МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
«Курганский государственный университет»

Кафедра информационных технологий
и методики преподавания информатики

МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ ДЛЯ ИЗУЧЕНИЯ ОСНОВ ОБЪЕКТНО-ОРИЕНТИРОВАННОГО ПРОГРАММИРОВАНИЯ С ИСПОЛЬЗОВАНИЕМ ЯЗЫКА С++

Методические рекомендации для студентов
направлений 01.03.01, 010100.62 «Математика»,
050100.62 «Педагогическое образование»
(профиль «Математика»),
44.03.01 «Педагогическое образование»
(профиль «Информатика»)

Курган 2015

Кафедра: «Информационных технологий и методики преподавания информатики»

Дисциплина: «Объектно-ориентированное программирование»
(направления 01.03.01, 010100.62, 050100.62, 44.03.01).

Составители: ст. преподаватель Ю.В. Адаменко
канд. пед. наук, доцент А.А. Медведев,

Утверждены на заседании кафедры «27» августа 2014 г.

Рекомендованы методическим советом университета «19» декабря 2014 г.

Лабораторная работа № 1. Лексические основы языка C++ *(4 часа)*

Начнем с рассмотрения простого примера:

```
//Простая программа, написанная на языке C++.
/*Программа выводит
строку «Моя первая программа на языке C++».*/
  #include <iostream.h>
  void main ()
  {
    cout << “\nМоя первая программа на языке C++\n”;
  }
```

//Простая программа, написанная на языке C++. **/*Программа выводит** **строку «Моя первая программа на языке C++».*/**	*это комментарии. Однострочный комментарий начинается символом // и заканчивается символом «конец строки», а многострочный начинается символом /* и заканчивается */.*
#include <iostream.h>	*это команда (директива) препроцессора, обеспечивающая включение в программу средств связи со стандартными потоками ввода/вывода данных.*
void main () **{** **……** **}**	*заголовок функции с именем **main**, именно с нее начинается выполнение программы (такая функция в программе должна быть **единственной!**).* *Спецификатор типа **void** указывает, что функция не возвращает никакого значения.* ***Круглые скобки** требуются в соответствии с форматом заголовка любой функции, в них помещается список параметров (в этом примере они отсутствуют).* *Тело функции (последовательность операторов, описаний и определений) помещено между **фигурными скобками**.*
cout << “\nМоя первая программа на языке C++\n”;	*оператор с именем **cout**, согласно информации, содержащейся в файле **iostream.h**, является именем объекта, который обеспечивает вывод информации на экран дисплея (в стандартный поток вывода). Информация для вывода передается объекту cout с помощью операции << («поместить в»). В данном случае это строка (строковая константа), заключенная в кавычки, почти любая последовательность символов (среди которых могут быть обозначения не изображаемых на экране дисплея управляющих символов, например, **\n** – переход в начало новой строки).*

Задание 1. Откройте Borland C++, создайте новый файл (File|New) и наберите данный пример. Сохраните файл в своей папке с именем **first.cpp** (File|Save). Компилируем файл **first.cpp** для устранения синтаксических ошибок (F9, или Alt+F9, или Project|Compile). В результате компиляции будет создан объектный файл (**first.obj**). Выполните программу (Ctrl+F9 или Debug|Run). В результате объектный файл дополняется нужными библиотечными функциями (компонуется) и будет создан исполняемый модуль программы (**first.exe**).

Прочитать в методическом пособии [1] стр. 3-6. Знать понятия лексемы, идентификатора, ключевого (служебного) слова, константы и их классификация.

Задание 2. Проиллюстрируем влияние абсолютного значения константы и использованных в ее изображении суффиксов L, U на тип данных, который ей присваивается на этапе компиляции.

Примечание: в программе будем использовать унарную операцию ***sizeof****, позволяющую определить размер в байтах области памяти, выделенной для стоящего справа операнда.*

В файле **suffiks.cpp** наберите текст следующей программы:

```
#include<iostream.h>
void main ()
{ cout<<"\n sizeof 111 = "<< sizeof 111;
  cout<<"\n sizeof 111u = "<< sizeof 111u;
!самостоятельно проверьте остальные суффиксы!
cout<<"\n\t sizeof 40000 = "<< sizeof 40000;
cout<<"\n\t sizeof 40000u = "<< sizeof 40000u;
!самостоятельно проверьте остальные суффиксы!
}
```

Задание 3. Проиллюстрируем на примере, какой размер памяти в байтах выделяется вещественным константам разного типа.

В файле **memory.cpp** наберите текст следующей программы:

```
#include<iostream.h>
void main ()
{ cout<<"\n sizeof 3.141592653589793 = "<< sizeof 3.141592653589793;
  cout<<"\n sizeof 3.14159 = "<< sizeof 3.14159;
!самостоятельно проверьте суффиксы F и L для последнего значения!
}
```

Задание 4. Проиллюстрируйте (*!самостоятельно!*) на примере, какую длину имеют следующие строки и символьные константы: пустая строка, ‘F’, “F”, перевод строки (как строковая константа и как строка), шестнадцатеричный код символа \xFF (как строковая константа и как строка). Текст программы сохраните в файле **dlinastr.cpp**.

Результат выполнения программы должен получиться следующим:

sizeof “” = 1

sizeof ‘F’ = 1	sizeof ‘\n’ = 1	sizeof ‘\xFF’ = 1
sizeof “F” = 2	sizeof “\n” = 2	sizeof “\xFF” = 2

Почитать в методическом пособии [1] стр. 12-15. Знать основные типы данных, служебные слова, применяемые отдельно или совместно с другими именами типов, способы размещения переменной в памяти и время ее существования.

Определения и описания переменных. Определение переменных имеет следующий формат:

s m тип имя1 иниц1, имя2 иниц2, …;

s – спецификатор класса памяти; m – модификатор **const** (определяемый объект недоступен в других модулях программы, его нельзя изменять во время выполнения программы) или **volatile** (значение объекта может изменяться в проме-

жутке между явным обращением к нему), **тип** – один из основных типов, **имя** – идентификатор, **иниц** – необязательный инициализатор, определяющий начальное значение соответствующего объекта.

Синтаксис инициализатора переменной:

=инициализирующее_выражение

или

(инициализирующее_выражение)

разрешена только внутри функции

Примеры описаний:

```
extern int q; //Внешняя переменная
float fn(int, double); //Прототип функции
extern const float pi; //Внешняя константа
struct st; //Имя структуры (класса)
typedef unsigned char simbol; //Новый тип symbol
```

Примеры определений:

```
char sm;//Переменная автоматической или внешней памяти
float dim=10.0;//Инициализированная переменная
double Enum (2.718282); //Инициализированная переменная
                        //автоматической памяти
const float pi=3.141593;//Константа
float x2 (float x) {return x*x;}//Функция
enum {zero, one, two};//Перечисление
```

Неоднозначность есть в определении переменной **char sm;**. Если такое описание находится в блоке, то это определение неинициализированной переменной автоматической памяти. Если такое описание появляется вне блоков и классов, то это определение статической внешней переменной, которой по умолчанию присваивается нулевое значение, а для символьной переменной – пробел. В файлах, где нужен доступ к этой глобальной переменной, должно быть помещено ее описание вида **extern char sm;**. Подобное описание нужно для доступа из функций к внешним объектам, определения которых размещены в том же файле, но ниже текста определения функции.

Задание 5. В программе **opred.cpp** определите следующие внешние переменные (до описания функции main): pi (вещественная, с явной инициализацией), s0 (целая, инициализация по умолчанию), s1 (целая, с явной инициализацией значением 5). В функции main описать переменные s0, внутреннюю переменную целого типа s1 с явной инициализацией значением 4, символьную переменную s2. За пределами функции main определить переменную s2 символьного типа с явной инициализацией значением ‘A’.

В результате работы программы на экране должно отобразиться следующее:

Инициализация по умолчанию: s0 = 0

Явная инициализация: s2 = A

Внутренняя переменная: s1 = 4

Внешняя переменная: pi = 3.14159

Почитать материалы шагов 7-9 [2]. Знать форматный ввод и вывод значений.

Рассмотреть примеры использования функций форматного ввода и вывода данных.

Лабораторная работа № 2. Операции *(4 часа)*

Почитать в [1] стр. 6–12. Знать арифметические операции, инкремент и декремент, операции отношения и присваивания, логические операции, условную операцию и операции преобразования типов.

Арифметические операции

Задание 6. В программе **arifm_oper.cpp** реализовать ввод двух целых значений (a>b, отличны от 0, использовать форматный ввод) и вывод в таблице следующих данных (рисунок 1).

```
Введите два целых числа: 13 3
                         10-е     8-е      16-е
Первое число             13       015      0xd
Второе число             3        03       0x3
Сумма                    16       020      0x10
Разность                 10       012      0xa
Произведение             39       047      0x27
Целочисленное деление    4        04       0x4
Остаток от делени        1        01       0x1
Деление                  4.333333
```

Рисунок 1 - Вид выполненного задания 6

Задание 7. Решить следующие задачи по вариантам (таблица 1) (назовите файлы **z7_номер задачи.cpp**)

Таблица 1 - Задачи по вариантам к заданию 7

Вариант	*1 Вычислить значение функции при введенных значениях a, b, c, d, x*	*2 Решить следующую задачу*
1	$f(x)= ax^3 +bx^2+cx+d$	Даны два числа. Найти среднее арифметическое их квадратов и среднее арифметическое их модулей
2	$f(x)=\frac{a}{bx+c}+d$	Скорость лодки в стоячей воде V км/ч, скорость течения реки U км/ч (U < V). Время движения лодки по озеру T1 ч, а по реке (против течения) — T2 ч. Определить путь S, пройденный лодкой
3	$f(x)=ae^{bx+c}+d$	Скорость первого автомобиля V1 км/ч, а второго - V2 км/ч, расстояние между ними S км. Определить расстояние между ними через T часов, если автомобили удаляются друг от друга
4	$f(x)=a\ sin(bx+c)+d$	Скорость первого автомобиля V1 км/ч, а второго - V2 км/ч, расстояние между ними S км. Определить расстояние между ними через T часов, если автомобили первоначально движутся навстречу друг другу
5	$f(x)= a\ cos(bx+c)+d$	Найти периметр и площадь прямоугольного треугольника, если даны длины катетов a и b
6	$f(x)=a\ ln(bx+c)+d$	Дана длина ребра куба. Найти площадь грани, площадь полной поверхности и объем куба

Продолжение таблицы 1

Вариант	*1 Вычислить значение функции при введенных значениях a, b, c, d, x*	*2 Решить следующую задачу*
7	$f(x)=a\sqrt{bx+c}+d$	Найти длину окружности и площадь круга заданного радиуса R
8	$f(x)=\frac{a}{\sqrt{bx+c}}+d$	Найти площадь кольца, внутренний радиус которого равен R1, а внешний радиус равен R2 (R1 < R2)
9	$f(x)=\frac{ax+c}{bx+d}$	Дана сторона равностороннего треугольника. Найти площадь этого треугольника и радиусы вписанной и описанной окружностей
10	$f(x)=a\sin x+b\cos x+c$	Дана длина окружности. Найти площадь круга, ограниченного этой окружностью
11	$f(x)=\frac{ax}{bx^2+cx+d}$	Дана площадь круга. Найти длину окружности, ограничивающей этот круг
12	$f(x)=\frac{ax^2+bx+c}{dx}$	Найти периметр и площадь равнобедренной трапеции с основаниями a и b (a > b) и углом α при большем основании (угол дан в радианах)
13	$f(x)=a^{bx+c}+d$	Найти периметр и площадь прямоугольной трапеции с основаниями a и b (a > b) и острым углом α (угол дан в радианах)
14	$f(x)=\sqrt{ax+\sqrt{bx+c}+d}$	Найти расстояние между двумя точками с заданными координатами (x1, y1) и (x2, y2)
15	$f(x)=\frac{a}{\sqrt{bx+\sin x^2+c}}$	Даны координаты трех вершин треугольника (x1, y1), (x2, y2), (x3, y3). Найти его периметр и площадь
16	$f(x)=\sin(ax^2+bx+c)$	Найти корни квадратного уравнения $Ax^2 + B\cdot x+C = 0$, заданного своими коэффициентами A, B, C (коэффициент A не равен 0), если известно, что дискриминант уравнения неотрицателен

Примечание: прототипы функций из файла math.h

int abs(int i);	*- модуль числа*
double cos(double x); **double sin(double x);**	*- тригонометрические функции – косинус, синус*
double exp(double x);	*- экспоненциальная функция*
double log(double x);	*- натуральный логарифм*
double pow(double x, double y);	*- x^y*
double sqrt(double x);	*- квадратный корень из x*

Инкремент и декремент

Задание 8. В программе **inc_decr.cpp** реализовать ввод целого числа и вывод в таблице следующих данных (рисунок 2).

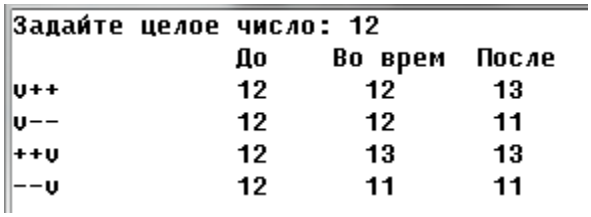

Рисунок 2 - Вид выполненного задания 8

Условная операция

Задание 9. 1 Сохранить файл **arifm_oper.cpp** с именем **usl_oper.cpp**. Изменить в файле **usl_oper.cpp** вывод значений следующим образом (рисунок 3). 2 Сохраните файл с именем **random.cpp**. Измените его таким образом, чтобы числа генерировались произвольно из диапазона от 0 до 99.

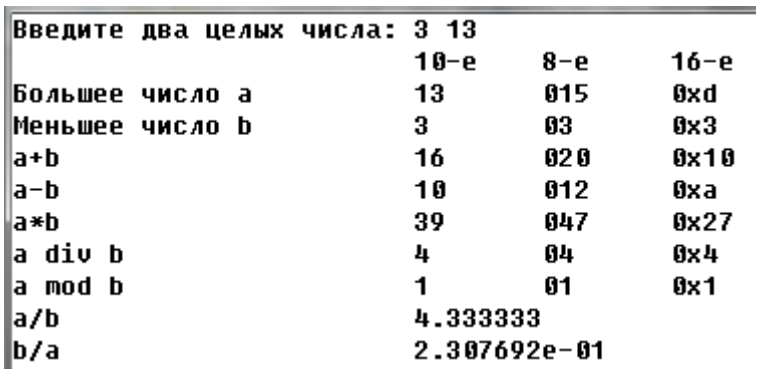

Рисунок 3 - Вид выполненного задания 3

Задание 10. Решить следующие задачи, используя условную операцию (назовите файлы **z10_номер задачи.cpp**):

1 Если введенное число четное, то уменьшить его в 2 раза, иначе уменьшить на 5.
2 Ввести два числа a и b, если оба числа отрицательные, то найти разность модулей a и b, если оба числа положительные, то найти их сумму, иначе найти модуль разности a и b.
3 Решить задачу по вариантам (таблица 2). Требуется вывести «Истина», если приведенное высказывание для предложенных исходных данных является истинным, и «Ложь» - в противном случае. Все числа, для которых указано количество цифр (двузначное число, трехзначное число и т.д.), считаются целыми.

Таблица 2 - Задачи по вариантам к заданию 10.3

Вар.	Проверить истинность высказывания:
1	Квадратное уравнение $A \cdot x^2 + B \cdot x + C = 0$ с данными коэффициентами A, B, C имеет вещественные корни
2	Данные числа x, y являются координатами точки, лежащей во второй координатной четверти
3	Данные числа x, y являются координатами точки, лежащей в первой или третьей координатной четверти
4	Точка с координатами (x, y) лежит внутри прямоугольника, левая верхняя вершина которого имеет координаты (x1, y1), правая нижняя — (x2, y2), а стороны параллельны координатным осям
5	Данное целое число является четным двузначным числом

Продолжение таблицы 2

Вар.	Проверить истинность высказывания:
6	Данное целое число является нечетным трехзначным числом
7	Среди трех данных целых чисел есть хотя бы одна пара совпадающих
8	Среди трех данных целых чисел есть хотя бы одна пара взаимно противоположных
9	Сумма цифр данного трехзначного числа является четным числом
10	Сумма двух первых цифр данного четырехзначного числа равна сумме двух его последних цифр
11	Данное четырехзначное число читается одинаково слева направо и справа налево
12	Все цифры данного трехзначного числа различны
13	Цифры данного трехзначного числа образуют возрастающую последовательность
14	Цифры данного трехзначного числа образуют возрастающую или убывающую последовательность
15	Цифры данного трехзначного числа образуют арифметическую прогрессию
16	Цифры данного трехзначного числа образуют геометрическую прогрессию

Лабораторная работа № 3. Условные конструкции *(4 часа)*

Почитать в [1] стр. 15-17. Знать понятия «пустой оператор», «метка», «составной оператор», «блок». Знать и уметь использовать при решении задач условный оператор if…else и переключатель switch. Почитать в методическом пособии [1] стр. 21 – Оператор break, знать его использование в переключателях.

Задание 11. Используя условный оператор и оператор выбора, решить следующие задачи по вариантам (таблица 3) (назовите файлы **z11_номер задачи.cpp**).

Таблица 3 - Задачи по вариантам к заданию 11

Вар.	
1	1 Заменить наименьшее из трех чисел суммой двух других чисел. 2 Дано целое число в диапазоне 100 – 999. Вывести строку — словесное описание данного числа, например: 256 — «двести пятьдесят шесть», 814 — «восемьсот четырнадцать»
2	1 Дано целое число, лежащее в диапазоне от –999 до 999. Вывести строку — словесное описание данного числа вида «отрицательное двузначное число», «нулевое число», «положительное однозначное число» и т.д. 2 В восточном календаре принят 60-летний цикл, состоящий из 12-летних подциклов, обозначаемых названиями цвета: зеленый, красный, желтый, белый и черный. В каждом подцикле годы носят названия животных: крысы, коровы, тигра, зайца, дракона, змеи, лошади, овцы, обезьяны, курицы, собаки и свиньи. По номеру года вывести его название, если 1984 год был началом цикла — годом зеленой крысы

Продолжение таблицы 3

Вар.	
3	1 Заменить наибольшее из трех чисел разностью двух других чисел. 2 Даны два целых числа: D (день) и M (месяц), определяющие правильную дату невисокосного года. Вывести значения D и M для даты, предшествующей указанной
4	1 Из трех данных чисел выбрать наименьшее и наибольшее и заменить третье число из разностью. 2 Единицы длины пронумерованы следующим образом: 1 - дециметр, 2 - километр, 3 - метр, 4 - миллиметр, 5 - сантиметр. Дан номер единицы длины и длина отрезка L в этих единицах (вещественное число). Вывести длину данного отрезка в метрах
5	1 Перераспределить значения переменных X и Y так, чтобы в X оказалось меньшее из этих значений, а в Y — большее. 2 Единицы длины пронумерованы следующим образом: 1 - дециметр, 2 - километр, 3 - метр, 4 - миллиметр, 5 - сантиметр. Дан номер единицы длины N1, длина отрезка L в этих единицах N1 (вещественное число) и номер единицы измерения N2. Вывести длину данного отрезка в единицах измерения N2
6	1 Значения переменных X, Y, Z поменять местами так, чтобы они оказались упорядоченными по возрастанию. 2 Единицы массы пронумерованы следующим образом: 1 — килограмм, 2 — миллиграмм, 3 — грамм, 4 — тонна, 5 — центнер. Дан номер единицы массы и масса тела M в этих единицах (вещественное число). Вывести массу данного тела в килограммах
7	1 Дано целое число, лежащее в диапазоне от 1 до 9999. Вывести словесное описание данного числа вида «четное двузначное число», «нечетное четырехзначное число» и т.д. 2 Единицы массы пронумерованы следующим образом: 1 — килограмм, 2 — миллиграмм, 3 — грамм, 4 — тонна, 5 — центнер. Дан номер единицы массы N1, масса тела M в этих единицах N1 (вещественное число) и единица массы N2. Вывести массу данного тела в единицах массы N2
8	1 Даны две переменные целого типа: A и B. Если их значения не равны, то присвоить каждой переменной сумму этих значений, а если равны, то присвоить переменным нулевые значения. 2 Единицы измерения количества информации пронумерованы следующим образом: 1 - бит, 2 - байт, 3 - килобайт, 4 - мегабайт. Дан номер единицы измерения N1, количество информации K в единице измерения N1 (вещественное число) и номер единицы измерения N2, в которую нужно перевести K
9	1 Даны две переменные целого типа: A и B. Если их значения не равны, то присвоить каждой переменной максимальное из этих значений, а если равны, то присвоить переменным нулевые значения. 2 Робот может перемещаться в четырех направлениях («С» - север, «З» - запад, «Ю» - юг, «В» - восток) и принимать три цифровые команды: 0 - продолжать движение, 1 - поворот налево, 2 - поворот направо. Дан символ C - исходное направление робота и число N - посланная ему команда. Вывести направление робота после выполнения полученной команды

Продолжение таблицы 3

Вар.	
10	1 Даны три переменные: X, Y, Z. Если их значения упорядочены по убыванию, то удвоить их; в противном случае заменить значение каждой переменной на противоположное. 2 Локатор ориентирован на одну из сторон света («С» - север, «З» - запад, «Ю» - юг, «В» - восток) и может принимать три цифровые команды: 1 - поворот налево, 2 - поворот направо, 3 - поворот на 180 градусов. Дан символ C - исходная ориентация локатора и числа N1 и N2 - две посланные ему команды. Вывести ориентацию локатора после выполнения данных команд
11	1 Даны три переменные: X, Y, Z. Если их значения упорядочены по возрастанию или убыванию, то удвоить их; в противном случае заменить значение каждой переменной на противоположное. 2 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)
12	1 Даны целочисленные координаты точки на плоскости. Если точка не лежит на координатных осях, то вывести 0, если совпадает с началом координат - 1, если не совпадает с началом координат, но лежит на оси OX или OY, то вывести соответственно 2 или 3. 2 Элементы равнобедренного прямоугольного треугольника пронумерованы следующим образом: 1 - катет (a), 2 - гипотенуза (c), 3 - высота, опущенная на гипотенузу (h), 4 - площадь (S). Дан номер одного из этих элементов и его значение. Вывести значения остальных элементов данного треугольника (в том же порядке)
13	1 Даны вещественные координаты точки, не лежащей на координатных осях OX и OY. Вывести номер координатной четверти, в которой находится данная точка. 2 Элементы равностороннего треугольника пронумерованы следующим образом: 1 - сторона (a), 2 - радиус вписанной окружности (R1), 3 - радиус описанной окружности (R2), 4 - площадь (S). Дан номер одного из этих элементов и его значение. Вывести значения остальных элементов данного треугольника (в том же порядке)
14	1 На числовой оси расположены три точки: A, B, C. Определить, какая из двух последних точек (B или C) расположена ближе к A, и вывести эту точку и ее расстояние от точки A. 2 Арифметические действия над числами пронумерованы следующим образом: 1 - сложение, 2 - вычитание, 3 - умножение, 4 - деление. Дан номер действия и два числа A и B (B не равно нулю). Выполнить над числами указанное действие и вывести результат
15	1 Значения переменных X, Y, Z поменять местами так, чтобы они оказались упорядоченными по убыванию. 2 Даны два целых числа: D (день) и M (месяц), определяющие правильную дату невисокосного года. Вывести значения D и M для даты, следующей за указанной

Продолжение таблицы 3

Вар.	
16	1 Дан номер некоторого года (положительное целое число). Вывести соответствующий ему номер столетия, учитывая, что, к примеру, началом 20 столетия был 1901 год. 2 Дано целое число в диапазоне 20–69, определяющее возраст (в годах). Вывести строку — словесное описание указанного возраста, обеспечив правильное согласование числа со словом «год», например: 20 — «двадцать лет», 32 — «тридцать два года», 41 — «сорок один год»

3 Для данного x вычислить значение функции f, заданной следующим образом

1	2	3	4
$\begin{cases} -1, x \le 0 \\ x^2, 0 < x \le 10 \\ (x-10)^2, x > 10 \end{cases}$	$\begin{cases} 0, x < 0 \\ 1, x \in [0,1), [2,3), \ldots \\ -1, x \in [1,2), [3,4), \ldots \end{cases}$	$\begin{cases} x^2, x < 0 \\ 2^x, 0 \le x \le 10 \\ \sqrt{x}, 10 < x \le 100 \\ \ln x, x > 100 \end{cases}$	$\begin{cases} 2^{-x}, x < 0 \\ e^{-x}, 0 \le x \le 10 \\ \sqrt{x}, 10 < x \le 100 \\ \ln x^{\sqrt{x}}, x > 100 \end{cases}$
5	6	7	8
$\begin{cases} \sqrt{\lvert x \rvert}, x < 0 \\ \frac{1}{x^2}, x \in [0,20) \\ \ln x, x \in [20,100) \\ 0, x > 100 \end{cases}$	$\begin{cases} 2^{x^2}, x < 0 \\ e^{-x}, 0 \le x \le 10 \\ \ln x^{\sqrt{x}}, 10 < x \le 100 \\ \frac{\sqrt{x}}{x^2 - x - 2}, x > 100 \end{cases}$	$\begin{cases} \frac{1}{x^2}, x < 0 \\ e^x, 0 \le x \le 10 \\ \ln x, 10 < x \le 100 \\ \sqrt{x}, x > 100 \end{cases}$	$\begin{cases} \frac{3 - x^2}{x^3}, x < 0 \\ e^{x-2}, 0 \le x \le 10 \\ x^2, 10 < x \le 100 \\ \sqrt{x^2 - x}, x > 100 \end{cases}$
9	10	11	12
$\begin{cases} 2^{x^2}, x < 0 \\ e^x, 0 \le x \le 10 \\ \frac{\cos\sqrt{x}}{x-2}, 10 < x \le 1 \\ \ln \pi^{\sqrt{x}}, x > 100 \end{cases}$	$\begin{cases} \frac{2^x}{\sqrt{\lvert x \rvert}}, x < 0 \\ \frac{\ln x}{x^2}, x \in [0,20) \\ \sqrt{2x}, x \in [20,100) \\ e^{\sqrt{x}}, x > 100 \end{cases}$	$\begin{cases} \frac{x^2}{e^{-x}}, x < 0 \\ 2^{e^x}, 0 \le x \le 10 \\ \sqrt{x}, 10 < x \le 100 \\ \ln x, x > 100 \end{cases}$	$\begin{cases} e^{\lvert x \rvert}, x < 0 \\ x^{-x}, 0 \le x \le 10 \\ \sqrt{\frac{x}{2}}, 10 < x \le 100 \\ \pi^{\sqrt{x}}, x > 100 \end{cases}$
13	14	15	16
$\begin{cases} \frac{2^x}{\sqrt{\lvert x \rvert}}, x < 0 \\ \frac{1}{x^2}, x \in [0,20) \\ \ln x, x \in [20,100) \\ \sqrt{x}, x > 100 \end{cases}$	$\begin{cases} \sqrt{\lvert x \rvert}, x < 0 \\ \frac{1}{x^2}, x \in [0,20) \\ \ln \pi^{x^2}, x \in [20,100) \\ \frac{\ln x}{\sqrt{x}}, x > 100 \end{cases}$	$\begin{cases} \lvert x \rvert, x < 0 \\ \frac{1}{x}, x \in [0,10) \\ x^2, x \in [10,20) \\ \sqrt{x}, x \in [20,100) \\ 0, x > 100 \end{cases}$	$\begin{cases} 2^{x^2}, x < 0 \\ 3^{-x}, 0 \le x \le 10 \\ \frac{\sqrt{x}}{x^3 - 2}, 10 < x \le 100 \\ x^{\sqrt{x}}, x > 100 \end{cases}$

Лабораторная работа № 4. Операторы цикла *(4 часа)*

Почитать в [1] стр. 17–24. Знать три конструкции цикла, операторы передачи управления.

Задание 12. Используя операторы цикла, решить задачи (таблица 4) (назовите файлы **z12_номер задачи.cpp**).

Таблица 4 - Задачи по вариантам к заданию 12

Вариант	1 Задан диапазон значений переменной x от N до M (N, M – целые числа). Вывести таблицу значений функции с шагом Δ=0,5	2 Дано натуральное число N. Вывести первые N членов ряда, заданных формулой i-го члена, и рассчитать их сумму. Вывести все члены ряда, большие некоторого числа e (<0,01)
1	$f(x)=x^3$	$a_i=(-1)^i\frac{1}{i}$
2	$f(x)=\frac{1}{x}$	$a_i=(-1)^{i+1}\frac{1}{i^2}$
3	$f(x)=e^x$	$a_i=\frac{1}{\sqrt{i}}$
4	$f(x)=sinx$	$a_i=e^{-i}$
5	$f(x)=cosx$	$a_i=2^{-i}$
6	$f(x)=lnx$	$a_i=(-1)^i 2^{-i}$
7	$f(x)=\sqrt{x}$	$a_i=(-1)^{i+1}e^{-i}$
8	$f(x)=\frac{1}{\sqrt{x}}$	$a_i=\frac{1}{i^2}$
9	$f(x)=\frac{x+1}{x-1}$	$a_i=\frac{1}{i^3}$
10	$f(x)=sin2x$	$a_i=\frac{(-1)^i}{\sqrt{i}}$
11	$f(x)=\frac{1}{x^2}$	$a_i=\frac{1}{e^i}$
12	$f(x)=x^2$	$a_i=\frac{(-1)^i}{i^i}$
13	$f(x)=2^x$	$a_i=\frac{(-1)^{i+1}}{i^3}$
14	$f(x)=\sqrt{x^2+x+1}$	$a_i=\frac{(-1)^i}{e^i}$
15	$f(x)=\frac{1}{\sin x}$	$a_i=\frac{1}{i}$
16	$f(x)=sinx^2$	$a_i=\frac{1}{i^i}$

3 В [1] стр. 51-53 выполнить задачи по вариантам.

Лабораторная работа № 5. Массивы. Одномерные массивы *(2 часа)*

Указатель - это переменная, содержащая адрес другой переменной, т.е. значением переменной типа указатель является целое число, равное адресу того объекта, на который ссылается указатель.

Указатель связан с типом объекта, на который он ссылается. Если в описании перед обозначением объекта поставить символ «*», то оно будет описывать указатель на объект того же типа и класса памяти, которые соответствуют данному обозначению без звездочки.

Унарная операция «*», называемая **операцией косвенной адресации**, рассматривает свой операнд как адрес объекта и обращается по этому адресу, возвращая его содержимое.

Унарная операция «**&**», называемая **операцией нахождения адреса**, будучи примененной к переменной, возвращает ее адрес.

Например, рассмотрим последовательность операторов:

```
int x,y,*px;  // Описание целочисленных переменных x,y и
                 // указателя на целое значение px.
px = &x;     // Значением переменной px станет адрес переменной x.
y  = *px;    // Переменная y приобретает значение «того»,
                // на что указывает px, т.е. значение переменной x.
```

Таким образом, два оператора присваивания в примере эквивалентны одному оператору **y=x;**, и ***px** может появляться в любом выражении, в котором может встретиться **x**, например, продолжим рассмотрение предыдущего примера:

```
*px = 0;         // Переменная x получает значение 0.
y = *px + 1; // Переменная y получает значение на 1 большее значения x.
*px += 1;    // Увеличение содержимого x на единицу.
*px++;       // Увеличение содержимого x на единицу, при
                // этом постфиксная операция ++ не изменяет px,
                // пока объект по адресу px не будет получен.
*++px;       // Префиксная операция ++(--) увеличивает
*--px;       // (уменьшает) px до получения значения x.
py = px;     // Копирование содержимого указателя px в указатель py
                // в результате чего py указывает на то же, что и px.
```

Массив представляет собой последовательность элементов одного типа. Каждому элементу массива соответствует индекс - целое неотрицательное число, определяющее его номер в последовательности. Первому элементу массива соответствует индекс 0. Элементы массива размещаются в памяти последовательно, друг за другом.

При определении массива ему выделяется память. Но как только память для массива выделена, имя массива воспринимается как константный указатель того типа, к которому отнесены элементы массива.

Для создания массива компилятору необходимо знать тип данных и требуемый класс памяти, т.е. то же самое, что и для простой переменной. Кроме того, должно быть известно, сколько элементов имеет массив. Определение одномерного массива имеет вид:

<тип элементов> <имя массива> [<количество элементов в массиве>];

В языке C++ определены только одномерные массивы. Многомерные массивы строятся на основе рекурсивного правила, согласно которому элементом массива может быть массив. Таким образом, в языке C++ двумерный массив (матри-

ца) - это одномерный массив, каждый элемент которого является одномерным массивом.

При определении массива может выполняться его инициализация, то есть элементы массива получают конкретные значения. Инициализация выполняется по умолчанию, если массив статический или внешний. В этих случаях всем элементам массива компилятор автоматически присваивает нулевые значения. Явная инициализация элементов массива разрешена только при его определении и возможна двумя способами: либо с указанием размера массива в квадратных скобках, либо без явного указания (без конкретного выражения) в квадратных скобках, например:

```
char CH[ ] = { 'A', 'B', 'C', 'D'}; // Массив из 4 элементов.
int pr[6]= {10,20,30,40};          // Массив из 6 элементов.
char St[ ] = "ABCD";    // Массив из 5 элементов (включая нулевой).
```

Пример 1. Инициализация целочисленного массива внешнего класса памяти.

```
#include <iostream.h>
  int n=5;
  int nd[5] = { 0,1,2,3,4 };     /* Инициализация целочисленного   */
                                 /* массива внешнего класса памяти */
main ()
{
  for (int i=0; i<=n; i++)
    cout << "i=" << i << " nd[i]=" << nd[i] << endl;
}
```

Следующие примеры являются ***ошибочными***:

```
float A[ ];  // Отсутствует размер массива.
double b[4] = {1,2,3,4,5,6,7,8};  // Количество элементов
                                  //не совпадает с размером массива.
```

В тех случаях, когда массив не определяется, а описывается, список начальных значений задавать нельзя. В описании массива может отсутствовать и его размер:

```
    extern float E[ ]; // Правильное описание внешнего массива.
```

Предполагается, что в месте определения массива для него выделена память и выполнена инициализация.

Элементы массива можно задавать в цикле при выполнении программы.

Пример 2. Ввод элементов массива при выполнении программы.

```
#include <iostream.h>
main ()
{
  int score[10];
  /* Ввод элементов массива. */
  for  (int i=0; i<=9; i++)
    {  cout << i+1 << "-й элемент: ";
       cin >> score[i];      }
  /* Проверка правильности ввода. */
  cout << "Введены следующие значения:\n");
  for  (i=0; i<=9; i++)     cout << score[i] << " ";
  cout << endl;
}
```

Имя массива является указателем-константой, значением которой служит **адрес первого элемента массива** (с индексом 0). Таким образом, доступ к первому элементу массива может быть осуществлен так:

<имя массива>[0]

или

***<имя массива>**

В общем случае **доступ к заданному элементу массива** можно осуществить двумя способами:

<имя массива>[<номер элемента>]

или

***(<имя массива>+<номер элемента>).**

Проиллюстрируем различные способы доступа к элементам массива.

Пример 3. Вывести на экран заданную строку несколькими способами.

```
#include <iostream.h>
void main()
{
  char x[]=«Пример строки»;
  int i=0;
//стандартный доступ к первому элементу массива
   cout<< “\n Вывод первого символа строки (1 способ): “<< x[0];
//*x = *(x+0) = x[0], все конструкции вида x[i] перед выполнением
//преобразуются компилятором к виду: *(x+i)
   cout<< “\n Вывод первого символа строки (2 способ):  “<< *x;
//традиционный способ вывода значения переменной
   cout<< “\n Вывод строки без цикла (1 способ):  “<< x;
//для символьного массива его имя содержит адрес первого элемента
// массива, следовательно, запись x эквивалентна &x[0]
   cout<< “\n Вывод строки без цикла (2 способ): “<< &x[0];
//стандартный доступ к элементам массива
   cout<< “\n Вывод строки с  циклом (1 способ):  “;
   while (x[i]!='\0')
          cout << x[i++];
//здесь используется преобразование x[i++] = *(x+i++).
   cout<< “\n Вывод строки с  циклом (2 способ):  “;
   i=0;
   while (*(x+i)!='\0')
         cout << *(x+i++);
//здесь используется преобразование x[i++] = *(x+i++) = *(i++ + x) = i++[x]
   cout<<”\n Вывод строки с  циклом (3 способ): “;            //7
   i=0;
   while (i[x]!='\0')
          cout << i++[x];
}
```

В языке С++ нет специального типа данных «строка». Вместо этого каждая символьная строка в памяти компьютера представляется в виде одномерного массива типа **char**, последним элементом которого является символ **'\0'**. Изображение строковой константы может использоваться по-разному. Если строка применяется для инициализации массива типа **char**, например, так: **char array[ ] =**

“инициализирующая строка”;, то адрес первого элемента строки становится значением указателя-константы (имени массива) **array**. Если строка используется для инициализации указателя типа **char *: char *pointer = “инициализирующая строка”;**, то адрес первого элемента строки становится значением указателя-переменной **pointer**. Если использовать строку в выражении, где разрешено применять указатель, то используется адрес первого элемента строки:

```
char * string;
string = “строковый литерал”;
```

В данном примере значением указателя **string** будет не вся строка «строковый литерал», а только адрес ее первого элемента.

Задание 13. Используя сведения о массивах, решите следующие задачи (таблица 5). *Примечание: **серией** назовем группу подряд идущих одинаковых элементов, а **длиной серии** — количество этих элементов (длина серии может быть равна 1)* (назовите файлы **z13 _номер задачи.cpp**).

Таблица 5 - Задачи по вариантам к заданию 13

Вар.	
1	1 Дан массив размера N. Вывести его элементы в обратном порядке. 2 Дан целочисленный массив размера N. Определить максимальное количество его одинаковых элементов
2	1 Дан массив размера N. Вывести вначале его элементы с четными индексами, а затем — с нечетными. 2 Дан целочисленный массив размера N. Удалить из массива все элементы, встречающиеся менее двух раз
3	1 Дан целочисленный массив A. Вывести номер первого из тех его элементов A[i], которые удовлетворяют двойному неравенству: A[1] < A[i] < A[10]. Если таких элементов нет, то вывести 0. 2 Дан целочисленный массив размера N. Если он является перестановкой, то есть содержит все числа от 1 до N, то вывести 0, в противном случае вывести номер первого недопустимого элемента
4	1 Дан целочисленный массив размера N. Преобразовать его, прибавив к четным числам первый элемент. Первый и последний элементы массива не изменять. 2 Дан массив размера N. Преобразовать его, вставив перед каждым положительным элементом нулевой элемент
5	1 Дан целочисленный массив размера N. Вывести вначале все его четные элементы, а затем — нечетные. 2 Дан целочисленный массив размера N. Вывести массив, содержащий длины всех серий исходного массива
6	1 Поменять местами минимальный и максимальный элементы массива 2 Дан целочисленный массив размера N. Преобразовать массив, увеличив каждую его серию на один элемент
7	1 Заменить все положительные элементы целочисленного массива на значение минимального. 2 Дан целочисленный массив размера N. Преобразовать массив, увеличив первую серию наибольшей длины на один элемент

Продолжение таблицы 5

Вар.	
8	1 Переставить в обратном порядке элементы массива, расположенные между его минимальным и максимальным элементами 2 Дано число k и целочисленный массив размера N. Поменять местами последнюю и k-ю серии массива. Если серий в массиве меньше k, то вывести массив без изменений
9	1 Дан массив размера N и число k (0 < k < 5, k < N). Осуществить циклический сдвиг элементов массива влево на k позиций. 2 Дано число k и целочисленный массив размера N. Удалить из массива все серии, длина которых больше k
10	1 Проверить, образуют ли элементы целочисленного массива размера N арифметическую прогрессию. Если да, то вывести разность прогрессии, если нет — вывести 0. 2 Дано число k и целочисленный массив размера N. Заменить серию, длина которой меньше k, на один нулевой элемент
11	1 Дан массив ненулевых целых чисел размера N. Проверить, чередуются ли в нем положительные и отрицательные числа. Если чередуются, то вывести 0, если нет, то вывести номер первого элемента, нарушающего закономерность. 2 Даны два массива A и B размера 5, элементы которых упорядочены по убыванию. Объединить эти массивы так, чтобы результирующий массив остался упорядоченным
12	1 Дан массив размера N. Определить количество участков, на которых его элементы монотонно возрастают. 2 Дано число k и целочисленный массив размера N. Поменять местами последнюю и k-ю серии массива. Если серий в массиве меньше k, то вывести массив без изменений
13	1 Дано вещественное число R и массив размера N. Найти элемент массива, который наиболее близок к данному числу. 2 Дан массив размера N. Вывести индексы массива в том порядке, в котором соответствующие им элементы образуют возрастающую последовательность
14	1 Дан массив размера N. Осуществить циклический сдвиг элементов массива вправо на одну позицию. 2 Упорядочить массив размера N по убыванию методом выбора

Лабораторная работа № 6. Массивы. Многомерные массивы *(2 часа)*

Многомерный массив представляет собой массив массивов, то есть массив, элементами которого служат массивы.

Определение многомерного массива в общем случае должно содержать сведения о типе, размерности и количествах элементов каждой размерности, например описание: **int ARRAY[4][3][6];** определяет массив, состоящий из четырех элементов, каждый из которых двухмерный массив с размерами 3 на 6. В памяти массив **ARRAY** размещается в порядке возрастания самого правого индекса, то есть самый младший адрес имеет элемент **ARRAY[0][0][0]**, затем идет

элемент **ARRAY[0][0][1]** и т.д.

С учетом порядка расположения в памяти элементов многомерного массива нужно размещать начальные значения его элементов в списке инициализации. Например, конструкция **int ARRAY [4][3][6] = {0,1,2,3,4,5,6,7}** инициализирует только первые 8 элементов этого массива: **ARRAY[0][0][0]=0**, **ARRAY[0][0][1]=1**, **ARRAY[0][0][2]=2**, **ARRAY[0][0][3]=3**, **ARRAY[0][0][4]=4**, **ARRAY[0][0][5]=5**, **ARRAY[0][1][0]=6**, **ARRAY[0][1][1]=7**.

При необходимости инициализировать только часть элементов многомерного массива, если они размещены не в его начале или не подряд, то можно вводить дополнительные фигурные скобки, каждая пара которых выделяет последовательность значений, относящихся к одной размерности. Нельзя использовать скобки без информации внутри них.

Следующее определение с инициализацией трехмерного массива:

```
int A[4][5][6] = { {  {0} /*Нулевой элемент.*/ },/*Нулевой массив.*/
                 {  {100},/*Нулевой элемент.*/
                    {110, 111} /*Первый элемент.*/
                                 }, /*Первый массив.*/
                 {  {200),/*Нулевой элемент.*/
                    {210},/*Первый элемент.*/
                    {220, 221, 222}/*Второй элемент.*/
                                 }; /*Второй массив.*/
```

Таким образом будут заданы следующие значения элементов массива A:

```
A[0][0][0]=0,
A[1][0][0]=100, A[1][1][0]=110, A[1][1][1]=111,
A[2][0][0]=200, A[2][1][0]=210, A[2][2][0]=220,
                          A[2][2][1]=221, A[2][2][2]=222.
```

Как и в случае одномерных массивов, доступ к элементам многомерных массивов возможен с помощью индексированных переменных и с помощью указателей. Следующая программа иллюстрирует оба подхода к организации доступа к элементам массива.

```
#include <iostream.h>
void main()
{  int b[3][2][4] = {  0,  1,  2,   3,
                          4,  5,  6,   7,
                       10, 11, 12,  13,
                       14, 15, 16,  17,
                      100, 101, 102, 103,
                      104, 105, 106, 107  };
 cout<< "\n b= " << b; //Адрес массива b[ ][ ][ ].
 cout<< "\n *b= " << *b; // Адрес массива b[0][ ][ ].
 cout<< "\n **b= " << **b; // Адрес массива b[0][0][].
 cout<< "\n ***b= " << ***b; // Элемент b[0][0][0].
 cout<< "\n *(b+1) = " << *(b+1); // Адрес массива  b[1][ ][ ].
 cout<< "\n *(b+2) = " << *(b+2); // Адрес массива b[2][][].
 cout<< "\n *(*b+1)= " << *(*b+1); // Адрес массива b[0][1][ ].
 //Вывод элемента b[1][1][1] двумя способами.
 cout<< "\n *(*(*(b+1)+1)+1)= " << *(*(*(b+1)+1)+1);
 cout<< "\n b[1][1][1]= " << b[1][1][1];    }
```

В общем случае для трехмерного массива индексированный элемент b[i][j][k] соответствует выражению *(*(*(b + i) + j) + k).

Задание 14. Используя сведения о многомерных массивах, решите следующие задачи (таблица 6) (назовите файлы **z14_номер задачи.cpp**).

Таблица 6 - Задачи по вариантам к заданию 14

Вар.	
1	1 Дано число k (0 < k < 11) и матрица размера m x n. Найти сумму и произведение элементов k-го столбца данной матрицы. 2 Дана целочисленная матрица размера M x N. Различные строки (столбцы) матрицы назовем похожими, если совпадают множества чисел, встречающихся в этих строках (столбцах). Найти количество строк, похожих на первую строку
2	1 Дана матрица размера m x n. Найти суммы элементов всех ее четных столбцов. 2 Дана целочисленная матрица размера M x N. Найти количество ее строк, все элементы которых различны
3	1 Дана матрица размера m x n. Найти минимальное значение в строках. 2 Дана целочисленная матрица размера M x N. Вывести номер ее первого столбца, содержащего максимальное количество одинаковых элементов
4	1 Дана матрица размера m x n. В каждой строке найти количество элементов, больших среднего арифметического всех элементов этой строки. 2 Дана квадратная матрица порядка M. Найти суммы элементов ее диагоналей, параллельных главной (начиная с диагонали A[1,M])
5	1 Дана матрица размера m x n. Преобразовать матрицу, поменяв местами минимальный и максимальный элемент в каждой строке. 2 Дана квадратная матрица порядка M. Вывести максимальные из элементов каждой ее диагонали, параллельной побочной (начиная с диагонали A[1,1])
6	1 Дана матрица размера m x n. Найти минимальное значение среди сумм элементов всех ее столбцов и номер столбца с минимальным значением. 2 Дана квадратная матрица порядка M. Зеркально отразить ее элементы относительно главной диагонали
7	1 Дана матрица размера m x n. Найти максимальный среди минимальных элементов каждой строки. 2 Дана квадратная матрица порядка M. Повернуть ее на 90 градусов в положительном направлении
8	1 Дана целочисленная матрица размера m x n. Вывести номер ее последнего столбца, содержащего равное количество положительных и отрицательных элементов (нулевые элементы не учитываются). 2 Дана квадратная матрица порядка M. Повернуть ее на 90 градусов против часовой стрелки
9	1 Дана матрица размера m x n. Вывести номер ее первой строки, содержащей только положительные элементы. Если таких строк нет, то вывести 0. 2 Дана квадратная матрица порядка M. Повернуть ее на 180 градусов в положительном направлении

Продолжение таблицы 6

Вар.	
10	1 Дана матрица размера m x n. Найти ее седловую точку (максимальную в строке и минимальную в столбце). 2 Дана квадратная матрица порядка M. Повернуть ее на 270 градусов против часовой стрелки
11	1 Дана квадратная матрица порядка M. Найти сумму элементов ее главной диагонали. 2 Дана квадратная матрица порядка M. Зеркально отразить ее элементы относительно горизонтальной оси симметрии
12	1 Дана квадратная матрица порядка M. Заменить нулями элементы матрицы, лежащие выше побочной диагонали. 2 Дана квадратная матрица порядка M. Зеркально отразить ее элементы относительно вертикальной оси симметрии
13	1 Дана квадратная матрица порядка M. Заменить нулями элементы, лежащие ниже главной диагонали, если их значение меньше значения элемента, стоящего на главной диагонали, иначе заменить значение этого элемента на сумму его и элемента главной диагонали. 2 Дана квадратная матрица порядка M. Зеркально отразить ее элементы относительно побочной диагонали
14	1 Дана квадратная матрица порядка M. Найти среди симметричных относительно главной диагонали элементов максимальные и их значениями заменить все элементы, лежащие выше главной диагонали, а если заменяемый элемент сам является максимальным, то его удвоить. 2 Дана целочисленная матрица размера m x n. Вывести номер ее первой строки, содержащей максимальное количество положительных элементов (нулевые элементы не учитываются). Если таких строк нет, то вывести 0

3 В [1] стр. 53-54 выполнить задачи по вариантам.

Лабораторная работа № 7. Функции *(4 часа)*

Функция - самостоятельная единица программы, спроектированная для реализации конкретной задачи. Функции в языке C++ играют ту же роль, какую играют функции, подпрограммы и процедуры в других языках, хотя детали их структуры могут быть разными. Вызов функции приводит к выполнению некоторых действий. Например, при обращении к функции **printf()** осуществляется вывод данных на экран. Другие же функции позволяют получать величины, используемые затем в других частях программы.

Напомним, что программа на C++ состоит из функций, среди которых обязательно должна присутствовать функция с именем **main()** (главная функция). Она обеспечивает задание точки входа в откомпилированную программу. Всем именам функций по умолчанию присваивается класс памяти **extern**, то есть каждая функция имеет внешний тип компоновки и статическую продолжительность существования. Как объект с классом памяти **extern**, каждая функция глобальна, то есть при определенных условиях доступна в модуле и даже во всех модулях программы. Для доступности в модуле функция должна быть в нем ***определена или описана до первого вызова***. В заголовке функции последовательно указываются:

- ***тип функции***;
- ***имя функции;***;
- ***совокупность формальных параметров (аргументов), которая называется сигнатурой функции.***;

Таким образом, определение функции имеет следующий формат:

<тип функции> <имя функции> (<спецификация формальных параметров>)
{
<тело функции>;
}

Здесь ***тип функции*** - тип возвращаемого функцией значения, в том числе **void**, если функция никакого значения не возвращает. ***Имя функции*** - это идентификатор. Имена функций как имена внешние должны быть уникальными среди других имен из модулей, в которых используются функции. ***Спецификация формальных параметров*** - это либо пусто, либо **void**, либо список спецификаций отдельных параметров, в конце которого может быть поставлено многоточие. Спецификация каждого параметра в определении функции имеет вид:

<тип параметра> <имя параметра> = <значение по умолчанию>,...

Тело функции - это всегда блок или составной оператор, то есть последовательность описаний и операторов, заключенная в фигурные скобки. Возврат из функции осуществляется с помощью оператора **return <выражение>;**, где ***выражение*** определяет возвращаемое функцией значение. Если функция не возвращает никакого значения, то есть имеет тип **void**, то выражение в операторе **return** опускается. В этом случае необязателен и оператор **return** в теле функции.

Строгое согласование по типам между формальными и фактическими параметрами требует, чтобы в модуле до первого обращения к функции было помещено либо ее определение, либо ее описание (прототип), содержащее сведения о ее типе (о типе результата, то есть возвращаемого значения) и о типах всех параметров. Именно наличие такого прототипа либо полного определения позволяет компилятору выполнять контроль соответствия типов параметров. Прототип функции может внешне почти полностью совпадать с заголовком ее определения:

<тип функции> <имя функции> (<спецификация формальных параметров>);

Основное различие - точка с запятой в конце описания (прототипа). Второе отличие - необязательность имен формальных параметров в прототипе даже тогда, когда они есть в заголовке определения функции.

Пример 1. Программа для введенного числа с плавающей точкой выводит ему противоположное. Реализация без прототипа.

```
#include <iostream.h>
float protiv (float a) //Функция расположена до функции main()
{      return -a;     }
void main()
{  float x;
    do
   {   cout << "\nЗадайте число (0 - выход из программы): ";
       cin >> x;
       cout << "Противоположным этому числу является: " << protiv(x);
   }
   while (x!=0);
}
```

Для данного случая функция **protiv()** расположена в тексте программы до функции **main()**. Поэтому функция **main()** «знает», что есть такая функция **protiv()**, «знает», сколько у нее параметров и значение какого типа эта функция возвращает. Если же эта функция была бы расположена за функцией **main()**, то нужно было бы использовать прототип, который бы «подсказал» функции **main()** характеристики функции **protiv()**.

Пример 2. Программа для введенного числа с плавающей точкой выводит ему противоположное. Реализация с прототипом.

```
#include <iostream.h>
void main()
{   float x;
    float protiv (float); //Прототип функции.
    do
    {     cout << "\nЗадайте число (0 - выход из программы): ";
          cin >> x;
          cout << "Противоположным этому числу является: " << protiv(x);
    }
    while (x!=0);
}
float protiv (float a)     //Функция расположена после функции main()
{   return -a;        }
```

Как видно из приведенных примеров, обращение к функции (вызов функции) осуществляется с использованием следующей конструкции:

<имя функции> (<список фактических параметров>);

В рассмотренных примерах параметры в функцию передаются по значению, но можно передавать параметры с использованием указателей. Рассмотрим пример.

Пример 3. Поменять значения местами.

```
#include <iostream.h>
void interchange (int *u,int *v)
{       int p;
        p = *u; *u = *v; *v = p;
}
void main ()
{    int x = 1, y = 3;
     cout << "Имели...     x= " << x << " y = " << y << endl;
     interchange (&x,&y);  //передаются адреса переменных
     cout << "Получили...  x= " << x << " y = " << y << endl;
}
```

Комментарии к примеру. При вызове функции **interchange**() с аргументами **&x,&y** вместо передачи значений **x** и **y** мы передаем их адреса. Естественно, что формальные аргументы **u** и **v**, имеющиеся в определении функции **interchange**(), должны быть описаны как указатели.

Поскольку **x** и **y** - переменные целого типа, то **u** и **v** должны быть указателями на переменные целого типа.

Оператор описания **int p;** используется с целью резервирования памяти. Мы хотим поместить значение переменной **x** в переменную **p**, поэтому пишем: **p = *u;.** Вспомните, что значение переменной **u** - это **&x**, поэтому переменная **u** ссылается

на **x**. Это означает, что операция ***u** дает значение **x**, которое как раз нам и требуется. Аналогично, оператор ***u=*v;** соответствует оператору **x=y;**.

Можно передавать параметры в функцию, используя ссылки.

Объявив **ссылочную переменную**, можно создать объект, который, как указатель, ссылается на другое значение, но, в отличие от указателя, **постоянно привязан к этому значению**. Таким образом, ссылка на значение всегда ссылается на это значение.

Ссылку можно объявить следующим образом:

<имя типа>& <имя ссылки> = <выражение>;

или

<имя типа>& <имя ссылки>(<выражение>);

Раз ссылка является **другим именем уже существующего объекта**, то в качестве инициализирующего объекта должно выступать **имя некоторого объекта, уже расположенного в памяти**. Значением ссылки после выполнения соответствующего определения с инициализацией становится **адрес этого объекта**. Проиллюстрируем это на конкретном примере.

Пример 4. Использование ссылок.

```
#include <iostream.h>
void main()
{
  int ivar = 1234;   //Переменной присвоено значение.
  int *iptr = &ivar; //Указателю присвоен адрес ivar.
  int &iref = ivar;  //Ссылка ассоциирована с ivar.
  int *p = &iref;    //Указателю присвоен адрес iref.
  cout << “ivar = “ << ivar << endl;
  cout << “*iptr = “ << *iptr << endl;
  cout << “iref = “ << iref << endl;
  cout << “*p = “ << *p << endl;
}
```

1 В отличие от указателей, которые могут быть объявлены неинициализированными или установлены в нуль (**NULL**), ссылки всегда ссылаются на объект. Для ссылок не существует аналога нулевого указателя.

2 Ссылки нельзя инициализировать в следующих случаях:

- при их объявлении с ключевым словом **extern**;
- при использовании в качестве параметров функции;
- при использовании в качестве типа возвращаемого значения функции;
- в объявлениях классов.

3 Не существует операторов, непосредственно производящих действия над ссылками!

Пример 5. Способы обмена значений с использованием указателей и ссылок.

```
#include <iostream.h>
void interchange_ptr (int *u,int *v) //Обмен с использованием указателей.
{     int temp=*u;
      *u = *v; *v = temp;
}
void interchange_ref (int &u,int &v) //Обмен с использованием ссылок.
{     int temp=u;
```

```
        u = v; v = temp;
}

void main ()
{    int x=5,y=10;
     cout << "Обмен с использованием указателей:\n";
     cout << "Вначале  x = " << x << " и  y = " << y << endl;
     interchange_ptr (&x,&y);
     cout << "Теперь  x = " << x << " и  y = " << y << endl;
     cout << "-------------------------------------" << endl;
     cout << "Обмен с использованием ссылок:\n»;
     cout << "Вначале  x = " << x << " и  y = " << y << endl;
     interchange_ref (x,y);
     cout << "Теперь  x = " << x << " и  y = " << y << endl;
}
```

Задание 15. Используя сведения о функциях, решите следующие задачи (таблица 7) (назовите файлы **z15_номер задачи.cpp**).

Таблица 7 - Задачи по вариантам к заданию 15

Вар.	
1	1 Описать функцию Min2(A,B) вещественного типа, находящую минимальное из двух вещественных чисел A и B. С помощью этой функции найти минимальное среди чисел A, B, C, D. 2 Описать функцию Exp1(x,eps) вещественного типа (параметры x, eps - вещественные, eps > 0), находящую приближенное значение функции exp(x): $\exp(x) = 1 + x + x^2 / 2! + x^3 / 3! + ... + x^n / n! + ...$. В сумме учитывать все слагаемые, большие eps. С помощью Exp1 найти приближенное значение экспоненты для данного x при шести произвольных eps
2	1 Описать функцию Max2(A,B) вещественного типа, находящую максимальное2из двух вещественных чисел A и B. С помощью этой функции найти максимальное из чисел A, B, C, D. 2 Описать функцию Sin1(x,eps) вещественного типа (параметры x, eps - вещественные, eps > 0), находящую приближенное значение функции $\sin(x) = x - x^3 / 3! + x^5 / 5! - ... + (-1)^n x^{2n+1} / (2n+1)! + ...$ В сумме учитывать все слагаемые, большие по модулю eps. С помощью Sin1 найти приближенное значение синуса для данного x при шести произвольных значениях eps
3	1 Описать функцию Minmax(A,B), записывающую в переменную A минимальное из значений A и B, а в переменную B - максимальное из этих значений (A и B - вещественные параметры, являющиеся одновременно входными и выходными). Используя эту функцию, найти минимальное и максимальное из чисел A, B, C, D. 2 Описать функцию Cos1(x,eps) вещественного типа (параметры x, eps - вещественные, eps > 0), находящую приближенное значение функции $\cos(x) = 1 - x^2 / 2! + x^4 / 4! - ... + (-1)^n x^{2n} / (2n)! + ...$ В сумме учитывать все слагаемые, большие по модулю eps. С помощью Cos1 найти приближенное значение косинуса для данного x при шести произвольных значениях eps

Продолжение таблицы 7

Вар.	
4	1 Описать функцию Fact(N) целого типа, вычисляющую значение факториала N! = 1·2·...·N (N > 0 - параметр целого типа). С помощью этой функции вычислить факториалы произвольных 10 чисел. 2 Описать функцию Ln1(x,n) вещественного типа (параметры x, eps — вещественные, |x| < 1, eps > 0), находящую приближенное значение функции $\ln(1+x) = x - x^2 / 2 + x^3 / 3 - ... + (-1)^n x^{n+1} / (n+1) + ...$ В сумме учитывать все слагаемые, большие по модулю eps. С помощью Ln1 найти приближенное значение ln(1+x) для данного x при шести произвольных значениях eps
5	1 Описать функцию FactR(N) вещественного типа, позволяющую вычислять приближенное значение факториала N! = 1·2·...·N для целых N (N > 0). С помощью этой функции вычислить факториалы пяти произвольных чисел. 2 Описать функцию Arctg1(x,n) вещественного типа (параметры x, eps — вещественные, |x| < 1, eps > 0), находящую приближенное значение функции $\text{arctg}(x) = x - x^3 / 3 + x^5 / 5 - ... + (-1)^n x^{2n+1} / (2n+1) + ...$ В сумме учитывать все слагаемые, большие по модулю eps. С помощью Arctg1 найти приближенное значение arctg(x) для данного x при шести произвольных значениях eps
6	1 Описать функцию Fact2(N) целого типа, вычисляющую значение «двойного факториала»: N!! = 1·3·5·...·N, если N - нечетное, N!! = 2·4·6·...·N, если N - четное (N > 0 - параметр целого типа). С помощью этой функции вычислить двойные факториалы десяти произвольных чисел. 2 Описать функцию PowerA(x,a,eps) вещественного типа (параметры x, a, eps - вещественные, |x| < 1, a > 0, eps > 0), находящую приближенное значение функции $(1+x)^a = 1 + a \cdot x + a \cdot (a-1) \cdot x^2 / 2! + ... + a \cdot (a-1) \cdot ... \cdot (a-n+1) \cdot x^n / n! + ...$. В сумме учитывать все слагаемые, большие по модулю eps. С помощью PowerA найти приближенное значение $(1+x)^a$ для данных x и a при шести произвольных значениях eps
7	1 Описать функцию Fib(N) целого типа, вычисляющую N-е число Фибоначчи F(N) по формуле: F(1) = F(2) = 1, F(k) = F(k–2) + F(k–1), k = 3, 4, ... С помощью этой функции вычислить 10 чисел Фибоначчи с указанными номерами. 2 Описать функцию Otr(Ax,Ay,Bx,By) вещественного типа, находящую длину отрезка AB на плоскости по координатам его концов: |AB| = sqrt((Ax-Bx)^2 + (Ay-By)^2) (Ax, Ay, Bx, By — вещественные параметры). С помощью этой функции найти длины отрезков AB, AC, AD, если даны координаты точек A, B, C, D
8	1 Описать функцию SumDigit(N), находящую сумму цифр целого числа N. Используя эту функцию, найти суммы цифр пяти произвольных чисел. 2 Описать функцию Perim(Ax,Ay,Bx,By, Cx,Cy) вещественного типа, находящую периметр треугольника ABC по координатам его вершин (Ax, Ay, Bx, By, Cx, Cy — вещественные параметры). С помощью этой функции найти периметры треугольников ABC, ABD, ACD, если даны координаты точек A, B, C, D

Продолжение таблицы 7

Вар.	
9	1 Описать функцию PDigit(N), находящую произведение цифр целого числа N. Используя эту функцию, найти произведение цифр пяти произвольных чисел. 2 Описать функцию Area(Ax,Ay,Bx,By, Cx,Cy) вещественного типа, находящую площадь треугольника ABC по формуле Герона: S_{ABC} = sqrt(p·(p–|AB|)·(p–|AC|)·(p–|BC|)), где p — полупериметр. С помощью этой функции найти площади треугольников ABC, ABD, ACD, если даны координаты точек A, B, C, D
10	1 Описать функцию NOD2(A,B) целого типа, находящую наибольший общий делитель двух натуральных чисел A и B, используя алгоритм Евклида: НОД(A,B) = НОД(B mod A,A), если A <> 0; НОД(0,B) = B. С помощью этой функции найти наибольшие общие делители пар A и B, A и C, A и D, если даны числа A, B, C, D. 2 Описать процедуру Dist(Px,Py,Ax,Ay, Bx,By,D), находящую расстояние D от точки P до прямой AB по формуле D = $2S_{PAB}$ / |AB|, где S_{PAB} — площадь треугольника PAB. С помощью этой процедуры найти расстояния от точки P до прямых AB, AC, BC, если даны координаты точек P, A, B, C.

Лабораторная работа № 8. Строки *(4 часа)*

Строковая константа - это последовательность из нуля или более символов, заключенных в кавычки. Кавычки не являются частью строковой константы, а служат только для ее ограничения.

Строки представляются в виде массива элементов типа **char**. Это означает, что символы строки можно представить расположенными в соседних ячейках памяти - по одному символу в ячейке. Но массив символов - не всегда строка!

В качестве примера рассмотрим следующую строку: «Символьная строка» (рисунок 4). Кавычки не являются частью строки. Они вводятся для того, чтобы отметить ее начало и конец, то есть играют ту же роль, что и апострофы в случае одиночного символа (каждая ячейка - 1 байт).

Рисунок 4 - Схема размещения в памяти элементов строки

Необходимо отметить, что на рисунке 4 последним элементом массива является символ '\0'. Это нуль-символ. В языке C++ он используется для того, чтобы отмечать конец строки. Нуль-символ - не есть цифра 0; он не выводится на печать и в таблице кодов ASCII имеет номер 0. Наличие нуль-символа означает, что количество ячеек массива должно быть по крайней мере на одну больше, чем количество символов, которые необходимо размещать в памяти.

Не следует путать символьную константу со строкой, содержащей один символ: 'X' - это не то же самое, что»X». В первом случае - это отдельный символ. Во втором случае - это строка, состоящая из одного символа (буквы X) и символа конца строки '\0'.

Пример 1.

```
#include <iostream.h>
int n=5;
char line[5] = { 'Д','о','м','!','\0' };/* Инициализация строки
                                        внешнего класса памяти*/
void main ()
{     cout << "Получили слово: ";
      for (int i=0; i<n; i++)  cout << line[i];
}
```

Пример 2. Инициализация строки с помощью строковой константы

```
#include <iostream.h>
int n=5;
char line[] = "Дом! "; /* Инициализация символьного мас-
                          сива внешнего класса памяти.   */
void main ()
{     cout << "Получили слово: ";
      for (int i=0; i<n; i++) cout << line[i];
}
```

Пример 3. Рассмотрим функцию **month_name()**, которая возвращает указатель на строку, содержащий имя n-го месяца.

Функция **month_name()** содержит локальный массив строк и при обращении к ней возвращает указатель на нужную строку.

В описании массива указателей на символы **name[]** инициализатором является просто список строк. Символы i-й строки помещаются в определенное место памяти, а указатель на ее начало хранится в элементе **name[i]**. Поскольку размер массива **name** не указан, компилятор сам подсчитывает количество инициализаторов и соответственно устанавливает правильное число.

```
#include <iostream.h>
#define n 15
void main ()
{    char *month_name(int);
     for (int i=0; i < n; i++)
       cout << "месяц номер " << i << " - " <<  month_name(i) << endl;
 }
char *month_name (int k)  /* Название k-го месяца */
{     static char *name[] = {
                        "неверный месяц","январь",
                        "февраль","март","апрель",
                        "май","июнь","июль","август",
                        "сентябрь", "октябрь", "ноябрь",
                        "декабрь"
                       };
     return (k<1||k>12)?name[0]:name[k];
}
```

Перечислим некоторые основные функции (таблица 8), предназначенные для работы со строками. Большинство прототипов этих функций (если не оговорено особо) находится в заголовочном файле **string.h**. Полный перечень функций для работы со строками можно найти в шаге 63 [2].

Таблица 8 - Основные функции работы со строками

Функция	Назначение
int getchar();	Возвращает значение символа(если он есть), который пользователь набрал на клавиатуре. После ввода символа нужно нажать клавишу **Enter**. Заголовочный файл - **stdio.h**
int getch();	Аналогично предыдущему, только символ на экране не отображается. Используется чаще для организации задержки выполнения программы. Заголовочный файл - **conio.h**
int putchar (int c);	Выводит символ **c** на экран. В случае успеха возвращает сам символ **c**, в противном случае - **EOF**. Заголовочный файл - **stdio.h**
char *gets (char *s);	Читает символы, включая пробелы и табуляции, до тех пор, пока не встретится символ новой строки, который заменяется нулевым символом. Последовательность прочитанных символов запоминается в области памяти, адресуемой аргументом **s**. В случае успеха возвращает аргумент **s**, в случае ошибки - нуль. Заголовочный файл - **stdio.h**
int puts (const char *s);	Выводит строку, заданную аргументом **const char *s**. Заголовочный файл - **stdio.h**
char *strcat (char *dest, const char *scr);	Объединяет исходную строку **scr** и результирующую строку **dest**, присоединяя первую к последней. Возвращает **dest**
char *strchr (const char *s, int c);	Ищет в строке **s** первое вхождение символа **c**, начиная с начала строки. В случае успеха возвращает указатель на найденный символ, иначе возвращает нуль
int strcmp (const char *s1, const char *s2);	Сравнивает две строки. Возвращает отрицательное значение, если **s1<s2**; нуль, если **s1==s2**; положительное значение, если **s1>s2**. Параметры - указатели на сравниваемые строки
char *strcpy (char *dest, const char *src);	Копирует исходную строку **src** и завершающий ее нулевой символ в строку результата **dest**. Возвращает **dest**
int strlen (const char *s);	Возвращает длину строки **s**, т.е. количество символов, предшествующих нулевому символу
char *strrev (char *s);	Изменяет порядок следования символов в строке на обратный (кроме завершающего нулевого символа). Функция возвращает строку **s**
char *strstr (const char *s1, const char *s2);	Ищет в строке **s1** строку **s2**. Возвращает адрес первого символа вхождения строки **s2**. Если строка отсутствует, возвращает нуль
char *strtok (char *s1, const char *s2);	Делит исходную строку **s1** на лексемы (подстроки), разделенные одним или несколькими символами из строки **s2**.

Пример 4. Определение длины строки

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

```
#include <iostream.h>
#define MAXLEN 256
main()
{
        char string[MAXLEN]; /* Место для 255 символов. */
        cout << "Задайте строку (можно задавать пробелы): ";
        gets(string);
        cout << endl; /* Начать новую строку. */
        cout << "Строка: " << string << endl;
        cout << "Ее длина =" << strlen(string);
}
```

Здесь определяется строковая переменная с именем **string** для приема ввода от функции **gets()**. После того как вы введете строку, программа передаст переменную **string** функции **strlen()**, которая вычислит длину строки в символах.

Пример 5. Копирование строк.

Оператор присваивания для строк не определен. Если **c1** и **c2** - символьные массивы, вы не сможете скопировать один в другой следующим образом: **c1 = c2;**

Но если **c1** и **c2** объявить как указатели типа **char ***, компилятор согласится с этим оператором, но вряд ли вы получите ожидаемый результат. Вместо копирования символов из одной строки в другую оператор **c1 = c2** ***скопирует указатель c2 в указатель c1***, перезаписав таким образом адрес в **c1**, потенциально потеряв информацию, адресуемую указателем.

Чтобы скопировать одну строку в другую, вместо использования оператора присваивания вызовите функцию копирования строк **strcpy()**. Для двух указателей **c1** и **c2** типа **char *** оператор **strcpy(c1, c2);** копирует символы, адресуемые указателем c2, в память, адресуемую указателем c1, включая завершающие нули. И только на вас лежит ответственность за то, что принимающая строка будет иметь достаточно места для хранения копии.

Аналогичная функция **strncpy()** ограничивает количество копируемых символов. Если источник (**source**) и приемник (**destination**) являются указателями типа **char *** или символьными массивами, то оператор **strcpy(destination, source, 10);** скопирует до 10 символов из строки, адресуемой указателем **source**, в область памяти, адресуемую указателем **destination**. Если строка **source** имеет больше 10 символов, то результат усекается. Если же меньше, то неиспользуемые байты результата устанавливаются равными нулю.

Замечание. Строковые функции, в имени которых содержится дополнительная буква ***n****, объявляют числовой параметр, ограничивающий некоторым образом действие функции. Эти функции безопаснее, но медленнее, чем их аналоги, не содержащие букву* ***n****. Программные примеры содержат следующие пары функций:* ***strcpy()*** *и* ***strncpy()****,* ***strcat()*** *и* ***strncat()****,* ***strcmp()*** *и* ***strncmp()****.*

Пример 6. Конкатенация строк. Пример показывает, как можно использовать функцию **strcat()** для получения в одной строке фамилии, имени и отчества, хранящихся отдельно, например, в виде полей базы данных. Введите фамилию, имя и отчество. Программа сцепит введенные вами строки и отобразит их как отдельную строку.

```
#include <iostream.h>
#include <string.h>
void main()
```

```
{ //Резервирование места для ввода трех строк.
  char *fam = new char[128];
  char *im = new char[128];
  char *otch = new char[128];
  //Ввод данных.
  cout << "Задайте" << endl;
  cout << "\tфамилию: ";
  cin >> fam;
  cout << "\tимя: ";
  cin >> im;
  cout << "\totчество: ";
  cin >> otch;
  //Резервирование места для результата.
  //Нужно учесть два пробела и результирующий
  //нулевой символ.
  char *rez=new char[strlen(fam)+strlen(im)+strlen(otch)+3];
  // «Сборка» результата.
  strcat(strcat(strcpy(rez,fam), " "),im);
  strcat(strcat(rez, " "),otch);
  //Возврат памяти в кучу.
  delete [] fam;
  delete [] im;
  delete [] otch;
  //Вывод результата.
  cout << "\nРезультат: " << rez;
  delete [] rez;
}
```

Приведенная программа демонстрируют важный принцип конкатенации строк: ***всегда инициализируйте первый строковый аргумент***. В данном случае символьный массив **rez** инициализируется вызовом функции **strcpy()**, которая вставляет **fam** в **rez**. После этого программа добавляет пробелы и две другие строки - **im** и **otch**. Никогда не вызывайте функцию **strcat()** с неинициализированным первым аргументом.

Если вы не уверены в том, что в строке достаточно места для присоединяемых подстрок, вызовите функцию **strncat()**, которая аналогична функции **strcat()**, но требует числового аргумента, определяющего число копируемых символов. Для строк **s1** и **s2**, которые могут быть либо указателями типа **char ***, либо символьными массивами, оператор **strncat(s1, s2, 4);** присоединяет максимум четыре символа из **s2** в конец строки **s1**. Результат обязательно завершается нулевым символом.

Существует один способ использования функции **strncat()**, гарантирующий безопасную конкатенацию. Он состоит в передаче функции **strncat()** размера свободной памяти строки-приемника в качестве третьего аргумента. Рассмотрим следующие объявления:

```
#define MAXLEN 128
char s1[MAXLEN] = "Кот";
char s2[] = "в шляпе";
```

Вы можете присоединить **s2** к **s1**, формируя строку ***«Кот в шляпе»***, с помо-

щью функции **strcat(): strcat(s1, s2);**

Если вы не уверены, что в **s1** достаточно места, чтобы запомнить результат, используйте альтернативный оператор: **strncat(s1, s2, (MAXLEN-1)-strlen(s1));** Этот способ гарантирует, что s1 не переполнится, даже если s2 нужно будет урезать до подходящего размера. Этот оператор прекрасно работает, если s1 - нулевая строка.

Пример 7. Поиск символов. Пример показывает, как использовать функцию **strchr()** для поиска отдельных символов в строке.

```
#include <iostream.h>
#include <string.h>
#include <stdio.h>
void main()
{
   char *filename = new char[128];
   cout << "Задайте имя файла: ";
   gets(filename);
   cout << "\nЗаданное имя файла: " << filename << endl;
   if (strchr (filename,'.'))
     cout << "Имя уже содержит расширение" << endl;
   else
     strcat (filename, ".TXT");
   cout << "Окончательное имя файла: " << filename << endl;
   delete [] filename;
}
```

Данная программа находит расширение в имени файла, выполняя поиск точки среди символов введенной строки. (В имени файла может быть только одна точка, которая должна предшествовать расширению, если оно имеется.) Ключевым в этой программе является оператор:

```
if (strchr (filename,'.'))
   cout << "Имя уже содержит расширение" << endl;
else
   strcat (filename, ".TXT");
```

Выражение **strchr (filename,'.')** возвращает указатель на символ точки в строке, адресуемой указателем **filename**. Если такой символ не найден, функция **strchr()** возвращает нуль. Поскольку ненулевые значения означают «истину», вы можете использовать функцию **strchr()** в качестве возвращающей значение «истина» / «ложь». Вы также можете применить функцию **strchr()** для присваивания указателя на подстроку, начинающуюся с заданного символа.

Функция **strchr()** отыскивает первое появление символа в строке. Конструкции:

```
char s[]="Abracadabra";
char *p = strchr(s,'a');
```

присваивают указателю **p** адрес первой строчной буквы 'a' в строке «Abracadabra».

Функция **strchr()** рассматривает завершающий нуль строки как значащий символ. Учитывая это, можно узнать адрес конца строки. Оператор **p = strchr(s,0);** (с учетом предыдущих описаний) установит **p** равным адресу нулевого символа в строке **s**. Используйте этот простой метод, чтобы найти конец строки.

Чтобы найти последнее появление символа в строке, вызовите функцию

strrchr(). Учитывая предыдущие объявления, оператор **p = strrchr(s,'b');** установит указатель **p** равным адресу подстроки «bra» в конце строки «Abracadabra».

Пример 8. Поиск подстрок. Данная программа аналогична примеру 6, но устанавливает расширение файла **.TXT**.

```
#include <iostream.h>
#include <string.h>
#include <stdio.h>
void main()
{
 char *filename = new char[128],*p;
 cout << "Задайте имя файла: ";
 gets(filename);
 cout << "\nЗаданное имя файла: " << filename << endl;
 strupr(filename);
 p = strstr (filename, ".TXT");
 if (p)
   cout << "Имя уже содержит требуемое расширение" << endl;
 else
  { p = strchr (filename,'.');
   if (p)
      *p=NULL; //Удалить любое другое расширение.
   strcat (filename, ".TXT");
  }
 cout << "Окончательное имя файла: " << filename << endl;
 delete [] filename;
}
```

Эта программа создает имя файла, которое обязательно заканчивается расширением **.TXT**. Чтобы определить, есть ли в имени файла это расширение, программа выполняет оператор **p = strstr (filename,** ".**TXT**"**);**

Подобно **strchr()**, функция **strstr()** возвращает адрес подстроки или нуль, если искомая строка не найдена. Если же цель будет обнаружена, указатель **p** установится равным ее адресу, в данном примере - адресу точки в подстроке **.TXT**. Поскольку расширение может быть введено и строчными буквами, программа выполняет оператор **strupr(filename);**, чтобы перед вызовом функции **strstr()** преобразовать буквы оригинальной строки в прописные.

Программа примера 7 также демонстрирует способ усечения строки в позиции заданного символа или подстроки. Здесь вызывается функция **strstr()**, чтобы установить указатель **p** равным адресу первой точки в строке **filename**. Если результат этого поиска не нулевой, то выполнится оператор, который запишет вместо точки нулевой байт: ***p = NULL;**

Тем самым будет присоединен новый конец строки в том месте, где раньше находилось расширение файла. Теперь строка готова к добавлению нового расширения путем вызова функции **strcat()**.

Пример 9. Разложение строк на подстроки.

Прикладное программирование часто требует нахождения компонентов строки, или ***лексем***. Этот процесс называется ***синтаксическим анализом***. Если составные части строки отделены друг от друга запятыми, пробелами или другими разделителями, можно использовать функцию **strtok()** для разложения строки на

несколько подстрок.

Рассмотрим следующие объявления:

char s[] = “Мама мыла раму с мылом“;
char *p1, *p2, *p3, *p4, *p5;

Строка **s** является инициализированным символьным массивом. Пять указателей на тип **char** пока что не инициализированы. Будем их использовать при разборе строки, адресуемой указателем **s**. Сначала вызовем функцию **strtok()**, передав в качестве первого аргумента указатель на строку, а в качестве второго - разделительный символ (представленный в данном случае строкой, состоящей из одного символа пробела): **p1 = strtok(s,** “ “**);**

Функция **strtok()** возвращает адрес первого компонента, отделенного от следующего заданным разделителем, в данном примере - пробелом. Рисунок 5 иллюстрирует состояние переменных программы как раз на этой стадии.

Рисунок 5 - После первого вызова **strtok()**

Как показано на рисунке 5, функция **strtok()** вставляет нулевой байт (представленный на рисунке как '\0') в позицию первого заданного ограничителя. Тем самым создается маленькая подстрока, адрес которой возвращается функцией **strtok()**, а в данном примере присваивается переменной **p1**. Если заданные разделители не обнаружены, функция **strtok()** возвращает нуль. Кроме того, эта функция настраивает свой внутренний указатель на символ, следующий за концом последней сформированной подстроки, чтобы последующий вызов **strtok()** мог продолжить разложение строки. Для этого передайте в качестве первого аргумента значение **NULL**, это послужит сигналом для функции **strtok()** использовать ее внутренний указатель как стартовый адрес для поиска другого разделителя. Таким образом, чтобы выделить другие компоненты строки, нужно просто вызвать функцию **strtok()** несколько раз, и каждый раз первым ее аргументом должен быть **NULL**:

p2 = strtok(NULL, “ “);
p3 = strtok(NULL, “ “);
p4 = strtok(NULL, “ “);
p5 = strtok(NULL, “ “);

На рисунке 6 показана разобранная строка и ее указатели от **p1** до **p5**. Каждый указатель адресует завершающуюся нулем подстроку внутри первоначальной строки. Каждая из подстрок теперь является отдельной строковой переменной, и эти пять указателей можно передавать другим строковым функциям, таким как **strlen()**,**strcpy()** и **strdup()**.

Рисунок 6 - После разбора строки с помощью функции **strtok()**

В предыдущем примере предполагалось, что заранее известно, из скольких компонентов состоит строка. В большинстве же случаев заранее это неизвестно, поэтому используется несколько другой механизм.

Чтобы выделить составляющие строки, более разумно было бы использовать цикл, в котором результаты работы функции **strtok()** передавались бы другим строковым функциям или сохранялись. Такой цикл обычно выглядит следующим образом:

```
p = strtok(buffer,";");
while (p)
{
  //Обработка подстроки, адресуемой указателем p.
   p = strtok(NULL,";");
}
```

Сначала указатель **p** устанавливается равным результату функции **strtok()**, которой были переданы указатель на исходную строку **buffer** и разделитель «;». Затем цикл **while** проверяет значение указателя **p** на равенство нулю. Если указатель **p** имеет ненулевое значение, то он адресует очередную подстроку в строке **buffer** и вы можете передавать его другой строковой функции. После обработки найденной подстроки внутри цикла осуществляется попытка поиска других подстрок с помощью нового вызова функции **strtok()**, но теперь в качестве первого аргумента передается значение **NULL**.

Пример 9 показывает, как использовать функцию **strtok()** для выделения слов из строки. Скомпилируйте и запустите программу. Затем (когда программа предложит сделать ввод) введите строку, состоящую из слов, разделенных пробелами. После этого будет проведен анализ введенной строки и отображение его результатов в виде отдельных слов.

```
#include <iostream.h>
#include <string.h>
#include <stdio.h>
void main()
{
 char buffer[128],*p;
 cout << "Задайте строку, слова в которой разделены пробелами:\n";
 gets(buffer);
 cout << "\nЗаданная строка: " << buffer << endl;
 cout << "Ее разбиение на слова: " << endl;
 p = strtok (buffer, " ");
```

```
  while (p)
   {
     cout << p << endl;
     p = strtok (NULL, " ");
   }
}
```

Задание 16. Используя сведения о функциях, решите следующие задачи (назовите файлы **z16_номер задачи.cpp**).

Вариант 1

1 Вывести строку длины N (N — четное), которая состоит из чередующихся символов C_1 и C_2, начиная с C_1.

2 Дана строка. Вывести строку, содержащую те же символы, но расположенные в обратном порядке.

3 Дана строка, состоящая из слов, разделенных пробелами (одним или несколькими). Определить количество слов, которые начинаются и заканчиваются одной и той же буквой.

4 Дана строка, состоящая из слов, разделенных пробелами (одним или несколькими). Определить количество слов, которые содержат ровно три буквы «А».

Вариант 2

1 Дана строка. Вывести символы, расположенные на четных местах.

2 Дана строка. Подсчитать количество содержащихся в ней цифр.

3 Дана строка, состоящая из слов, разделенных пробелами (одним или несколькими). Определить длину самого короткого слова.

4 Дана строка, состоящая из слов, разделенных пробелами (одним или несколькими). Вывести строку, содержащую эти же слова, но разделенные одним символом «.» (точка). В конце точку не ставить.

Вариант 3

1 Дана строка. Преобразовать все строчные буквы в прописные, а прописные в строчные.

2 Дана строка. Если она представляет собой запись целого числа, то вывести 1; если вещественного (с дробной частью), то вывести 2; если строку нельзя преобразовать в число, то вывести 0.

3 Дана строка, состоящая из слов, разделенных пробелами (одним или несколькими). Вывести строку, содержащую эти же слова (разделенные одним пробелом), но расположенные в обратном порядке.

4 Дана строка, состоящая из слов, разделенных пробелами (одним или несколькими). Преобразовать каждое слово в строке, удалив из него все последующие вхождения первой буквы этого слова (количество пробелов между словами не изменять).

Вариант 4

1 Дано целое число. Вывести набор символов, содержащий цифры этого числа в обратном порядке.

2 Дана строка, изображающая двоичную запись целого числа. Вывести строку, изображающую восьмеричную запись этого же числа.

3 Дана строка, состоящая из слов, разделенных пробелами (одним или несколькими). Вывести строку, содержащую эти же слова (разделенные одним пробелом), но

расположенные в алфавитном порядке.
4 Дана строка. Преобразовать строку так, чтобы каждое слово начиналось с заглавной буквы.

Вариант 5

1 Дана строка, изображающая двоичную запись целого числа. Вывести строку, изображающую десятичную запись этого же числа.
2 Дана строка, изображающая целое число. Вывести сумму цифр этого числа.
3 Дана строка. Подсчитать количество содержащихся в строке знаков препинания.
4 Дана строка-предложение. Зашифровать ее, поместив вначале все символы, расположенные на четных местах, а затем в обратном порядке все символы, расположенные на нечетных местах (например, строка «Программа» превратится в «ргамамроП»).

Вариант 6

1 Дана строка S и число N. Преобразовать строку S в строку длины N следующим образом: если длина строки S больше N, то отбросить первые символы, если длина строки S меньше N, то в ее начало добавить символы «.» (точка).
2 Даны два числа: N1 и N2, и две строки: S1 и S2. Получить из этих строк новую строку, объединив N1 первых символов строки S1 и N2 последних символов строки S2.
3 Дана строка-предложение, содержащая избыточные пробелы. Преобразовать ее так, чтобы между словами был ровно один пробел.
4 Дана строка-предложение на русском языке и число k ($0 < k < 10$). Зашифровать строку, выполнив циклическую замену каждой буквы на букву того же регистра, расположенную в алфавите на k-й позиции после шифруемой буквы (например, для $k = 2$ «А» перейдет в «В», «а» — в «в», «Б» — в «Г», «я» — в «б» и т.д.). Букву «ё» в алфавите не учитывать, знаки препинания и пробелы не изменять.

Вариант 7

1 Даны две строки: S1 и S2. Проверить, содержится ли строка S2 в строке S1. Если да, то вывести номер позиции, начиная с которой S2 содержится в S1, если нет, то вывести 0.
2 Даны две строки: S1 и S2. Определить количество вхождений строки S2 в строку S1.
3 Дана строка, содержащая полное имя файла. Выделить из строки название последнего каталога (без символов «\»). Если файл содержится в корневом каталоге, то вывести символ «\».
4 Дана строка-предложение на русском языке. Зашифровать ее, выполнив циклическую замену каждой буквы на следующую за ней в алфавите и сохраняя при этом регистр букв («А» перейдет в «Б», «а» — в «б», «Б» — в «В», «я» — в «а» и т.д.). Букву «ё» в алфавите не учитывать («е» должна переходить в «ж»). Знаки препинания и пробелы не изменять.

Вариант 8

1 Дана строка S и символ C. Удвоить каждое вхождение символа C в строку S.
2 Даны строки S1, S2 и символ C. Перед каждым вхождением символа C в строку S1 вставить строку S2.
3 Дано зашифрованное предложение на русском языке (способ шифрования описан в задаче 4 варианта 6) и кодовое смещение k ($0 < k < 10$). Расшифровать предложение.
4 Дана строка, содержащая полное имя файла, то есть имя диска, список каталогов

(путь), собственно имя и расширение. Выделить из этой строки имя и расширение файла.

Вариант 9

1 Даны две строки: S1 и S2. Удалить из строки S1 первую подстроку, совпадающую с S2. Если таких подстрок нет, то вывести S1 без изменений.
2 Даны три строки: S1, S2, S3. Заменить в строке S1 все вхождения строки S2 на S3.
3 Дано зашифрованное предложение на русском языке (способ шифрования описан в задаче 4 варианта 6) и его расшифрованный первый символ С. Определить кодовое смещение k и расшифровать предложение.
4 Дана строка. Вывести самое длинное слово в предложении (если таких слов несколько, то вывести их все).

Вариант 10

1 Дана строка. Вывести подстроку, расположенную между первой и второй точками исходной строки. Если в строке менее двух точек, то вывести всю исходную строку.
2 Дана строка, состоящая из слов, разделенных пробелами (одним или несколькими). Определить количество слов в строке.
3 Дано предложение, зашифрованное по правилу, описанному в задаче 4 варианта 5. Расшифровать это предложение.
4 Дана строка, содержащая несколько круглых скобок. Если скобки расставлены правильно (то есть каждой открывающей соответствует одна закрывающая), то вывести число 0. В противном случае вывести или номер позиции, в которой расположена первая ошибочная закрывающая скобка, или, если закрывающих скобок не хватает, число –1.

Лабораторная работа № 9. Файлы. Функции высокоуровневого ввода вывода *(8 часов)*

Прежде чем читать или записывать информацию в файл, его нужно открыть с помощью стандартной библиотечной функции **fopen**(), которая работает с физическим именем файла и возвращает значение логического имени (указатель на файл), которое и используется системой для чтения и записи в данный файл. Общий вид этой функции следующий:

<указатель на файл> = fopen (<имя файла>,<режим доступа>); .

В этой функции указатель на файл должен быть описан так:

FILE *fp;

где тип **FILE** фактически является типом **int** (см. файл заголовков stdio.h). Возможные режимы доступа перечислены в таблице 9.

Таблица 9 -Режимы доступа к файлам для функции fopen()

Строка режима	Описание
“**r**“	Открывает файл только для чтения. Модификация файла не разрешена.
“**w**“	Создает новый файл только для записи. Перезаписывает любой существующий файл с тем же именем. Чтение информации из файла не разрешено.
“**av**	Открывает файл в режиме «только для записи» с добавлением новой информации в конец файла. Если файл не существует, он создается, а любой существующий файл с таким же именем перезаписывается. Чтение информации из файла не разрешено.
“**r+**“	Открывает существующий файл для чтения и записи.
“**w+**“	Создает новый файл для чтения и записи. Перезаписывает любой существующий файл с тем же именем.
“**a+**“	Открывает файл в режиме чтения и записи для добавления новой информации в конец файла. Если файл не существует, он создается, и любой существующий файл с тем же именем перезаписывается.

При невозможности открыть файл с заданным именем функция **fopen**() возвращает нулевое (**NULL**) значение указателя файла, что сразу сигнализирует об ошибке.

Функция **fclose (<указатель на файл>);** «разрывает» связь между указателем на файл и физическим именем, установленную функцией **fopen**(), и освобождает указатель для другого файла.

Для чтения информации из уже открытого файла или записи в него существует несколько возможностей. Рассмотрим пока самые простые - функции **getc**() и **putc**(). Функция **getc (<указатель на файл>);** возвращает очередной символ из файла с заданным указателем. Если уже достигнут конец файла, то функция возвращает значение **EOF**.

Функция **putc (<переменная>, <указатель на файл>);** помещает значение переменной в файл с заданным указателем.

С началом работы любой программы автоматически открываются три файла: стандартный вход, стандартный выход и стандартный выход для ошибок.

Соответствующие ссылки на файлы называются **stdin**, **stdout** и **stderr**. По умолчанию все они связаны с терминалом, однако **stdin** и **stdout** можно связать с файлами и межпрограммными каналами. Ссылки на **stdin**, **stdoutи stderr** имеются в файле стандартных заголовков **stdio**.**h**.

Приведем несколько примеров, иллюстрирующих использование рассмотренных функций.

Пример 1. Вывод содержимого файла.

```
#include <iostream.h>
#include <stdio.h>
void main ()
{
  FILE *fp; /* Указатель на файл. */
  char ch;
  if ((fp = fopen (“f1.cpp“,“r“)) != NULL)
```

```
  {  /* Открываем файл «f1.cpp» для чтения, одновременно */
     /* проверяя, существует ли он. Указатель fp ссыла- */
     /* ется на файл «f1.cpp». */
     while ((ch = getc (fp)) != EOF)
       /* Получаем символ из файла и проверяем, не */
       /* является ли этот символ концом файла. */
            putc (ch,stdout); /* Выводим этот символ. */
     fclose (fp); /* Закрываем файл. */
}
  else cout << "Файл \"f1.cpp\" отсутствует.\n";
}
```

Пример 2. Копирование содержимого одного файла в другой.

```
#include <iostream.h>
#include <stdio.h>
void main ()
{
   FILE *in,*out;
   char ch;
   if ( (in=fopen ("f1.cpp","r")) != NULL )
     { out = fopen ("f2.cpp","w");
       while ((ch=getc(in)) != EOF)
            putc (ch,out);
       fclose (in);
       fclose (out);
      }
   else cout << "Исходный файл не найден!\n";
}
```

Перечислим другие функции, позволяющие организовать работу с файлами (таблица 10) (для всех функций заголовочный файл **stdio.h**).

Таблица 10 - Функции работы с файлами на высоком уровне

Функция	Описание
int feof (<указатель на файл>);	Проверка достижения конца файла. Возвращает истину (ненулевое значение), если указатель находится в конце файла, и ложь - в противном случае
int fflush (<указатель на файл>);	Запись на диск содержимого буфера для заданного файла
char *fgets(<указатель на буфер>,<максимальное число читаемых символов>,<указатель на файл>);	Последовательное чтение символов из файла до начала новой строки или заданное количество символов, уменьшенное на 1. Прочитанные символы помещаются в буфер, начиная с ячейки, адресуемой первым параметром
int fprintf (<указатель на файл>,<формат>,<аргументы>);	Записывает данные в файл по заданному формату. Возвращает количество записанных байтов. Если обнаружены ошибки, то возвращает EOF

Продолжение таблицы 10

Функция	Описание
int fputs(<строка>,<указатель на файл>);	Запись заданной строки в файл
int fread (<указатель на буфер>,<размер в байтах одного элемента>,<количество читаемых элементов>,<указатель на файл>);	Чтение данных из файла, начиная с текущего положения указателя. Возвращает число прочитанных элементов. Число прочитанных байтов равно результату функции, умноженному на число байтов в элементе. В случае ошибки возвращается нуль
int fscanf (<указатель на файл>,<формат>,<список адресных аргументов>);	Чтение данных из файла, преобразование их в соответствующий формат и размещение в ячейки, заданные адресными аргументами
int fseek (<указатель на файл>,<количество байтов>,<начальная позиция отсчета>);	Перемещение указателя файла на заданное количество байтов. В случае успеха возвращается нуль, в случае ошибки - ненулевое значение. Третий параметр может принимать следующие значения: SEEK_SET - отсчет идет от начала файла, SEEK_END - отсчет идет от конца файла, SEEK_CUR - отсчет идет от начала текущей позиции
void rewind (<указатель на файл>);	Установка указателя на начало файла

Пример 3. Напишите программу очистки содержимого данного файла.

```
#include <iostream.h>
#include <stdio.h>
void main ()
{
    FILE *in;
    if ( (in=fopen (“f1.cpp”,”w”)) != NULL )
       fclose (in);
   else cout << “Исходный файл не найден!\n”;
}
```

Пример 4. Заданный файл копировать в новый файл, который имеет расширение red и содержит только те символы исходного файла, номера позиций которых в исходном файле кратны трем (0,3,6,9,...).

```
#include <iostream.h>
#include <stdio.h>
#include <string.h>
void main ()
{
    FILE *in,*out;
    char ch;
    char name[20];
    int count=0;
    if  ((in=fopen(“f1.cpp”,”r”)) != NULL)
      {
```

```
        /* Копируем имя файла в строку. */
        strcpy (name, “f1.cpp”);
        /* Проверяем, есть ли расширение. */
        char *p = strchr (name,'.');
        if (p)
          *p=NULL; //Удалить расширение.
        /* Добавляем .red к имени файла. */
        out = fopen (strcat (name,”.red”),”w”);
        while  ( (ch=getc (in)) != EOF)
          /* Печатаем каждый третий символ файла. */
          if  (count++%3==0)
            putc (ch,out);
        fclose (in); fclose (out);
      }
      else  cout << “Файл не существует!”;
    }
}
```

Пример 5. Вывод на экран дисплея содержимого нескольких текстовых файлов последовательно один за другим (имена файлов заданы в командной строке).

Если исполняемый файл имеет имя pr.exe, а текстовые файлы f1.txt, f2.txt, то командная строка должна иметь следующий вид: C:\>pr.exe f1.txt f2.txt (в этом случае argc=3)

```
#include <iostream.h>
#include <stdio.h>
void main (int argc,char *argv[])
{
    FILE *in;
    char   t;
    if  (argc < 2)
      cout << “Не заданы параметры в командной строке \n”;
    else
      for  (int i=1; i < argc; i++)
        if  ((in=fopen(argv[i],”r”)) != NULL)
        {
          cout << “Содержимое файла \” ” << argv[i] << “\”:” << endl;
          while  ((t=getc(in)) != EOF)
             cout << t;
          fclose (in);
          cout << endl;
        }
        else cout << “Файла с именем \”“ << argv[i] << “\” нет»;
}
```

Пример 6. Дан символьный файл. Записать компоненты файла в обратном порядке (в командной строке сначала указывается имя файла-источника, а затем имя файла- приемника).

```
#include <iostream.h>
#include <stdio.h>
void main (int argc,char *argv[])
```

```
{
   FILE *in,*out;
   char a;
   if  (argc < 3)
    cout << “Не заданы параметры в командной строке \n”;
   else
   {
    in = fopen (argv[1],”r”); out = fopen (argv[2],”w”);
    int i = -1;
    while  (!fseek(in,i,SEEK_END))
    {
     a=getc(in);
     if ((int)a==0x0a) {putc('\n',out); i--; }
     else
       putc (a,out);
     i--;
    }
    fclose (in); fclose (out);
   }
}
```

Прокомментируем текст приведенной программы. Сначала немного поясним общий ход решения задачи. Будем читать посимвольно содержимое первого файла (с конца файла!) и записывать в начало второго файла. Для того чтобы осуществить чтение с конца файла, нужно разместить там файловый указатель. Это выполняется функцией **fseek(in,i,SEEK_END)**.

Будем выполнять эту функцию, пока возможно (то есть пока она не вернет нулевое значение). Параметр **SEEK_END** говорит о том, что отсчет будет идти от конца файла. В этом случае количество байтов, которое нужно отсчитать должно быть отрицательным, поэтому переменная **i** последовательно принимает значения: -1, -2 и т.д.

Поясним условие **if ((int)a==0x0a) {putc('\n',out); i--; }**

Переход на следующую строку в файле кодируется двумя байтами: 0D 0A. Так как мы читаем содержимое первого файла с конца, то первый прочитанный байт будет 0A. Если мы его встретили, то во второй файл помещаем символ перехода на новую строку ('\n') и пропускаем следующий байт (0D) командой i--, переходя к очередному байту.

Для форматного ввода или вывода в файл можно использовать функции:

fprintf (<указатель на файл>,<указатель на формат>,<аргументы>);

fscanf (<указатель на файл>,<указатель на формат>,<аргументы>);

Они идентичны функциям printf() и scanf(), но только первый параметр - указатель на файл.

***Пример** 7*. Дан файл f, компоненты которого являются целыми числами: a) найти количество четных чисел среди компонент; b) найти количество удвоенных нечетных чисел среди компонент; c) найти количество квадратов нечетных чисел среди компонент (имя файла задается в командной строке).

```
#include <iostream.h>
#include <stdio.h>
void main (int argc,char *argv[])
```

```
{   FILE      *in;
    int  t,i,k,p,m;
    k = p = m = 0;
    if  (argc < 2)
      cout << “Имя файла не задано в командной строке.”;
    else
    {
      in = fopen (argv[1],”w+”);
      cout << “Вводите компоненты файла. Окончание ввода - число  -1\n”;
      do
      {
        cin >> t; fprintf (in,”%d\n”,t);
      }
      while (t!=EOF);
      fseek (in,0,SEEK_SET);//Перемещение указателя в начало файла.
      /* Вывод. */
      cout << “Контрольный вывод файла:\n”;
      fscanf (in,”%d”,&t);
      while (t!=EOF)
      {
        cout << t << endl; fscanf (in,”%d”,&t);
      }
      fseek (in,0,SEEK_SET);//Перемещение указателя в начало файла.
      //Решение задачи.
      fscanf (in,”%d\n”,&t);
      while (t!=EOF)
      {
        if  (t%2==0)
        {
          k++;
          if  (t%4!=0)
            p++;
        }
        else
          for (i=1; i<=t/2+1; i++)
            if  (t==i*i)
            {
              m++; break;
            }
        fscanf (in,”%d\n”,&t);
      }
      fclose (in);
      cout << “*****************************************\n”;
      cout << “Количество четных чисел          : “ << k << endl;
      cout << “Количество удвоенных нечетных чисел: “ << p << endl;
      cout << “Количество квадратов нечетных чисел: “ << m << endl;
    }
}
```

Пример 8. Дан файл, содержащий целые числа. Найти наибольшее из них.

```
#include <iostream.h>
#include <stdio.h>
void main (int argc,char *argv[])
{
    FILE *f;
    int k,l;
    /* ------ */
    if  (argc < 2)
      cout << “Имя файла не задано в командной строке.”;
    else
    {
      cout << “Файл: \n”; f = fopen (argv[1],”w+”);
      do
      {
        cin >> k; fprintf (f,”%d\n”,k);
      }
      while  (k!=EOF);
      fseek (in,0,SEEK_SET);//Перемещение указателя в начало файла.
      //Решение задачи.
      fscanf (f,”%d\n”,&l);
      do
      {
        fscanf (f,”%d\n”,&k);
        if  (k>l)
          l = k;
      }
      while  (k!=EOF);
      fclose (f);
      cout << “Максимальное значение = “ << l << endl;
    }
}
```

Работа с файлами. Функции низкоуровневого ввода/вывода

Компьютер «рассматривает» содержимое любого файла как последовательность байтов, интерпретацию которых осуществляют функции высокоуровневого ввода/вывода. Однако в некоторых случаях эту интерпретацию должен осуществлять сам программист. Для реализации этой задачи в C++ имеются функции низкоуровневого ввода/вывода. Перечислим функции, позволяющие работать с файлами на низком уровне. Если не оговорено особо, то заголовочный файл **io.h**.

1 int open (<имя файла>, <доступ>, <дополнительный доступ>);. Открывает файл в заданном режиме. Функция возвращает дескриптор файла, который затем используется в других функциях работы с файлами. Второй параметр может принимать следующие значения (они записаны в заголовочном файле fcntl.h): O_RDONLY (только чтение); O_WRONLY (только запись); O_RDWR (чтение и запись), объединенная с помощью логической операции ИЛИ с требуемой комбинацией констант O_APPEND (добавить данные к концу файла), O_CREAT (создать новый файл), O_TRUNC (переписать существующий файл), O_EXCL (вместе с O_CREAT запрещает повторное создание существующего файла),

O_BINARY(открыть в двоичном режиме), O_TEXT (открыть как текст). Третий параметр устанавливается, если второй параметр имеет значение O_CREAT. Значение третьего параметра может быть следующим: S_IWRITE(разрешена запись в файл), S_IREAD (разрешено чтение из файла) или S_IWRITE | S_IREAD (разрешены чтение и запись). Заголовочные файлы: fcntl.h, sys\stat.h, io.h.

2 int close (<дескриптор файла>);. Закрывает открытый файл. В случае успеха возвращает 0, при ошибке возвращает -1.

3 int creat (<имя файла>,<доступ>);. Создает новый файл. В случае успеха возвращает дескриптор файла. В случае ошибки возвращает -1. Второй параметр может иметь одно из следующих значений: S_IWRITE (разрешена запись в файл), S_IREAD(разрешено чтение из файла) или S_IWRITE | S_IREAD (разрешены чтение и запись). Заголовочные файлы:sys\stat.h, io.h.

4 int read (<дескриптор файла>,<указатель на буфер>,<число читаемых байтов>);. Читает указанное число байтов из файла. Возвращает число байтов, являющихся содержимым файла и реально записанных в буфер. При достижении конца файла возвращается значение 0. Буфер должен иметь достаточный размер.

5 int write (<дескриптор файла>,<указатель на буфер>,<число записываемых байтов>);. Записывает указанное количество байтов в файл. Возвращает число байтов, реально записанных в файл. В случае, когда дисковое пространство частично недоступно, будет возвращено значение меньшее, чем указано в функции.

6 long lseek (<дескриптор файла>,<смещение в байтах>, <начальная позиция>);. Изменяет положение файлового указателя. В случае успеха возвращает новую позицию указателя как смещение в байтах относительно начала файла. В случае ошибки возвращает -1. Третий параметр задает точку отсчета: SEEK_SET (отсчет идет от начала файла), SEEK_CUR (отсчет идет от текущего положения указателя файла),SEEK_END (отсчет идет от конца файла).

7 long filelength (<дескриптор файла>);. Возвращает размер файла в байтах или -1 в случае ошибки.

8 int eof (<дескриптор файла>);. Определяет достижение конца файла. Возвращает истину (1), если указатель находится в конце файла и ложь (0) в противном случае.

Пример 9. Использование функций низкоуровневого ввода/вывода. Программа копирует файл s1 в файл s2 через буфер buf[NBYTES], где NBYTES - размер буфера.

```
#include <io.h>
#include <fcntl.h>
#include <iostream.h>
#include <sys\stat.h>
#define NBYTES 128
void main ()
{
   int fd1,fd2;
   int n_read,n_write;
   char buf[NBYTES];
   char s1[20],s2[20];
   cout << "Имя файла-источника: ";
   cin >> s1;
```

```
    cout << “Имя файла-приемника: “;
    cin >> s2;
    fd2 =creat (s2, S_IWRITE | S_IREAD);
    fd1 = open (s1, O_RDONLY);
    cout << “Содержимое исходного файла:” << endl;
    while (!eof(fd1))
    {
        /*Читаем заданное количество байтов в буфер.*/
      n_read = read (fd1,buf,NBYTES);
        /*Выводим их на экран.*/
      for(int i=0;i< n_read;i++)  cout << buf[i];
        /*Записываем прочитанные байты в файл-приемник.*/
      n_write = write (fd2,buf, n_read);
    }
    cout<< “\n Содержимое результирующего файла:” << endl;
    /*Перемещаем указатель файла-приемника на начало.*/
    lseek(fd2,0,SEEK_SET);
    while (!eof(fd2))  /*Выводим содержимое файла.*/
    {
      n_read = read (fd2,buf,NBYTES);
      for(int i=0;i< n_read;i++)  cout << buf[i];
    }
}
```

Список источников:

1 Медведев А. А. Основы языка программирования Си++ [Текст] : методические рекомендации для проведения лабораторных работ / А. А. Медведев. - Курган : Изд-во Курганского гос. ун-та, 1999. – 62 с.

2 Сайт кафедры «Информационные технологии и методика преподавания информатики». Раздел меню: Программирование | Язык программирования С++ | Начала. URL: http://it.kgsu.ru/C++/oglav.html (дата обращения 17.11.2015).

Адаменко Юлия Владимировна

Медведев Аркадий Андреевич

МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ ДЛЯ ИЗУЧЕНИЯ ОСНОВ ОБЪЕКТНО-ОРИЕНТИРОВАННОГО ПРОГРАММИРОВАНИЯ С ИСПОЛЬЗОВАНИЕМ ЯЗЫКА С++

Методические рекомендации для студентов
направлений 01.03.01, 010100.62 «Математика»,
050100.62 «Педагогическое образование»
(профиль «Математика»),
44.03.01 «Педагогическое образование»
(профиль «Информатика»)

Редактор О. Г. Арефьева

……

Подписано в печать	Формат 60×84 1/16	Бумага 65 г/м^2
Печать цифровая	Усл. печ. л. 3,0	Уч.-изд. л. 3,0
Заказ №	Тираж 25	Не для продажи

……...

РИЦ Курганского государственного университета.
640000, г. Курган, ул. Советская, 63/4.
Курганский государственный университет.