массива необходимо:

- описать указатель (`тип *имя_указателя;`);
- определить размер массива;
- выделить участок памяти для хранения массива и присвоить указателю адрес этого участка памяти.

Наиболее предпочтительным подходом к динамическому распределению памяти является использование операций языка C++ `new` и `delete`.

Операция `new` выделяет память из области свободной памяти, а операция `delete` высвобождает выделенную память.

Минимальный набор действий, необходимых для динамического размещения одномерного массива действительных чисел размером n:

```
int n;
double* a;
. . .
a = new double[n]; // Захват памяти для n элементов
. . .
delete[]a;    // Освобождение памяти
```

Минимальный набор действий, необходимых для динамического размещения двухмерного массива действительных чисел размером $m \times n$:

```
int i, m, n;  // m, n – размеры массива
double** a;
. . .
a = new double* [m];   // Захват памяти под указатели
for (i = 0; i < m; i++)
    a[i] = new double[n]; // и под элементы
. . .
for (i = 0; i < m; i++)
    delete[]a[i]; // Освобождение памяти
delete[]a;
```

ПРИМЕР. Задана целочисленная матрица A размером $m \times n$. Получить вектор B, присвоив его j-му элементу значение 1, если все элементы j-го столбца матрицы положительные, 1 – если все элементы j-го столбца матрицы отрицательные, и 0 – в остальных случаях.

```
#include <iostream>
#include <windows.h>
#include <ctime>

using namespace std;

int main()
{
    SetConsoleOutputCP(1251);
    int i, j, m, n, **A, *B;

    cout << "Введите m и n: ";
    cin >> m >> n;

    A = new int* [m];
    for (i = 0; i < m; i++)
        A[i] = new int[n];

    srand(time(NULL));
    cout << "Матрица A:" << endl;
    for (i = 0; i < m; i++)
    {
        for (j = 0; j < n; j++)
        {
            A[i][j] = rand() % 11 - 5;
            cout << A[i][j] << "\t";
        }
        cout << endl;
    }

    B = new int[n];

    cout << "Вектор B:" << endl;
    for (j = 0; j < n; j++)
```

```
    {
        bool allPositive = true;
        bool allNegative = true;
        for (i = 0; i < m; i++)
        {
            if (A[i][j] <= 0)
                allPositive = false;
            if (A[i][j] >= 0)
                allNegative = false;
        }
        if (allPositive)
            B[j] = 1;
        else if (allNegative)
            B[j] = -1;
        else
            B[j] = 0;
        cout << B[j] << "\t";
    }
    for (i = 0; i < m; i++)
        delete[]A[i];
    delete[]A;
    delete[]B;

    return 0;
}
```

```
Введите m и n: 2 10
Матрица A:
0       1       -2      -3      5       -4      0       -5      5       4
-2      -5      5       -5      -1      3       2       1       4       2
Вектор B:
0       0       0       -1      0       0       0       0       1       1
```

Рисунок 3.4 – Результат выполнения программы (пример 13)

ЛАБОРАТОРНАЯ РАБОТА № 8. ОДНОМЕРНЫЕ МАССИВЫ

Цель работы: Приобрести навыки работы с одномерными массивами: объявление, инициализация, ввод/вывод, обработка (поиск элементов, удовлетворяющих условию, вычисление сумм, произведений и т.д.). Научиться использовать циклы для перебора элементов массива.

Порядок выполнения работы

Написать программу вычисления в одномерном целочисленном массиве (элементы массива генерируются случайным образом):

1. Произведения элементов массива, расположенных между максимальным и минимальным элементами.
2. Суммы элементов массива, расположенных между первым и

последним нулевыми элементами.

3. Суммы элементов массива, расположенных до последнего положительного элемента.

4. Суммы элементов массива, расположенных между первым и последним положительными элементами.

5. Произведения элементов массива, расположенных между первым и вторым нулевыми элементами.

6. Суммы элементов массива, расположенных между первым и вторым отрицательными элементами.

7. Количества отрицательных элементов, расположенных между максимальным и минимальным элементами.

8. Суммы элементов массива, расположенных после последнего отрицательного элемента.

9. Суммы элементов массива, расположенных после последнего элемента, равного нулю.

10. Номера элемента с наименьшим отклонением от среднего арифметического всех элементов.

11. Суммы элементов, превосходящих среднее арифметическое модулей всех элементов.

12. Количества четных элементов между минимальным и максимальным элементами.

13. Количества положительных элементов между первым и последним нулями.

14. Произведения элементов после последнего нулевого элемента.

15. Количества нечетных элементов между первым и последним положительным элементом.

16. Произведения элементов, расположенных до первого отрицательного элемента.

17. Количества элементов, расположенных между первым и вторым отрицательными элементами.

18. Произведения положительных элементов между первым и последним нулем.

19. Количества четных элементов, расположенных после последнего отрицательного элемента.

20. Суммы элементов, кратных 5 и расположенных между первым и вторым отрицательным элементами.

Контрольные вопросы

1. Как объявить и инициализировать одномерный массив в C++?
2. Каким образом осуществляется доступ к элементам массива?
3. Как найти максимальный (минимальный) элемент в массиве и его индекс?
4. Как организовать поиск в массиве элементов, удовлетворяющих определенному условию?
5. Как организовать цикл для обработки массива «с конца к началу»?

ЛАБОРАТОРНАЯ РАБОТА № 9. ДВУМЕРНЫЕ МАССИВЫ

Цель работы: научиться работать с двумерными массивами (матрицами): (создание, заполнение, вывод на экран). Освоить алгоритмы обработки матриц, включая анализ строк и столбцов, работу с диагоналями.

Порядок выполнения работы

Дана целочисленная прямоугольная матрица. Написать программу решения задачи согласно варианту:

1. Определить количество строк, не содержащих ни одного нулевого элемента.
2. Определить количество столбцов, содержащих, по крайней мере, один нулевой элемент.
3. Определить произведение элементов в тех строках, которые не содержат отрицательных элементов.
4. Определить сумму элементов в тех строках, которые содержат отрицательные элементы.
5. Вывести на экран номер первого столбца, содержащего нулевой элемент.
6. Найти номер первого столбца, не содержащего ни одного отрицательного элемента.
7. Найти количество столбцов, среднее арифметическое элементов которых меньше заданной величины.
8. Определить сумму элементов строк, не содержащих ни одного нулевого элемента.
9. Определить сумму элементов столбцов, содержащих по крайней мере один нулевой элемент.
10. Определить сумму положительных элементов, лежащих выше главной диагонали.
11. Определить сумму отрицательных элементов, лежащих ниже главной диагонали.
12. Найти номер последнего из ее столбцов, содержащих только положительные элементы
13. Найти номер последней из ее строк, содержащих нулевой элемент.
14. Найти количество ее столбцов, которые состоят только из чётных элементов.
15. Поменять местами главную диагональ и первый столбец.
16. Определить количество столбцов, не содержащих отрицательных элементов.
17. Определить произведение элементов в сроках, не содержащих нули.
18. Найти количество строк, содержащих один нулевой элемент.
19. Найти количество строк, содержащих, по крайне мере, один четный элемент.
20. Определить среднее арифметическое строк, не содержащих ноль.

Контрольные вопросы

1. Как объявить двумерный массив в C++?
2. Как с помощью вложенных циклов организовать перебор всех элементов матрицы?
3. Как определить главную диагональ матрицы? Каково условие для ее элементов (индексы i и j)?
4. Как заполнить матрицу случайными числами в заданном диапазоне? Как заполнить её по определенному правилу (например, единицами на главной диагонали)?
5. Опишите алгоритм проверки, что все элементы столбца удовлетворяют условию.

ЛАБОРАТОРНАЯ РАБОТА № 10. ДИНАМИЧЕСКИЕ МАССИВЫ

Цель работы: освоить технику динамического выделения и освобождения памяти для одномерных и двумерных массивов с помощью операторов new и delete. Понять преимущества и недостатки динамических массивов по сравнению со статическими.

Порядок выполнения работы

Написать программу решения задачи согласно варианту. Использовать динамическое размещение данных. Значения элементов матрицы генерируются случайным образом.

1. Дана матрица A, размером $m \times n$. Заполнить одномерный массив, найдя сумму положительных элементов в каждом столбце матрицы.
2. Дана матрица A, размером $m \times n$. Заполнить одномерный массив, найдя произведение положительных элементов в каждом столбце матрицы.
3. Дана матрица A, размером $m \times n$. Найти минимальный элемент в каждой строке матрицы. Затем каждую строку матрицы разделить на минимальный элемент строки.
4. Дана матрица A, размером $m \times n$. Заполнить одномерный массив, найдя минимальный среди положительных элементов в каждой строке матрицы.
5. Дана матрица A, размером $m \times n$. Заполнить одномерный массив, найдя количество положительных элементов в каждом столбце матрицы.
6. Дана матрица A, размером $m \times n$. Заполнить одномерный массив, найдя минимальный среди отрицательных элементов в каждой строке матрицы.
7. Дана матрица A, размером $m \times n$. Заполнить одномерный массив, найдя среднее арифметическое положительных элементов в каждом столбце матрицы.
8. Дана матрица A, размером $m \times n$. Найти минимальный элемент в каждой строке матрицы. Затем каждую строку матрицы умножить на

минимальный элемент строки.

9. Дана матрица *A*, размером *m*×*n*. Заполнить одномерный массив, найдя среднее геометрическое положительных элементов в каждом столбце матрицы.

10. Дана матрица *A*, размером *m*×*n*. Найти минимальный элемент в каждой строке матрицы. Затем к каждому элементу каждой строки прибавить минимальный элемент строки.

11. Дана матрица *A*, размером *m*×*n*. Найти минимальный элемент и его номер в каждой строке матрицы. Затем из каждого элемента каждой строки вычесть номер минимального элемента строки.

12. Дана матрица *A*, размером *m*×*n*. Заполнить одномерный массив, найдя сумму отрицательных элементов в каждом столбце матрицы.

13. Дана матрица *A*, размером *m*×*n*. Найти номер минимального элемента в каждой строке матрицы. Затем к каждому элементу каждой строки прибавить номер минимального элемента строки.

14. Дана матрица *A*, размером *m*×*n*. Заполнить одномерный массив, найдя произведение отрицательных элементов в каждом столбце матрицы.

15. Дана матрица *A*, размером *m*×*n*. Заполнить одномерный массив, найдя количество отрицательных элементов в каждом столбце матрицы.

16. Дана матрица *A*, размером *m*×*n*. Заполнить одномерный массив, найдя в каждом столбце количество элементов меньших среднего арифметического в данном столбце.

17. Дана матрица *A*, размером *m*×*n*. Заполнить одномерный массив, найдя номер максимального элемента в каждой строке матрицы.

18. Дана матрица *A*, размером *m*×*n*. Заполнить одномерный массив, найдя сумму отрицательных элементов в каждой строке.

19. Дана матрица *A*, размером *m*×*n*. Заполнить одномерный массив, найдя в каждой строке количество элементов меньших среднего арифметического в данной строке.

20. Дана матрица *A*, размером *m*×*n*. Найти максимальный элемент в каждой строке матрицы. Затем каждую строку матрицы разделить на максимальный элемент строки.

Контрольные вопросы

1. В чем основное отличие статических и динамических массивов?
2. Опишите последовательность действий для создания динамического двумерного массива.
3. Почему необходимо явно освобождать память, выделенную под динамический массив?
4. В каких случаях предпочтительнее использовать статические массивы, а в каких – динамические?
5. Что произойдет, если не освободить память после использования динамического массива?

4 ФУНКЦИИ В ЯЗЫКЕ С++

4.1 Общие сведения о функциях

Подпрограмма – именованная, логически законченная группа операторов языка, которую можно вызвать для выполнения любое количество раз из различных мест программы. В языке С++ подпрограммы реализованы в виде функций.

Как известно, программа на С++ состоит из одной или нескольких функций. Функция должна быть описана перед своим использованием.

Описание функции состоит из заголовка и тела функции.

```
тип имя_функции([список_параметров]) // заголовок функции
{
    операторы; // тело функции
}
```

`Тип` функции показывает, какое значение возвращает функция: целое, вещественное, строковое и так далее.

`список_параметров` содержит перечень типов и имен величин, передаваемых в функцию и разделенных запятыми. Если функция не имеет параметров, то скобки все равно необходимы, хотя в них ничего не указывается.

В случае, если вызываемые функции определены после функции `main` *или даже в другом файле*, необходимо до первого вызова функции (перед функцией `main`) сообщить программе тип возвращаемого функцией значения, а также количество и типы ее параметров. Подобное сообщение называется *прототипом функции*.

Прототип может полностью совпадать с заголовком функции, хотя при объявлении функции компилятору необходимо знать только имя функции, количество аргументов, их типы и тип возвращаемого функцией значения. Поэтому имена параметров необязательно указывать в прототипах функций.

Если возврата значения функцией не требуется, то тип функции – `void`.

Возврат значений функцией выполняется с помощью оператора `return`:

```
return значение;
```

Если тип функции – `void`, то оператор возврата просто – `return;` или его можно даже опустить в теле функции.

`Return` вызывает немедленный выход из функции и возврат в вызвавшую ее подпрограмму.

Функция возвращает единственное скалярное значение. Это может быть число, адрес, структура.

Для вызова функции в программе пишется имя функции, за которым в скобках следует список аргументов, которые являются переменными и константами, определенными в программе. При этом происходит выполнение тела функции, в котором параметры заменяются значениями аргументов. Тип аргумента должен соответствовать типу соответствующего параметра или

допускать приведение к типу параметра.

ПРИМЕР. Написать программу для вычисления расстояния между двумя точками по их координатам.

```
#include <iostream>
#include <cmath>

using namespace std;

double dist(double x_1, double y_1, double x_2, double y_2)
{
    return sqrt(pow(x_2 - x_1, 2) + pow(y_2 - y_1, 2));
}

int main()
{
    double x1, x2, y1, y2, d;
    cout << "Input x1, y1, x2, y2:" << endl;
    cin >> x1 >> y1 >> x2 >> y2;
    d = dist(x1, y1, x2, y2);
    cout << "dist=" << d << endl;
    return 0;
}
```

```
Input x1, y1, x2, y2:
5 2 2 -2
dist=5
```

Рисунок 4.1 – Результат выполнения программы (пример 14)

ПРИМЕР. Написать программу для вывода таблицы умножения на заданное число.

```
#include <iostream>

using namespace std;

void multab(int);

int main()
{
    int n;
    cout << "Input n:" << endl;
    cin >> n;
    multab(n);
    return 0;
}

void multab(int n)
```

```
{
    for (int i = 1; i <= 10; i++)
        cout << n << "*" << i << "=" << n * i << endl;
}
```

```
Input n:
9
9*1=9
9*2=18
9*3=27
9*4=36
9*5=45
9*6=54
9*7=63
9*8=72
9*9=81
9*10=90
```

Рисунок 4.2 – Результат выполнения программы (пример 15)

4.2 Рекурсия

Рекурсия – вызов функции из неё же самой, непосредственно (простая рекурсия) или через другие функции (сложная или косвенная рекурсия).

ПРИМЕР. Написать программу, использующую рекурсивную функцию, вычисляющую факториал числа.

```
#include <iostream>

using namespace std;

long long int fact(int n)
{
    if (n == 1)
        return n;
    else
        return n * fact(n - 1);
}

int main()
{
    int n;
    cout << "n=";
    cin >> n;
    cout << n << "!=" << fact(n);
    return 0;
}
```

```
n=5
5!=120
```

Рисунок 4.3 – Результат выполнения программы (пример 16)

4.3 Массивы и функции

Массив передаётся в функцию как указатель. Причём неважно, какой это массив: статический или динамический. Также обычно в функцию необходимо передать размер массива (количество элементов), поскольку в общем случае, имея только указатель, невозможно определить размер массива.

ПРИМЕР. Написать программу, которая будет вызывать функцию для вывода на печать элементов одномерного массива.

```
#include <iostream>
#include <windows.h>

using namespace std;

void print(int x[], int n)
{
    cout << "Массив:" << endl;
    for (int i = 0; i < n; i++)
        cout << x[i] << ' ';
}

int main()
{
    SetConsoleOutputCP(1251);
    const int n = 5;
    int x[n] = {3, 5, 1, 7, 4};
    print(x, n);
    return 0;
}
```

```
Массив:
3 5 1 7 4
```

Рисунок 4.4 – Результат выполнения программы (пример 17)

Заголовок может выглядеть и так:

```
void print(int* x, int n)
```

Передача в подпрограмму матрицы в качестве параметра несколько сложнее, чем передача одномерного массива. Когда мы передаем многомерные массивы, необходимо указывать все размерности, за исключением первой, то есть в списке параметров допустима запись:

```
тип_данных имя[][константа]
```

Это приемлемо только для небольших программ с матрицами одинакового размера.

Решением является использование динамического массива. В этом случае в подпрограмму будет передаваться указатель на указатель.

ПРИМЕР. Дана прямоугольная матрица. Написать программу, выводящую на консоль сумму положительных элементов данной матрицы.

```
#include <iostream>
#include <windows.h>

using namespace std;

int sumArr(int** a, int m, int n)
{
    int s = 0;
    for (int i = 0; i < m; i++)
        for (int j = 0; j < n; j++)
            if (a[i][j] > 0)
                s += a[i][j];
    return s;
}

int main()
{
    SetConsoleOutputCP(1251);
    int** a, i, j, m, n;
    cout << "Введите размеры матрицы: ";
    cin >> m >> n;
    a = new int* [m];
    for (i = 0; i < m; i++)
        a[i] = new int[n];

    for (i = 0; i < m; i++)
        for (j = 0; j < n; j++)
            a[i][j] = rand() % 11 - 5;

    for (i = 0; i < m; i++)
    {
        for (j = 0; j < n; j++)
            cout << a[i][j] << '\t';
        cout << endl;
    }

    cout << "Сумма положительных элементов матрицы: " << sumArr(a, m, n);

    for (i = 0; i < m; i++)
        delete[]a[i];
    delete[]a;

    return 0;
}
```

```
Введите размеры матрицы: 3 4
3       4       4       -4
2       0       0       5
-4      -5      2       2
Сумма положительных элементов матрицы: 22
```

Рисунок 4.5 – Результат выполнения программы (пример 18)

ЛАБОРАТОРНАЯ РАБОТА № 11. ФУНКЦИИ

Цель работы: Научиться проектировать и использовать собственные функции для структурирования программы. Освоить механизмы передачи параметров в функцию и возврата значения из функции. Понять принцип рекурсии.

Порядок выполнения работы

Написать программу решения задачи согласно варианту.

1. Описать функцию pairsOfDivisors(N), которая возвращает количество пар в числе *N*, в которых одна цифра из пары делится на другую цифру из пары. Под парой подразумевается две идущие подряд цифры. Например, в числе 1256, парами являются следующие сочетания цифр: 1 и 2, 2 и 5, 5 и 6. Функция также должна вывести сами пары. С помощью описанной функции найти пары в 3 числах.

2. Описать функцию invertDigits(K), меняющую порядок следования цифр целого положительного числа K на обратный. С помощью этой функции поменять порядок следования цифр на обратный для каждого из пяти заданных целых чисел.

3. Описать функцию root(a, n), вычисляющую значение $x = \sqrt{a}$ по формуле $x_n = \frac{1}{2}\left(x_{n-1} + \frac{a}{x_{n-1}} \right)$. В качестве начального приближения использовать значение $x_1 = \frac{1}{2}(1 + a)$. С помощью этой функции вычислить корень для пяти различных a при заданном n.

4. Описать функцию convertTime(T) возвращающую по времени T (в секундах) строку в виде HH:MM:SS, где HH – количество часов, MM – количество минут и SS – количество секунд во временном отрезке. Используя эту функцию, получить результат для пяти заданных отрезков времени. Строки в задаче не использовать.

5. Описать функцию divMaxMin(N), которая возвращает целочисленное деление максимальной цифры и минимальной цифры числа N. Если минимальная цифра равна 0, то функция должна вернуть значение 0. С помощью данной функции найти частное для M целых чисел.

6. Описать функцию isPalindrom(K), возвращающую true, если целый параметр K (>0) является палиндромом (последовательность составляющих его

чисел читается одинаково слева направо и справа налево), и false – в противном случае. С помощью этой функции найти количество палиндромов в наборе из 5 целых положительных чисел.

7. Описать функцию element(x, i, j, n), возвращающую по индексам элемента в матрице его значение. Матрица заполняется следующим образом:

$$\begin{bmatrix} x^n & x^{n-1} & x^{n-2} & \cdots & 1 \\ x^{n-1} & 0 & 0 & \cdots & x \\ x^{n-2} & 0 & 0 & \cdots & x^2 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x & x^2 & \cdots & x^n \end{bmatrix}$$

С использованием этой функции заполнить квадратную матрицу для заданных x и n.

8. Описать функцию even(K), возвращающую новое число, которое состоит только из четных цифр исходного числа K. Получить таким образом пять чисел.

9. Описать функцию convertDigits(K, p), переводящую целое десятичного числа K в систему счисления с основанием p (>1).

10. Описать функцию fibonacci(n), вычисляющую n-ое число Фибоначчи. Числа Фибоначчи определяются следующим образом: $f_1 = 0$, $f_2 = 1$, $f_3 = 1$, …, $f_n = f_{n-1} + f_{n-2}$. С помощью этой функции заполнить массив первыми 15 числами Фибоначчи.

11. Описать функцию isCorrectDate(d, m, y), проверяющую может ли существовать дата d.m.y. С использованием данной функции проверить 4 даты.

12. Описать функцию shortFract(n, d) сокращения простой дроби, которая задаётся двумя целочисленными значениями: числителем n и знаменателем d. Функция должна выводить ответ на экран в виде дроби (например: 1/4, 3/17). С помощью этой функции сократить три заданные дроби.

13. Описать функцию gcd(N, M), определяющую наибольший общий делитель чисел M и N используя метод Эйлера: если M делится на N, то gcd(N, M) = N, иначе gcd(N, M) = gcd(M % N, N). С помощью этой функции найти наибольший общий делитель для четырех различных пар чисел.

14. Описать функцию y(n), вычисляющую значение выражения по формуле:

$$y(n) = \cfrac{1}{n + \cfrac{1}{(n-1) + \cfrac{1}{(n-2) + \cfrac{1}{\ldots}}}}$$

$$\ldots + \cfrac{1}{1 + \cfrac{1}{2}}$$

Используя эту функцию, получить результат для пяти различных n.

15. Описать функцию sumDigits(K, p), вычисляющую сумму цифр,

составляющих заданное целое число. С использованием этой функции получить сумму цифр для пяти заданных положительных чисел.

16. Описать функцию convertDigits(K), переводящую целое арабское число K в римское.

17. Описать функцию, реализующую функцию Аккермана, A(m,n):

$$A(m,n)=\begin{cases} n+1, & при\ m=0 \\ A(m-1,1), & при\ m>0, n=0 \\ A(m-1,A(m,n-1)), & при\ m>0, n>0 \end{cases}$$

Для четырех пар целых неотрицательных числа m и n вывести с использованием разработанной функции значение A(m,n).

18. Описать функцию LastProgressionMember(N, a0, d), определяющую номер последнего члена арифметической прогрессии, не превышающего N (a0 – первый член прогрессии, d – ее разность). С использованием данной функции определить номер последнего члена, не превышающего 250 для 4 разных прогрессий.

19. Десятичное число M эквивалентно числу N в некоторой другой системе счисления. Описать функцию numberSystem(M,N), которая находит основание этой системы или возвращает 0, если такой системы счисления не существует. С использованием данной функции проверить 3 пары чисел. Считать, что основание системы счисления не может быть больше 10.

20. Описать функцию prod(n), вычисляющую произведение n (n > 2, четное) первых множителей следующего выражения:

$$y=\frac{2}{1}\cdot\frac{2}{3}\cdot\frac{4}{3}\cdot\frac{4}{5}\cdot\frac{6}{5}\cdot\frac{6}{7}\cdot\ldots.$$

С использованием этой функции получить значения *y* при пяти различных *n*.

Контрольные вопросы

1. Что такое функция? Из каких частей состоит описание функции?
2. Что такое прототип функции и для чего он нужен?
3. Какие способы передачи параметров в функцию вы знаете?
4. Чем отличается оператор return в функциях типа void и в функциях, возвращающих значение?
5. Что такое рекурсия? Какое условие обязательно должно присутствовать в рекурсивной функции?

ЛАБОРАТОРНАЯ РАБОТА № 12. ФУНКЦИИ И МАССИВЫ

Цель работы: закрепить навыки работы с функциями, научиться передавать массивы (как статические, так и динамические) в функции в качестве параметров. Разработать программу, использующую функции для обработки нескольких массивов.

Порядок выполнения работы

Написать программу решения задачи согласно варианту.

1. Координаты первой группы точек заданы массивом X(n, 2), координаты второй группы точек – массивом Y(m, 2). Для каждой группы точек с использованием функции определить, сколько точек из группы находится внутри окружности радиусом R и с центром в начале координат.

2. Даны два массива X(n) и Y(m). Вычислить

$$Z = \frac{s_1 + s_2}{k_1 \cdot k_2},$$

где s_1 и k_1 – сумма и количество четных элементов массива X(n); s_2 и k_2 – сумма и количество четных элементов массива Y(m).

Сумму четных элементов массива вычислять с помощью одой функции, количество – с помощью другой.

3. Даны две матрицы A(n, m) и B(k, l). Для каждой матрицы с использованием функции вывести на консоль количество отрицательных элементов в каждом столбце.

4. Даны две матрицы A(n, m) и B(k, l). Для каждой матрицы с использованием функции подсчитать среднее арифметическое элементов, принадлежащих отрезку [0, 1].

5. Даны две матрицы A(n, m) и B(k, l). Для каждой матрицы с использованием функции вывести на консоль все элементы кратные 3.

6. Даны два массива X(n) и Y(m). Вычислить

$$Z = \frac{\sum \sin\left(X_i\right) - \sum \sin\left(Y_i\right)}{2}.$$

Сумму синусов элементов вычислять с использованием функции.

7. Даны две матрицы A(m, m) и B(n, n). Для каждой матрицы с использованием функции вычислить сумму отрицательных элементов на главной диагонали.

8. Даны две матрицы A(m, m) и B(n, n). Для каждой матрицы с использованием функции вычислить количество четных элементов под главной диагональю.

9. Даны два массива X(n) и Y(m). С использованием функции вычислить среднеквадратичное отклонение элементов в каждом массиве. Среднеквадратичное отклонение элементов массива определять по формуле:

$$S = \sqrt{\frac{\sum_{i=1}^{n}\left(array_i - \overline{array}\right)^2}{n}}.$$

где $\overline{array}$ – среднее арифметическое значение элементов в массиве.

10. Даны два массива X(n) и Y(m). Вычислить

$$Z = \frac{X_{\max} - Y_{\min}}{2}.$$

$X_{\max}$ и $Y_{\min}$ вычислять с использованием функций.

11. Даны два массива X(n) и Y(m). С использованием функции

вычислить среднее арифметическое положительных элементов в каждом массиве.

12. Даны две целочисленные матрицы A(n, m) и B(k, l). Для каждой матрицы с использованием функции вывести на консоль количество четных элементов в каждой строке.

13. Даны два массива X(n) и Y(m). Для каждого массива с использованием функций вычислить

$$Z = \frac{k_p + k_n}{2},$$

где k_p – количество положительных элементов массива; k_n – количество отрицательных элементов массива.

14. Даны два массива X(n) и Y(m). Для каждого массива с использованием функций вычислить среднее значение и среднее отклонение элементов.

15. Даны две целочисленные матрицы A(n, m) и B(k, l) и число a. Для каждой матрицы с использованием функции подсчитать количество нечетных элементов, принадлежащих отрезку [a, 2a].

16. Даны два массива X(n) и Y(m). Для каждого массива с использованием функций вычислить сумму элементов массива, модуль которых больше заданного числа a и меньшего заданного числа b.

17. Даны две матрицы A(n, m) и B(k, l). Для каждой матрицы с использованием функции вывести на консоль все элементы, модуль которых меньше заданного неотрицательного числа n.

18. Даны два массива X(n) и Y(m). Для каждого массива с использованием функций вычислить и определить номера максимального элемента среди элементов, стоящих на нечетных местах. В главной программе вычислить произведение полученных значений.

19. Даны две матрицы A(m, m) и B(n, n). Для каждой матрицы с использованием функции вычислить произведение нечетных элементов на главной диагонали.

20. Даны две матрицы A(n, m) и B(k, l). Для каждой матрицы с использованием функции вывести на консоль все элементы, кратные заданному числу n (n>1).

Контрольные вопросы

1. Как передать одномерный массив в функцию?
2. Почему при передаче массива в функцию обычно передают и его размер?
3. Можно ли внутри функции изменить элементы переданного массива? Почему?
4. Как передать двумерный динамический массив в функцию?
5. Каковы преимущества использования функций для обработки массивов?

5 ОБРАБОТКА СТРОК В ЯЗЫКЕ С++

В языке С++ для удобной работы со строками есть класс **string**, для использования которого необходимо подключить заголовочный файл `string`.

Строки можно объявлять и одновременно присваивать им значения:

```
string S1, S2 = "Hello";
```

Строка `S1` будет пустой, строка `S2` будет состоять из 5 символов.

К отдельным символам строки можно обращаться по индексу. Например `S[0]` – это первый символ строки.

Строки можно создавать с использованием следующих конструкторов:

`string()` – создает пустую строку;

`string(S)` – копия строки `S`;

`string(n, c)` – повторение символа `c` заданное число `n` раз;

`string(c)` – строка из одного символа `c`;

`string(S, pos, n)` – строка, содержащая `n` символов строки `S`, начиная с символа с индексом `pos`.

Строка выводится точно так же, как и числовые значения:

```
cout << S;
```

Для считывания строки с консоли можно использовать операцию `>>` для объекта `cin`:

```
cin >> S;
```

В этом случае считывается строка до первого пробела или символа конца строки. Это удобно для того, чтобы разбивать текст на слова, или чтобы читать данные до конца файла при помощи `while (cin >> S)`.

Можно считывать строки целиком вместе с пробелами, используя функцию `getline`:

```
getline(cin, S);
```

Со строками можно выполнять следующие арифметические операции:

`=` – присваивание значения.

`+=` – добавление в конец строки другой строки или символа.

`+` – конкатенация двух строк, конкатенация строки и символа.

`==`, `!=` – посимвольное сравнение.

`<`, `>`, `<=`, `>=` – лексикографическое сравнение.

У строк есть разные методы, многие из них можно использовать несколькими разными способами (с разным набором параметров).

Таблица 5.1 – Функции и методы класса **string**

Название	Описание
1	2
`S.size()` `S.length()`	Возвращает длину строки `S`
`S.resize(n)` `S.resize(n, c)`	Изменяет длину строки `S` на `n`. Если задан параметр `c`, то при увеличении длины строки `S` добавляемые символы будут равны `c`

Продолжение таблицы 5.1

1	2
S.at(n)	Обращение к символу строки S с индексом n. Данный метод аналогичен обращению S[n]
S.begin()	Возвращает итератор, указывающий на первый символ строки S
S.end()	Возвращает итератор, указывающий на последний символ строки S
S.front()	Возвращает первый символ строки S
S.back()	Возвращает последний символ строки S
S.assign(n, c) S.assign(str) S.assign(str, pos, n)	Заменяет содержимое строки S n символами c. Заменяет содержимое строки S копией строки str. Заменяет содержимое строки S n символами строки str, начиная с символа с индексом pos
S.append(n, c) S.append(str) S.append(str, pos, n)	Добавляет в конец строки n одинаковых символов, равных c. Добавляет в конец строки S содержимое строки str. Добавляет в конец строки S n символов строки str, начиная с символа с индексом pos
S.insert(i, n, c) S.insert(i, str) S.insert(i, str, pos, n)	Вставляет в строку S n одинаковых символов, равных c. Вставляет содержимое строки str. Вставляет n символов строки str, начиная с символа с индексом pos. Первый вставленный символ будет иметь индекс i, а все символы, которые ранее имели индекс i и более, сдвигаются вправо
S.replace(pos, n, n2, c) S.replace(pos, n, str) S.replace(pos, n, str, pos2, n2)	Заменяет n символов строки S, начиная с символа с индексом pos, на n2 символов c. Заменяет n символов строки S, начиная с символа с индексом pos, на содержимое строки str. Заменяет n символов строки S, начиная с символа с индексом pos, на n2 символов строки str, начиная с символа с индексом pos2
S.swap(str)	Обменивает содержимым строки S и str
S.push_back(c)	Добавляет в конец строки символ c
S.pop_back()	Удаляет последний символ строки
S.clear() S.erase()	Очищает строку, строка становится пустой
S.erase(pos) S.erase(pos, n)	Удаляет символы из строки S, начиная с символа с индексом pos и до конца строки. Удаляет из строки S n символов, начиная с символа с индексом pos

Окончание таблицы 5.1

1	2

`S.erase(pos)`
 	
`S.erase(pos, n)`
	Удаляет символы из строки `S`, начиная с символа с индексом `pos` и до конца строки.
	Удаляет из строки `S` `n` символов, начиная с символа с индексом `pos`

`S.empty()`	Возвращает `true`, если строка пуста, `false` – в противном случае

`S.substr(pos)`
 	
`S.substr(pos, n)`
	Возвращает подстроку данной строки, начиная с символа с индексом `pos` и до конца строки.
	Возвращает подстроку данной строки, начиная с символа с индексом `pos` количеством `n` символов

`S.find(str)` `S.find(str, pos)`	Ищет первое вхождение строки `str` в строку `S`. Ищет первое вхождение строки `str`, начиная с позиции `pos`. Если строка `str` не найдена, то возвращается `-1`
`S.rfind(str)` `S.rfind(str, pos)`	Ищет последнее вхождение строки `str` в строку `S`. Способы вызова аналогичны способам вызова метода `find`.
`S.find_first_of(str)` `S.find_first_of(str, pos)`	Ищет в строке `S` первое появление любого из символов данной строки `str`. Если задано значение `pos`, то поиск начинается с позиции `pos`
`S.find_last_of(str)` `S.find_last_of(str, pos)`	Ищет в строке `S` последнее появление любого из символов данной строки `str`. Если задано значение `pos`, то поиск начинается с позиции `pos`
`S.find_first_not_of(str)` `S.find_first_not_of(str, pos)`	Ищет в данной строке первое появление символа, отличного от символов строки `str`. Если задано значение `pos`, то поиск начинается с позиции `pos`
`S.find_last_not_of(str` `S.find_last_not_of(str, pos)`	Ищет в данной строке последнее появление символа, отличного от символов строки `str`. Если задано значение `pos`, то поиск начинается с позиции `pos`
`S.compare(str)`	Сравнивает две строки. Возвращает `0`, если `S=str`, `1`, если `S>str`, и `–1`, если `S<str`
`int stoi(str)`	Преобразует строку `str` в целочисленное значение
`float stof(str)` `double stod(str)`	Преобразует строку `str` в вещественное значение
`to_string(value)`	Преобразует числовое значение в строку `string`
`S.c_str()`	Возвращает массив символов строки `S`

ПРИМЕР. Рассмотрим применение некоторых из описанных выше методов.

```
#include <iostream>
#include <string>

using namespace std;

int main()
{
    int k, m, n;
    string s, p, t, digits = "0123456789";

    s = string("Pupkin Vasiliy Vasilyevich");
    cout << s << endl;
    n = s.length();
    cout << "n=" << n << endl;
    k = s.find("Microsoft");
    cout << "index of Microsoft: " << k << endl;
    k = s.find("Pupkin");
    cout << "index of Pupkin: " << k << endl;
    k = s.find("Vasil");
    cout << "index of Vasil: " << k << endl;
    k = s.find("Vasil", 10);
    cout << "index of Vasil from 10: " << k << endl;
    p = s.substr(7);
    cout << "substr(7): " << p << endl;
    s.erase(7, 8);
    cout << "erase(7, 8): " << s << endl;
    s.erase(8);
    cout << "erase(8): " << s << endl;
    s.pop_back();
    cout << "pop_back: " << s << endl;
    s[2] = 's';
    cout << "s[2] = 's': " << s << endl;
    s.insert(3, "h");
    cout << "insert(3,\"h\"): " << s << endl;
    s += "Aleksandr Sergeevich";
    cout << "+=Aleksandr Sergeevich: " << s << endl;
    cout << "front: " << s.front() << endl;
    cout << "back: " << s.back() << endl;

    p.assign("My phone number is 291234567. Remember!");
    cout << endl << p << endl;
    k = p.find_first_of(digits);
    cout << "first of digits " << k << endl;
    m = p.find_last_of(digits);
    cout << "last of digits " << m << endl;
    p = p.substr(k, m - k + 1);
    cout << "substr(" << k << ", " << m - k + 1 << "): " << p << endl;
    t = "45";
```

```
    m = stoi(t);
    cout << endl << "stoi(\"t\"): " << m << endl;

    return 0;
}
```

```
index of Vasil: 7
index of Vasil from 10: 15
substr(7): Vasiliy Vasilyevich
erase(7, 8): Pupkin Vasilyevich
erase(8): Pupkin V
pop_back: Pupkin
s[2] = 's': Puskin
insert(3,"h"): Pushkin
+=Aleksandr Sergeevich: Pushkin Aleksandr Sergeevich
front: P
back: h

My phone number is 291234567. Remember!
first of digits 19
last of digits 27
substr(19, 9): 291234567

stoi("t"): 45
```

Рисунок 5.1 – Результат выполнения программы (пример 19)

ЛАБОРАТОРНАЯ РАБОТА № 13. КЛАСС STRING

Цель работы: научиться работать со строками класса string из стандартной библиотеки C++. Освоить основные методы этого класса для ввода, вывода, поиска, модификации и анализа строк.

Порядок выполнения работы

Написать программу обработки строки согласно варианту. Ввод строки осуществлять с консоли.

1. Вывести все слова наименьшей длины.
2. Вывести те слова, в которых нет повторения букв.
3. Вывести те слова, в которых буква «а» повторяется дважды.
4. Вывести те слова, которые начинаются и оканчиваются одинаковой буквой.
5. Вывести те слова, которые не содержат цифр.
6. Вывести те слова, у которых есть хотя бы одна буква «а», стоящая рядом с «м».
7. Вывести все слова в обратном порядке.
8. Вывести все слова, поменяв местами первую и последнюю буквы.

9. Удалить во всех словах букву «а», позиции справа заполнить «*».

10. Заменить во всех словах каждое вхождение буквы «х» на «ks».

11. Удалить все слова, заканчивающиеся на гласную букву.

12. Определить, есть ли в заданном предложении цифры, если есть – найти их сумму.

13. Символы самого длинного слова заменить символами 'm'.

14. В словах с нечетным количеством символов удалить среднюю букву.

15. В каждом слове, где есть буква «а», добавить после нее «да».

16. Удалить из строки слова, содержащие символ 'r'. Слова в строке разделены одним или несколькими пробелами.

17. Заменить в строке все группы подряд идущих пробелов на один пробел.

18. Поменять местами первое и второе слово строки. Слова в строке разделены одним или несколькими пробелами.

19. Заменить в строке все гласные на «X».

20. Удалить слова с нечетным количеством символов.

Контрольные вопросы

1. Какие преимущества у класса string перед символьными массивами?
2. Как с помощью методов класса string определить позицию подстроки в строке?
3. Какие методы класса string используются для получения подстроки и определения длины строки?
4. Как разбить строку на слова?
5. Как преобразовать число в строку и строку в число?

6 СТРУКТУРЫ В ЯЗЫКЕ С++

Структура – это составной тип данных, в котором под одним именем объединены данные различных типов. Отдельные данные структуры называются полями.

Объявляется структура следующим образом:

```
struct ИмяСтруктуры
{
    тип1 имяПоля1;
    тип2 имяПоля2;
    ...
    типN имяПоляN;
};
```

К полям структуры можно обращаться следующим образом:

```
имяСтруктуры.имяПоля
```

или

указательНаСтруктуру->имяПоля

Полями структур могут быть произвольные типы данных. Можно, например, создать структуру, полями которой будут массивы или другие структуры.

Структуры можно передавать в функции и возвращать из них.

ПРИМЕР. Написать программу вывода на консоль среднего балла студентов. О студенте хранится следующая информация: фамилия, имя и оценки по трем предметам (математика, физика, химия).

```
#include <iostream>
#include <string>
#include <windows.h>

using namespace std;

struct student
{
    string lastName;
    string firstName;
    int math;
    int physics;
    int chemistry;
};

void printAverage(student st)
{
    cout << "Студент: " << st.lastName << " " << st.firstName << endl;
    cout << "Средний балл: " << (double)(st.math + st.physics +
st.chemistry) / 3 << endl;
}

int main()
{
    SetConsoleOutputCP(1251);

    student st1 = { "Иванов", "Михаил", 7, 9, 8 };

    student st2 = { lastName: "Сорокин", firstName : "Сергей", math : 9,
physics : 10, chemistry : 8 };

    student st3 = { .lastName = "Михайлова", .firstName = "Анна", .math =
7, .physics = 4, .chemistry = 5 };

    student st4, * st5;

    st4.lastName = "Петрова";
    st4.firstName = "Мария";
    st4.math = 6;
    st4.physics = 6;
    st4.chemistry = 6;
```

```
    st5 = new student;
    st5->lastName = "Березкин";
    st5->firstName = "Василий";
    st5->math = 9;
    st5->physics = 10;
    st5->chemistry = 9;

    printAverage(st1);
    printAverage(st2);
    printAverage(st3);
    printAverage(st4);
    printAverage(*st5);

    return 0;
}
```

```
Студент: Иванов Михаил
Средний балл: 8
Студент: Сорокин Сергей
Средний балл: 9
Студент: Михайлова Анна
Средний балл: 5.33333
Студент: Петрова Мария
Средний балл: 6
Студент: Березкин Василий
Средний балл: 9.33333
```

Рисунок 6.1 – Результат выполнения программы (пример 20)

ЛАБОРАТОРНАЯ РАБОТА № 14. СТРУКТУРЫ

Цель работы: научиться объединять разрозненные данные в единый тип с помощью структур. Приобрести навыки работы с массивами структур, передачи структур в функции и возврата их из функций.

Порядок выполнения работы

Написать программу на языке C++ решения задачи согласно своему варианту. В программе при необходимости использовать функции, реализующие операции со структурами, а также массив переменных созданной структуры.

1. Создать структуру fraction, описывающую простую дробь. Поля: *n* – числитель, *d* – знаменатель. Для заданных двух дробей выполнить операции умножения и деления.

2. Создать структуру date, описывающую дату. Поля: *day*, *month*, *year*. Среди *N* человек с заданной датой рождения найти самого младшего и самого старшего человека.

3. Создать структуру parallelepiped, описывающую прямоугольный

параллелепипед. Поля: *a* и *b* – длины сторон прямоугольника-основания, *h* – высота. Найти, у какого из *N* заданных параллелепипедов объем наименьший.

4. Создать структуру time, описывающую время. Поля: *hours, minutes*. С использованием данной структуры решить следующую задачу: у прибора зафиксировано время начала работы и время завершения работы в текущий день. Определить общее время работы прибора за неделю.

5. Создать структуру triangle, описывающую треугольник. Поля: *a, b, c* – длины трех сторон. Найти, у какого из трех заданных треугольников площадь наибольшая.

6. Создать структуру straightLine, описывающую прямую. Поля: *x1, y1, x2, y2* – координаты двух точек, через которые проходит прямая. Для *K* заданных прямых вывести их уравнение в виде $y = a\,x + b$.

7. Создать структуру conus, описывающую конус. Поля: *r* – радиус круга-основания, *h* – высота. Определить для *N* заданных конусов объем и площадь поверхности.

8. Создать структуру point, описывающую точку на плоскости. Поля: *x, y* – координаты точки. С помощью этой структуры для каждой из *N* точек вывести сообщение, в какой координатной четверти она расположена.

9. Создать структуру parabola, описывающую параболу. Поля: *a, b, c* – коэффициенты уравнения $y = a\,x^2 + b\,x + c$. Для *N* заданных парабол определить точки пересечения с осью *OX*.

10. Создать структуру parallelogram, описывающую параллелограмм. Поля: *a, b* – длины сторон и *phi* – угол между сторонами. Для *K* заданных фигур определить k_1 – количество квадратов и k_2 – количество прямоугольников.

11. Создать структуру vector3, описывающую вектор в трехмерной системе координат. Поля: *x, y, z* – координаты вектора. Для *M* заданных векторов определить длину каждого и найти номер самого длинного вектора.

12. Создать структуру circle, описывающую окружность. Поля: *x0, y0* – координаты центра, *R* – радиус. Для каждой из *N* заданных окружностей определить, пресекается ли она с осями *OX* и *OY*.

13. Создать структуру rhomb, описывающую ромб. Поля: *d1, d2* – диагонали ромба. Для *K* заданных ромбов определить *k1* – количество квадратов и найти ромб с минимальной площадью.

14. Создать структуру complex, описывающую комплексное число. Поля: *re* – действительная и *im* – мнимая часть. Для *N* заданных комплексных чисел найти числа, у которых модуль наибольший и наименьший.

15. Создать структуру point3, точку в пространстве. Поля: *x, y, z* – координаты точки. Найти расстояние между двумя заданными точками.

16. Создать структуру date, описывающую дату. Поля: *day, month, year*. Для N человек с заданной датой рождения вывести сообщение, в какой декаде родился человек, в каком сезоне и в каком веке.

17. Создать структуру complex, описывающую комплексное число. Поля: *a* – амплитуда и *phi* – аргумент (в градусах). Для *N* заданных комплексных чисел вывести их в виде: *re*+i**im*.

18. Создать структуру straightLine, описывающую прямую. Поля: *a, b* –

коэффициенты уравнения $y = a\,x + b$. Для M заданных прямых определить $k1$ – количество прямых, параллельных оси OX, и $k2$ – количество прямых, параллельных оси OY.

19. Создать структуру polynomial, описывающую квадратный трехчлен вида $ax^2 + bx + c$. Поля: a, b, c – коэффициенты многочлена. Для N заданных трехчленов, определить количество корней и вывести их значения, если они существуют.

20. Создать структуру matrix2x2, описывающую квадратную матрицу 2×2. Поля: a11, a12, a21, a22 – элементы матрицы. Для N заданных матриц вычислить определитель и найти матрицу(ы), у которой определитель наибольший по модулю.

Контрольные вопросы

1. Что такое структура? Для чего она используется?
2. Как обратиться к полю структуры? В чем разница между операторами `.` и `->`?
3. Как объявить массив структур?
4. Можно ли передать структуру в функцию по значению и по адресу? В чем разница?
5. Как инициализировать структуру при ее объявлении?

7 ФАЙЛЫ В ЯЗЫКЕ C++

В языке C++ для работы с файлами имеется три потока: `ifstream` для чтения данных из файла, `ofstream` – для записи в файл и `fstream` – для работы как в режиме чтения, так и записи. Для работы с данными потоками подключают заголовочный файл `fstream`.

Для записи данных в текстовый файл необходимо:

1. Описать файловый поток:

```
ofstream f;
```

2. Открыть файл с помощью метода `open`.
3. Записать информацию в файл с помощью операции помещения в поток `<<`.
4. Закрыть файл методом `close`.

Для чтения данных из текстового файла необходимо:

1. Описать файловый поток:

```
ifstream f;
```

2. Открыть файл с помощью метода `open`.
3. Записать информацию в текстовый файл с помощью операции извлечения из потока `>>` или функции `getline`. При чтении каждой порции данных необходимо проверять, что чтение данных возможно.
4. Закрыть файл методом `close`.

Открывается файл следующим образом:

```
f.open("name", mode);
```

Здесь `f` – имя файлового потока; `name` – полное имя файла; `mode` – режим работы с открытым файлом.

Таблица 7.1 – Возможные значения **mode**

`ios::in`	Открытие файла в режиме чтения (режим по умолчанию для `ifstream`)
`ios::out`	Открытие файла в режиме записи (режим по умолчанию для `ofstream`). Если файл не существует, он создается
`ios::app`	Открытие файла в режиме дозаписи (запись в конец файла)
`ios::ate`	Передвинуться в конец уже открытого файла
`ios::trunc`	Удалить содержимое файла если он существует
`ios::binary`	Открыть двоичный файл

После удачного открытия файла в `f` будет храниться `true`, в противном случае – `false`.

После открытия файла для записи в него можно писать точно так, как и выводить на консоль, только вместо стандартного потока вывода `cout` необходимо указывать имя файлового потока:

```
f << a;
```

Чтение данных из файла осуществляется так, как и с консоли, только вместо потока ввода `cin` используют файловый поток:

```
f >> a;
```

Для проверки достигнут ли конец файла, служит метод `f.eof()`.

ПРИМЕР. Написать программу записи в текстовый файл следующих данных о людях: ФИО, город проживания, телефон.

```
#include <iostream>
#include <fstream>
#include <string>
#include <windows.h>

using namespace std;

int main()
{
    SetConsoleOutputCP(1251);
    SetConsoleCP(1251);
    int i, n;
    ofstream fo;
    string temp;
    cout << "n=";
    cin >> n;
    cin.ignore();
    fo.open("D:\\abc.txt");
    if (fo)
    {
```

```
        for (i = 1; i <= n; i++)
        {
            cout << "ФИО: ";
            getline(cin, temp);
            fo << temp << '\t';
            cout << "город: ";
            getline(cin, temp);
            fo << temp << '\t';
            cout << "телефон: ";
            getline(cin, temp);
            fo << temp;
            if (i != n)
                fo << endl;
        }
        fo.close();
    }
    else
        cout << "Ошибка при открытии файла";
    return 0;
}
```

```
n=5
ФИО: Иванов Александр Петрович
город: Гомель
телефон: 1234567
ФИО: Николаева Елена Анатольевна
город: Витебск
телефон: 2468013
ФИО: Печкин Сергей Алексеевич
город: Брест
телефон: 9876543
ФИО: Петровская Мария Андреевна
город: Минск
телефон: 9638527
ФИО: Блинов Василий Николаевич
город: Витебск
телефон: 3971268
```

Рисунок 7.1 – Результат выполнения программы (консоль)

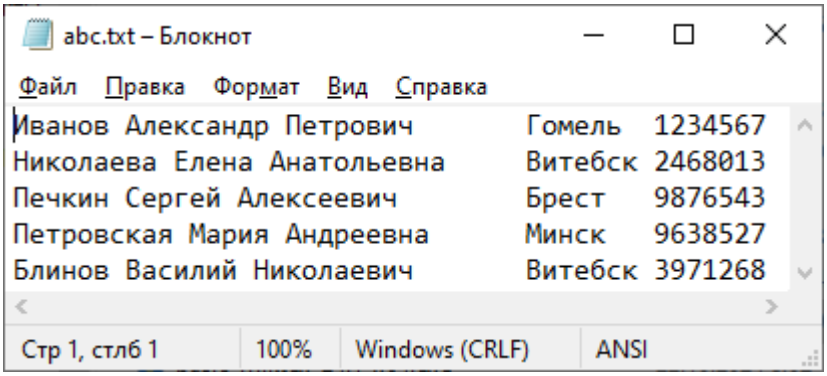

Рисунок 7.2 – Результат выполнения программы (записанный файл)

ПРИМЕЧАНИЕ. После ввода значения n потоке остается символ '\n'. Если следующим вводом должно быть извлечение символьных или строковых данных, то пользователю не будет предложено их ввести. Вместо них будет записан символ конца строки. Чтобы удалить данный символ из потока, использован метод ignore.

ПРИМЕР. Прочитать данные о людях из файла, записанного в предыдущем примере. Вывести на консоль сведения о людях, проживающих в Витебске.

```
#include <iostream>
#include <fstream>
#include <string>
#include <windows.h>

using namespace std;

struct person
{
    string name;
    string city;
    long long int phone;
};

int main()
{
    SetConsoleOutputCP(1251);
    ifstream fi;
    string temp;
    int prev, next;
    person a;

    fi.open("D:\\abc.txt");

    if (fi)
    {
        cout << "В Витебске проживают: " << endl;
        while (!fi.eof())
        {
            getline(fi, temp);
            prev = 0;
            next = temp.find('\t', prev + 1);
            a.name = temp.substr(prev, next);
            prev = next + 1;
            next = temp.find('\t', prev);
            a.city = temp.substr(prev, next - prev);
            a.phone = stoi(temp.substr(next + 1, temp.length() - next));

            if (a.city == "Витебск")
                cout << a.name << '\t' << a.phone << endl;
        }
```

```
        fi.close();
    }
    else
        cout << " Ошибка при открытии файла" << endl;

    return 0;
}
```

```
В Витебске проживают:
Николаева Елена Анатольевна     2468013
Блинов Василий Николаевич       3971268
```

Рисунок 7.3 – Результат выполнения программы

ПРИМЕЧАНИЕ. При условии, что в файле ФИО всегда состоит из трех слов, для чтения данных из файла можно использовать следующий более простой фрагмент кода:

```
while (!fi.eof())
{
    fi >> a.name;
    fi >> temp;
    a.name += " " + temp;
    fi >> temp;
    a.name += " " + temp;
    fi >> a.city;
    fi >> a.phone;

    if (a.city == "Витебск")
        cout << a.name << '\t' << a.phone << endl;
}
```

ЛАБОРАТОРНАЯ РАБОТА № 15. ТЕКСТОВЫЕ ФАЙЛЫ

Цель работы: научиться выполнять базовые операции с текстовыми файлами (открытие, чтение, запись, закрытие). Освоить обработку данных, считанных из файла, и сохранение результатов работы программы в файл.

Порядок выполнения работы

Написать две программы: первую для записи данных в текстовый файл, вторую – для чтения и обработки данных согласно варианту.

1. В файле находится список студентов группы. О каждом студенте хранятся следующие сведения: Ф.И.О., год рождения, номер зачетной книжки, пол, результаты последней сессии по четырем экзаменам (физика, математика, история, химия). Вывести на консоль информацию о студентах, получивших отличные оценки по математике.

2. В файле находятся результаты метеорологических наблюдений по

месяцам. Элемент списка хранит следующие данные: период наблюдения (месяц и год), количество дней месяца, количество выпавших осадков, количество облачных дней, количество дней с переменной облачностью. Определить и вывести на консоль сведения о месяце в заданном году, в течение которого выпало наибольшее количество осадков.

3. В файле находится список книг, хранящихся в библиотеке. О каждой книге хранятся следующие сведения: инвентарный номер, шифр УДК, название книги, Ф.И.О. автора, место издания, год издания. Вывести на консоль сведения о книгах данного автора, выпущенных не раньше указанного года.

4. В файле находится список читателей библиотеки. О каждом читателе хранятся следующие сведения: Ф.И.О., номер читательского билета, адрес, место работы или учебы, количество взятых книг, срок возврата книг (день, месяц, год). Вывести на консоль сведения о читателях, для которых истек срок возврата книг.

5. В файле находится список студентов факультета. О каждом студенте хранятся следующие сведения: Ф.И.О., группа, адрес (город, улица, дом). Вывести на консоль сведения о студентах, заданной группы, проживающих в указанном городе.

6. В файле находится список товаров в магазине. О каждом товаре хранятся следующие сведения: наименование, отдел магазина, цена, дата поступления, срок годности. Вывести на консоль сведения о товарах заданного отдела, для которых срок годности истекает в указанный день.

7. В файле находится список дисциплин кафедры. О каждой дисциплине хранятся следующие сведения: название, курс, группа, количество часов лекций, лабораторных, практических, сдается зачет или экзамен. Вывести на консоль сведения о дисциплинах с экзаменом, у которых более 32 часов лекций.

8. В файле находится список занятий в аудитории. О каждом занятии хранятся следующие сведения: группа, название дисциплины, день недели, номер пары, количество человек. Вывести на консоль сведения о занятиях, в самый загруженный день недели.

9. В файле находится список оргтехники. О каждом устройстве хранятся следующие сведения: вид устройства, модель, год приобретения, цена, подразделение. Вывести на консоль сведения об устройствах указанного подразделения, срок службы которых более 5 лет.

10. В файле находится расписание полетов самолетов. О каждом рейсе хранятся следующие сведения: пункт назначения, день недели, время отправления, время полета, стоимость билета. Вывести на консоль сведения о самом дешевом рейсе в указанный город.

11. В файле находятся сведения о сотрудниках организации. О каждом сотруднике хранятся следующие сведения: ФИО, возраст, должность, подразделение, стаж, стаж работы в организации. Вывести на консоль сведения о сотрудниках, работавших только в данной организации.

12. В файле находится список CD-дисков. О каждом CD-диски хранятся следующие сведения: название альбома, исполнитель, стиль, год выпуска, длительность, стоимость. Для заданного стиля вывести на консоль имена

исполнителей, диски которых выпущены не ранее 2015 года.

13. В файле находится список отгружаемой со склада продукции. О каждой отгрузке хранятся следующие сведения: число, месяц, год, наименование продукции, количество, куда отгружен. Для заданного года вывести на консоль сведения о продукции, отгруженной в указанную организацию.

14. В файле находится список сотрудников подразделения. О каждом сотруднике хранятся следующие сведения: Ф.И.О., должность, год рождения, образование. Вывести на консоль сведения о сотрудниках с высшим образованием не старше 1980 года рождения.

15. В файле находится список автотранспорта предприятия. О каждом автомобиле хранятся следующие сведения: тип транспортного средства (легковой, автобус, грузовой), номер, год выпуска, пробег. Вывести на консоль сведения об автобусах, не старше 7 лет.

16. В файле находится информация о свободных местах в поездах. О каждом поезде хранятся следующие сведения: время отправления, пункт назначения, число свободных мест. Вывести на консоль информацию о поездах, следующих в Москву, на которые имеется больше заданного количества свободных мест.

17. В файле находится информация об участниках спортивных соревнований. О каждом участнике хранятся следующие сведения: название команды, фамилия спортсмена, возраст, рост и вес. Вывести информацию о спортсменах заданной команды, рост которых выше 190 см.

18. В файле находится список стран мира. О каждой стране хранятся следующие сведения: название страны и ее столицы, название части света, в которой находится страна, площадь страны, население. Вывести информацию о странах, находящихся в Африке и численностью населения больше заданной величины.

19. Файл содержит список пациентов. У каждого: Ф.И.О., возраст, диагноз, дата последнего приёма (день, месяц, год), рекомендации врача. Вывести на консоль информацию о пациентах, которым пора пройти повторный осмотр (более 6 месяцев прошло с момента последнего приёма).

20. Файл содержит список городов. Для каждого: название города, страна, население, площадь (км²), плотность населения (чел./км²), уровень безработицы (%). Вывести на консоль информацию о городе с наибольшей плотностью населения среди тех, где уровень безработицы выше заданного значения.

Контрольные вопросы

1. Какие типы файлов по способу обработки бывают?
2. Как проверить, что файл был успешно открыт?
3. Чем отличается чтение из файла с помощью оператора **>>** от функции **getline**?
4. Как организовать чтение данных из файла до его конца?
5. Почему важно закрывать файл после работы с ним?

8 ДИНАМИЧЕСКИЕ СТРУКТУРЫ ДАННЫХ

8.1 Общие сведения о динамических структурах данных

Динамические структуры данных – это структуры данных, память под которые выделяется и освобождается по мере необходимости.

Динамические структуры данных в процессе существования в памяти могут изменять не только число составляющих их элементов, но и характер связей между элементами.

Динамические структуры, по определению, характеризуются отсутствием физической смежности элементов структуры в памяти, непостоянством и непредсказуемостью размера (числа элементов) структуры в процессе ее обработки.

Поскольку элементы динамической структуры располагаются по непредсказуемым адресам памяти, адрес элемента такой структуры не может быть вычислен из адреса начального или предыдущего элемента. Для установления связи между элементами динамической структуры используются указатели, через которые устанавливаются явные связи между элементами. Такое представление данных в памяти называется *связным*.

Достоинства связного представления данных – в возможности обеспечения значительной изменчивости структур:

- размер структуры ограничивается только доступным объемом машинной памяти;
- при изменении логической последовательности элементов структуры требуется не перемещение данных в памяти, а только коррекция указателей;
- большая гибкость структуры.

Вместе с тем, связное представление не лишено и недостатков, основными из которых являются следующие:

- на поля, содержащие указатели для связывания элементов друг с другом, расходуется дополнительная память;
- доступ к элементам связной структуры может быть менее эффективным по времени.

Динамические структуры данных можно классифицировать *по принципу организации данных*.

1. Линейные структуры – элементы располагаются последовательно, один за другим:
   - односвязный список;
   - двусвязный список;
   - циклические списки;
   - стек;
   - очередь;
   - дек.
2. Древовидные структуры (иерархические) – элементы организованы в

виде иерархии, напоминающей дерево:

– бинарное дерево;

– сильноветвящееся дерево.

3. Хеш-таблицы – используют хеш-функцию для преобразования ключа в индекс массива.

4. Графы – наиболее общая структура, состоящая из вершин (узлов) и ребер (связей).

8.2 Линейные односвязные списки

Линейный связный список – это структура данных, состоящая из элементов одного типа, связанных между собой последовательно посредством указателей.

В такой структуре все элементы линейно упорядочены, но порядок определяется не номерами, как в массиве, а указателями, входящими в состав элементов списка.

Каждый элемент односвязного списка состоит из двух областей памяти: поля данных (информационное поле) D и ссылок (адресное поле) P.

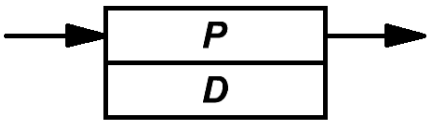

Рисунок 8.1 – Элемент односвязного списка

Поле данных – область памяти, в которой содержатся те данные, ради которых и создается структура.

Полей данных может быть несколько. Каждый элемент списка содержит поле данных, называемое *ключ*, который однозначно идентифицирует этот элемент. Ключ обычно бывает либо целым числом, либо строкой.

Поле ссылок – область памяти, в которой содержатся один указатель, связывающий данный элемент со следующим. Последний элемент списка указывает на `NULL`.

Каждый односвязный список имеет особый элемент – указатель на начало списка, который называется *головой* списка (`Head`).

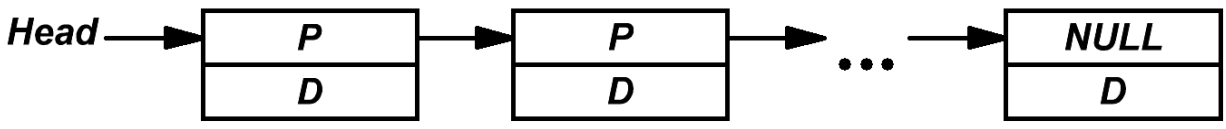

Рисунок 8.2 – Односвязный список

В односвязном списке можно передвигаться только в сторону конца списка. Узнать адрес предыдущего элемента, опираясь на содержимое текущего узла, невозможно.

Реализовать узел односвязного списка можно с помощью класса или структуры. Также можно создавать односвязный список на основе шаблонного класса, оперирующего некоторым обобщенным типом.

Основными операциями, осуществляемыми с однонаправленными списками, являются:

- создание списка;
- печать (просмотр) списка;
- вставка элемента в начало списка;
- вставка элемента в конец списка;
- вставка элемента в указанную позицию;
- удаление элемента из списка;
- поиск элемента в списке;
- проверка пустоты списка;
- удаление списка.

Могут реализовываться и дополнительные операции (например, сортировка списка, перестановка элементов).

ПРИМЕР. Решить задачу частотного анализа текста – определения всех слов, встречающихся в тексте и их количества. Анализируемый текст берется из файла ***input.txt***, полученный список помещается в файл ***output.txt***.

Для анализа запишем в файл ***input.txt*** следующий текст:

```
Lo in the orient when the gracious light
Lifts up his burning head, each under eye
Doth homage to his new-appearing sight,
Serving with looks his sacred majesty;
And having climbed the steep-up heavenly hill,
Resembling strong youth in his middle age,
Yet mortal looks adore his beauty still,
Attending on his golden pilgrimage:
But when from highmost pitch, with weary car,
Like feeble age he reeleth from the day,
The eyes (fore duteous) now converted are
From his low tract and look another way:
So thou, thyself outgoing in thy noon,
Unlooked on diest unless thou get a son.
```

```
#include <cctype>
#include <fstream>
#include <iostream>
#include <string>

using namespace std;

struct Node
{
    string word; // поле данных
    int count; // поле данных
    Node* next; // поле ссылок (ссылка на следующий узел)
```

```
};

typedef Node* PNode; // тип данных: указатель на узел

PNode createNode(string NewWord) // создание узла
{
    PNode NewNode = new Node; // указатель на новый узел
    (NewNode->word).assign(NewWord); // записать слово
    NewNode->count = 1; // счетчик слов = 1
    NewNode->next = NULL; // следующего узла нет
    return NewNode; // результат функции – адрес узла
}

void addFirst(PNode& Head, PNode NewNode) // вставка узла в начало списка
{
    NewNode->next = Head;
    Head = NewNode;
}

void addAfter(PNode p, PNode NewNode) // вставка узла в список после
заданного
{
    NewNode->next = p->next;
    p->next = NewNode;
}

void addBefore(PNode& Head, PNode p, PNode NewNode) // вставка узла в
список перед заданным
{
    PNode q = Head;
    if (Head == p) // если p голова списка
    {
        addFirst(Head, NewNode); // вставка перед первым узлом
        return;
    }
    while (q && q->next != p) // ищем узел, за которым следует p
        q = q->next;
    if (q) // если нашли такой узел,
        addAfter(q, NewNode); // добавить новый после него
}

void addLast(PNode& Head, PNode NewNode) // вставка узла в конец списка

{
    PNode q = Head;
    if (Head == NULL) // если список пуст,
    {
        addFirst(Head, NewNode); // вставляем первый элемент
        return;
    }
    while (q->next)
```

```
        q = q->next; // ищем последний элемент
    addAfter(q, NewNode);
}

PNode find(PNode Head, string NewWord) // поиск узла с заданным ключом
{
    PNode q = Head;
    while (q && (q->word).compare(NewWord))
        q = q->next;
    return q;
}

PNode findPlace(PNode Head, string NewWord) // поиск места вставки
{
    PNode q = Head;
    while (q && (NewWord.compare((q->word)) > 0))
        q = q->next;
    return q;
}

void deleteNode(PNode& Head, PNode OldNode) // удаление заданного узла
(задается адрес) из списка
{
    PNode q = Head;
    if (Head == OldNode)
        Head = OldNode->next; // удаляем первый элемент
    else
    {
        while (q && q->next != OldNode) // ищем элемент
            q = q->next;
        if (q == NULL)
            return; // если не нашли, выход
        q->next = OldNode->next;
    }
    delete OldNode; // освобождаем память
}

void deleteList(PNode Head) // удаление списка
{
    if (Head != NULL)
    {
        deleteList(Head->next);
        delete Head;
    }
}

void editWord(string* word) // обрезаем от слова знаки препинания и
скобки
{
    if (!isalpha((*word).back()))
        (*word).pop_back();
```

```
    if (!isalpha((*word).back()))
        (*word).pop_back();
    if (!isalpha((*word).front()))
        (*word).erase(0, 1);
    for (unsigned int i = 0; i < (*word).length(); i++)
        (*word).at(i) = tolower((*word).at(i));
}

int main()
{
    PNode Head = NULL, p, where;
    ifstream fi;
    ofstream fo;
    string word;
    fi.open("input.txt");
    if (fi)
    {
        while (!fi.eof())
        {
            fi >> word;
            editWord(&word);
            p = find(Head, word);
            if (p != NULL)
                p->count++;
            else
            {
                p = createNode(word);
                where = findPlace(Head, word);
                if (!where)
                    addLast(Head, p);
                else
                    addBefore(Head, where, p);
            }
        }
        fi.close();
        fo.open("output.txt");
        if (fo)
        {
            p = Head;
            while (p)
            {
                fo.width(20);
                fo << left << p->word << '\t' << p->count << endl;
                p = p->next;
            }
            fo.close();
        }
        else
            cout << "file opening error!" << endl;
        deleteList(Head);
    }
```

```
    else
        cout << "file not found!" << endl;
    return 0;
}
```

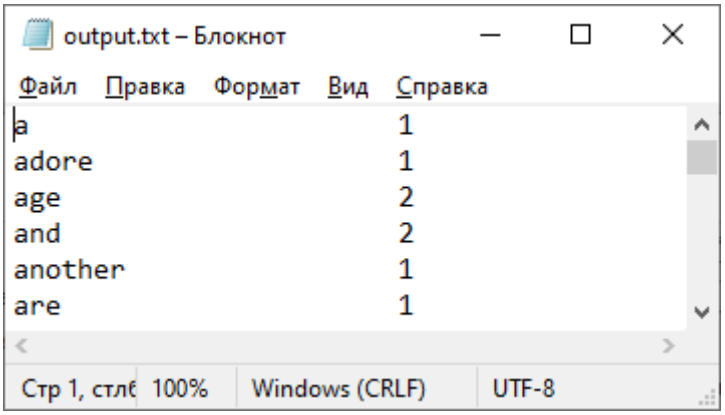

Рисунок 8.3 – Фрагмент полученного файла (output.txt)

8.3 Бинарные поисковые деревья

Дерево – это структура данных, представляющая собой совокупность элементов и отношений, образующих иерархическую структуру этих элементов.

Каждый элемент дерева называется *вершиной (узлом)* дерева.

Вершины дерева соединены направленными дугами, которые называют *ветвями* дерева.

В каждый узел (за исключением корня) ведет ровно одна ветвь.

Предком для узла x называется узел дерева, из которого существует путь в узел x.

Потомком узла x называется узел дерева, в который существует путь (по стрелкам) из узла x.

Родителем для узла x называется узел дерева, из которого существует непосредственная дуга в узел x.

Дочерним узлом (сыном) узла x называется узел дерева, в который существует непосредственная дуга из узла x.

Корень – это начальный узел дерева, в который не ведет ни одной дуги.

Листом дерева называется узел, не имеющий потомков.

Внутренней вершиной (внутренним узлом) называется узел, имеющий и предка, и потомков.

Число ветвей, которое нужно пройти от корня к узлу x, называется *длиной пути* к x.

Высота – длина самого длинного пути от корня до листа.

Бинарное дерево – это частный случай дерева, когда каждый узел имеет не более двух потомков – левого и правого.

Бинарное дерево поиска – это бинарное дерево, обладающее дополнительными свойствами: значение левого потомка меньше значения родителя, а значение правого потомка больше значения родителя для каждого узла дерева.

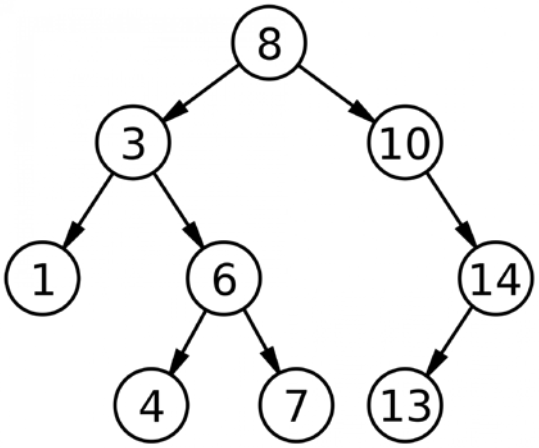

Рисунок 8.4 – Бинарное дерево поиска

Деревья являются рекурсивными структурами, так как каждое поддерево также является деревом.

В такой структуре каждый элемент является:

– либо пустой структурой;

– либо элементом, с которым связано конечное число поддеревьев.

Действия с рекурсивными структурами удобнее всего описываются с помощью рекурсивных алгоритмов.

ПРИМЕР. Рассмотрим пример построения дерева, каждый узел которого хранит два целочисленных значения, меньшее из двух чисел – ключ. Данные для построения хранятся в файле. Затем для заданного ключа выведем значение или сообщение, что такого ключа нет.

Для примера в качестве входного возьмем файл ***example.txt***, содержимое которого представлено на рисунке 8.5.

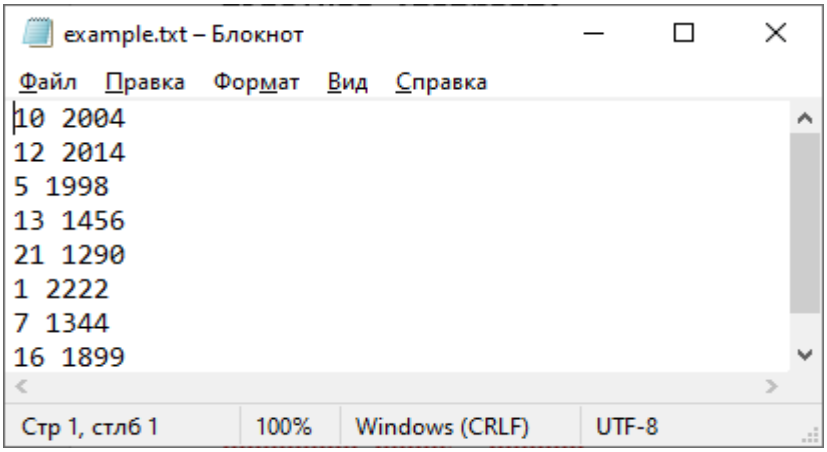

Рисунок 8.5 – Содержимое входного файла (example.txt)

```
#include <iostream>
#include <fstream>
#include <windows.h>

using namespace std;
```

```
struct Node
{
    int key;
    int value;
    Node *left, *right;
};
typedef Node* PNode;

void addToTree(PNode& Tree, int newKey, int newValue)
{
    if (!Tree)
    {
        Tree = new Node;
        Tree->key = newKey;
        Tree->value = newValue;
        Tree->left = NULL;
        Tree->right = NULL;
        return;
    }
    if (newKey < Tree->key)
        addToTree(Tree->left, newKey, newValue);
    else
        addToTree(Tree->right, newKey, newValue);
}

PNode search(PNode Tree, int what)
{
    if (!Tree)
        return NULL;
    if (what == Tree->key)
        return Tree;
    if (what < Tree->key)
        return search(Tree->left, what);
    else
        return search(Tree->right, what);
}

PNode makeTreeFromFile(string fileName)
{
    PNode Tree = NULL;
    ifstream f;
    f.open(fileName);
    if (f)
    {
        PNode current;
        int k, v;
        while (!f.eof())
        {
            f >> k >> v;
            current = search(Tree, k);
```

```
            if (!current)
                addToTree(Tree, k, v);
            else
                cout << "An element with key " << k << " already exists!"
<< endl;
        }
    }
    else
        cout << "file " << fileName << " not found!" << endl;
    return Tree;
}

void printTree(PNode Tree, int level)
{
    int i;
    if (Tree)
    {
        printTree(Tree->right, level + 1);
        for (i = 0; i < level; i++)
            cout << "\t";
        cout << Tree->key;
        printTree(Tree->left, level + 1);
    }
    else
        cout << endl;
}

int main()
{
    PNode Tree = makeTreeFromFile("example.txt");
    printTree(Tree, 0);
    int number = 7;
    PNode node = search(Tree, number);
    if (node)
        cout << "The key " << number << " with value " << node->value <<
endl;
    else
        cout << "The key " << number << " not found in tree!" << endl;
    return 0;
}
```

```
                        21
                                16
                13
        12
10
                7
        5
                1
The key 7 with value 1344
```

Рисунок 8.6 – Результат выполнения программы

ЛАБОРАТОРНАЯ РАБОТА № 16. ЛИНЕЙНЫЕ ОДНОСВЯЗНЫЕ СПИСКИ

Цель работы: понять принципы организации и работы с динамическими структурами данных на примере линейного односвязного списка. Освоить базовые операции со списками: создание, добавление, удаление, поиск, просмотр.

Порядок выполнения работы

- Создать проект, код которого представлен в примере на стр. 77 – 81.
- В данный проект для списка добавить функцию (или функции) для решения задачи по вариантам. Результирующий данный или измененный список записать в файл ***result.txt***.

1. Удалить заданное слово из списка.
2. Выполнить перестановку заданного элемента в начало списка.
3. Разработать функцию вычисления количества элементов в списке.
4. Выполнить перестановку двух заданных элементов в списке.
5. Найти в списке самые длинные слова.
6. Найти в списке слова с количеством упоминаний 2.
7. Удалить из списка слова, начинающиеся на заданную букву.
8. Найти в списке слова с максимальным упоминанием.
9. Удалить все элементы списка, расположенные после указанного элемента.
10. Выполнить перестановку заданного элемента в конец списка.
11. Удалить все элементы списка, расположенные до указанного элемента.
12. Найти в списке слова с минимальным упоминанием.
13. Найти в списке самые короткие слова.
14. Удалить из списка слова длиной в 3 символа.
15. Найти количества слов, состоящих из заданного количества букв.
16. Выполнить перестановку заданного элемента в середину списка.
17. Удалить из списка слова с упоминанием больше 2.
18. Выполнить перестановку первого и последнего слова.
19. Найти количество слов с упоминанием больше 3.
20. Найти количество слов в тексте с учетом их упоминаний.

Контрольные вопросы

1. Что такое узел списка? Из каких полей он состоит?
2. Что такое «голова» списка (Head)?
3. В чем принципиальное отличие хранения данных в массиве и в связном списке? Какие преимущества и недостатки у связных списков по сравнению с массивами?

4. Как избежать потери ссылок при удалении/добавлении элементов из списка?

5. Почему при удалении списка необходимо освобождать память для каждого узла?

ЛАБОРАТОРНАЯ РАБОТА № 17. БИНАРНЫЕ ДЕРЕВЬЯ ПОИСКА

Цель работы: изучить принципы построения и работы с бинарными деревьями поиска. Освоить рекурсивные алгоритмы добавления элемента в дерево и поиска элемента по ключу.

Порядок выполнения работы

- Создать проект, код которого представлен в примере на стр. 82 – 84.
- В данный проект для списка добавить метод (или методы) для решения задачи по вариантам.

1. Поменять местами информацию, содержащую максимальный и минимальный ключи.
2. Подсчитать число листьев в дереве.
3. Удалить из дерева ветвь с вершиной, имеющей заданный ключ.
4. Определить глубину дерева.
5. Определить число узлов на заданном уровне дерева.
6. Удалить из левой ветви дерева узел с максимальным значением и все связанные с ним узлы.
7. Определить число листьев на заданном уровне дерева.
8. Определить количество узлов в дереве, имеющих только одного потомка.
9. Определить количество узлов правой ветви дерева.
10. Определить количество записей в дереве, начинающихся с введенной с клавиатуры цифры.
11. Найти среднее значение всех ключей дерева и узел, имеющий ближайший к этому значению ключ.
12. Определить количество узлов левой ветви дерева.
13. Определить число узлов в дереве, имеющих двух потомков.
14. Найти запись с ключом, ближайшим к среднему значению между максимальным и минимальным значениями ключей.
15. Определить среднее значений на заданном уровне дерева.
16. К каждому значению прибавить заданное число.
17. Значение в корневом узле заменить средним арифметическим всех значений.
18. Удалить из левой ветви дерева узел с минимальным значением и все связанные с ним узлы.
19. Найти среднее значение улов правой ветви.
20. Поменять местами узлы с заданными значениями.

Контрольные вопросы

1. Дайте определение бинарного дерева.
2. Какое свойство должно выполняться для любого узла бинарного дерева поиска?
3. Почему рекурсия является естественным способом для работы с деревьями?
4. Опишите алгоритм поиска элемента в бинарном дереве.
5. Что такое обход дерева? Какие виды обхода вы знаете?

ЛИТЕРАТУРА

1. Гуриков, С. Р. Основы алгоритмизации и программирования на Visual C++ : учебное пособие / С. Р. Гуриков. – Москва : ИНФРА-М, 2022. – 515 с.

2. Колдаев, В. Д. Структуры и алгоритмы обработки данных : учебное пособие / В. Д. Колдаев. – Москва : РИОР : ИНФРА-М, 2021. – 296 с.

3. Навроцкий, А. А. Основы алгоритмизации и программирования в среде Visual C++ : учеб.-метод. пособие / А. А. Навроцкий ; БГУИР. – Минск, 2014. – 160 с.

4. Немцова, Т. И. Программирование на языке высокого уровня. Программирование на языке C++ : учебное пособие / Т. И. Немцова, С. Ю. Голова, А. И. Терентьев ; под ред. Л. Г. Гагариной. – Москва : ФОРУМ : ИНФРА-М, 2024. – 512 с.

Учебное издание

ОСНОВЫ АЛГОРИТМИЗАЦИИ И ПРОГРАММИРОВАНИЯ НА ЯЗЫКАХ ВЫСОКОГО УРОВНЯ

Методические указания
по выполнению лабораторных работ

Составитель:
Соколова Анна Сергеевна

Редактор *Р.А. Никифорова*
Корректор *А.С. Прокопюк*
Компьютерная верстка *А.С. Соколова*

Подписано к печати 26.11.2025. Усл. печ. листов 5,6.
Уч.-изд. листов 6,9. Заказ № 228.

Учреждение образования «Витебский государственный технологический университет»
210038, г. Витебск, Московский пр., 71.
Отпечатано на ризографе учреждения образования
«Витебский государственный технологический университет».
Свидетельство о государственной регистрации издателя, изготовителя,
распространителя печатных изданий № 1/172 от 12 февраля 2014 г.
Свидетельство о государственной регистрации издателя, изготовителя,
распространителя печатных изданий № 3/1497 от 30 мая 2017 г.