

Министерство общего и профессионального
образования Российской Федерации
Пермский государственный технический университет
Кафедра Автоматизированных систем управления

О. Л. Викентьева, О. А. Полякова

Программирование на языке С++
Лабораторный практикум для студентов
специальности АСУ

Пермь, 2003

Лабораторная работа №1. "Знакомство с Си++. Выполнение программы простой структуры"

Цель: Знакомство со средой программирования, создание, отладка и выполнение простой программы, содержащей ввод/вывод информации и простейшие вычисления.

1. Краткие теоретические сведения

Язык Си создан в 1972 г. Деннисом Ритчи при разработке ОС Unix. Он проектировался как инструмент системного программирования с ориентацией на разработку хорошо структурированных программ. Таким образом он сочетает в себе, с одной стороны, средства языка программирования высокого уровня: описание типов данных, операторы for, while, if и т. д. , а , с другой стороны, содержит средства языка типа Ассемблер : регистровые переменные, адресную арифметику, возможность работы с полями бит и т. д.

1.1. Структура программы

Программа на языке Си имеет следующую структуру:

```
#директивы препроцессора
. . . . .
#директивы препроцессора
функция а ( )
    операторы
функция в ( )
    операторы
void main ( )           //функция, с которой начинается
выполнение программы
    операторы
        описания
        присваивания
        функция
        пустой оператор
        составной
        выбора
        циклов
        перехода
```

Директивы препроцессора - управляют преобразованием текста программы до ее компиляции. Исходная программа, подготовленная на языке Си в виде текстового файла проходит 3 этапа обработки:

- 1) препроцессорное преобразование текста;
 - 2) компиляция;
 - 3) компоновка (редактирование связей или сборка) .
- После этих 3 этапов формируется исполняемый машинный код программы.

Задача препроцессора – преобразование текста программы до ее компиляции. Правила препроцессорной обработки определяет программист с помощью директив препроцессора. Директива начинается с #. Например,

1) #define – указывает правила замены в тексте.

```
#define ZERO 0.0
```

Означает, что каждое использование в программе имени ZERO будет заменяться на 0.0.

2) #include< имя заголовочного файла> – предназначена для включения в текст программы текста из каталога «Заголовочных файлов», поставляемых вместе со стандартными библиотеками. Каждая библиотечная функция Си имеет соответствующее описание в одном из заголовочных файлов. Список заголовочных файлов определен стандартом языка. Употребление директивы include не подключает соответствующую стандартную библиотеку, а только позволяют вставить в текст программы описания из указанного заголовочного файла. Подключение кодов библиотеки осуществляется на этапе компоновки, т. е. после компиляции. Хотя в заголовочных файлах содержатся все описания стандартных функций, в код программы включаются только те функции, которые используются в программе.

После выполнения препроцессорной обработки в тексте программы не остается ни одной препроцессорной директивы. Программа представляет собой набор описаний и определений, и состоит из набора функций. Среди этих функций всегда должна быть функция с именем main. Без нее программа не может быть выполнена. Перед именем функции помещаются сведения о типе возвращаемого функцией значения (тип результата). Если функция ничего не возвращает, то указывается тип void: void main (). Каждая функция, в том числе и main должна иметь набор параметров, он может быть пустым, тогда в скобках указывается (void).

За заголовком функции размещается тело функции. Тело функции – это последовательность определений, описаний и исполняемых операторов, заключенных в фигурные скобки. Каждое определение, описание или оператор заканчивается точкой с запятой.

Определения – вводят объекты (объект – это именованная область памяти, частный случай объекта – переменная), необходимые для представления в программе обрабатываемых данных. Примером являются

```
int y = 10 ; //именованная константа
float x ; //переменная
```

Описания – уведомляют компилятор о свойствах и именах объектов и функций, описанных в других частях программы.

Операторы – определяют действия программы на каждом шаге ее исполнения.

1.2. Константы и переменные

Константа – это значение, которое не может быть изменено. Синтаксис языка определяет 5 типов констант:

- символы;
- константы перечисляемого типа;
- вещественные числа;
- целые числа;
- нулевой указатель (NULL).

Переменные можно изменять. При задании значения переменной в соответствующую ей область памяти помещается код этого значения. Доступ к значению возможен через имя переменной, а доступ к участку памяти – по его адресу. Каждая переменная перед использованием в программе должна быть определена, т. е. ей должна быть выделена память. Размер участка памяти, выделяемой для переменной и интерпретация содержимого зависят от типа, указанного в определении переменной. Простейшая форма определения переменных:

тип список_имен_переменных;

Основные типы данных

тип данных	название	размер, бит	диапазон значений
unsigned char	беззнаковый целый длиной не менее 8 бит	8	0 . . 255
char	целый длиной не менее 8 бит	8	-128 . . 127
enum	перечисляемый	16	-32768 . . 32767
unsigned int	беззнаковый целый	16	0 . . 65535
short int (short)	короткий целый	16	-32768 . . 32767
unsigned short	беззнаковый короткий целый	16	0 . . 65535
int	целый	16	-32768 . . 32767
unsigned long	беззнаковый длинный целый	32	0 . . 4294967295
long	длинный целый	32	-214748348 . . 2147483647
float	вещественный одинарной точности	32	3.4E-38 . . 3.4E+38
double	вещественный двойной точности	64	1.7E-308 . . 1.7E+308
long double	вещественный	80	3.4E-4932 . .

	максимальной точности		1.1E+4932
--	--------------------------	--	-----------

В соответствии с синтаксисом языка переменные автоматической памяти после определения по умолчанию имеют неопределенные значения. Переменным можно присваивать начальные значения, явно указывая их в определениях:

тип имя_переменной = начальное_значение;

Этот прием называется инициализацией.

Примеры:

float pi = 3.14 , cc=1.3456;

unsigned int year = 1999;

1.3. Операции

Унарные:

&	получение адреса операнда
*	обращение по адресу (разыменование)
-	унарный минус, меняет знак арифметического операнда
~	побитовое инвертирование внутреннего двоичного кода (побитовое отрицание)
!	логическое отрицание (НЕ). В качестве логических значений используется 0 – ложь и не 0 – истина, отрицанием 0 будет 1, отрицанием любого ненулевого числа будет 0.
++	увеличение на единицу: префиксная операция – увеличивает операнд до его использования, постфиксная операция увеличивает операнд после его использования.
--	уменьшение на единицу: префиксная операция – уменьшает операнд до его использования, постфиксная операция уменьшает операнд после его использования.
sizeof	вычисление размера (в байтах) для объекта того типа, который имеет операнд

Бинарные операции.

Аддитивные:

+	бинарный плюс (сложение арифметических операндов)
-	бинарный минус (вычитание арифметических операндов)

Мультипликативные:

*	умножение операндов арифметического типа
/	деление операндов арифметического типа (если операнды целочисленные, то выполняется целочисленное деление)
%	получение остатка от деления целочисленных операндов

Операции сдвига (определены только для целочисленных операндов).

Формат выражения с операцией сдвига:

операнд_левый операция_сдвига операнд_правый

<<	сдвиг влево битового представления значения левого целочисленного операнда на количество разрядов, равное значению правого операнда
>>	сдвиг вправо битового представления значения правого целочисленного операнда на количество разрядов, равное значению правого операнда

Поразрядные операции:

&	поразрядная конъюнкция (И) битовых представлений значений целочисленных операндов
	поразрядная дизъюнкция (ИЛИ) битовых представлений значений целочисленных операндов
^	поразрядное исключающее ИЛИ битовых представлений значений целочисленных операндов

Операции сравнения:

<	меньше, чем
>	больше, чем
<=	меньше или равно
>=	больше или равно
==	равно
!=	не равно

Логические бинарные операции:

&&	конъюнкция (И) целочисленных операндов или отношений, целочисленный результат ложь (0) или истина (1)
	дизъюнкция (ИЛИ) целочисленных операндов или отношений, целочисленный результат ложь (0) или истина (1)

Условная операция.

В отличие от унарных и бинарных операций в ней используется три операнда.

Выражение1 ? Выражение2 : Выражение3;

Первым вычисляется значение выражения1. Если оно истинно, то вычисляется значение выражения2, которое становится результатом. Если при вычислении выражения1 получится 0, то в качестве результата берется значение выражения3.

Например:

$x < 0 ? -x : x$; //вычисляется абсолютное значение x .

Операция явного (преобразования) приведения типа.

Существует две формы: каноническая и функциональная:

1) (имя_типа) операнд

2) имя_типа (операнд)

Приоритеты операций.

Ранг	Операции
1	() [] -> .
2	! ~ - ++ -- & * (тип) sizeof тип()
3	* / % (мультипликативные бинарные)
	+ - (аддитивные бинарные)
5	<< >> (поразрядного сдвига)
6	< > <= >= (отношения)
7	== != (отношения)

8	& (поразрядная конъюнкция «И»)
9	^ (поразрядное исключающее «ИЛИ»)
10	(поразрядная дизъюнкция «ИЛИ»)
11	&& (конъюнкция «И»)
12	(дизъюнкция «ИЛИ»)
13	? : (условная операция)
14	= *= /= %= -= &= ^= = <=>= (операция присваивания)
15	, (операция запятая)

1.4. Выражения

Из констант, переменных, разделителей и знаков операций можно конструировать выражения. Каждое выражение состоит из одного или нескольких операндов, символов операций и ограничителей, в качестве которых чаще всего выступают квадратные скобки. Если выражение формирует целое или вещественное число, то это арифметическое выражение. В арифметических выражениях допустимы операции: + - * / %.

Отношение - это пара арифметических выражений, объединенных знаком операции отношения. Логический тип в Си отсутствует, поэтому принято, что отношение имеет ненулевое значение, если оно истинно и 0, если оно ложно.

1.5. Ввод и вывод

1.5.1. Ввод и вывод в стандартном Си

Обмен данными с внешним миром программа на стандартном Си реализует с помощью библиотеки функций ввода-вывода

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

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

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

```
printf ( "Значение числа Пи равно %f\n", pi);
```

Форматная строка может содержать

- 1) символы печатаемые текстуально;
- 2) спецификации преобразования
- 3) управляющие символы.

Каждому аргументу соответствует своя спецификация преобразования:

%d - десятичное целое число;

%f - число с плавающей точкой;

%c - символ;

%s - строка.

\n - управляющий символ новая строка.

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

В качестве аргументов используются указатели. Например:

```
scanf(" %d%f ", &x,&y);
```

1.5.2. Ввод и вывод в Си++

Используется библиотечный файл `iostream.h`, в котором определены стандартные потоки ввода данных от клавиатуры `cin` и вывода данных на экран дисплея `cout`, а также соответствующие операции

- 1) `<<` – операция записи данных в поток;
- 2) `>>` – операция чтения данных из потока.

Например:

```
#include <iostream.h>
. . . . .
cout << "\nВведите количество элементов: ";
cin >> n;
```

2. Постановка задачи

1. Вычислить значение выражения при различных вещественных типах данных (`float` и `double`). Вычисления следует выполнять с использованием промежуточных переменных. Сравнить и объяснить полученные результаты.
2. Вычислить значения выражений. Объяснить полученные результаты.

3. Варианты

№	Задание 1	Задание 2
1	$\frac{(a+b)^2 - (a^2 + 2ab)}{b^2},$ при $a=1000, b=0.0001$	1) $n+++m$ 2) $m-- > n$ 3) $n-- > m$
2	$\frac{(a-b)^2 - (a^2 - 2ab)}{b^2},$ при $a=1000, b=0.0001$	1) $++n+++m$ 2) $m++<n$ 3) $n++>m$
3	$\frac{(a+b)^3 - (a^3 + 3a^2b)}{3ab^2 + b^3},$ при $a=1000, b=0.0001$	1) $n---m$ 2) $m--<n$ 3) $n++>m$
4	$\frac{(a+b)^3 - (a^3)}{3ab^2 + b^3 + 3a^2b},$ при $a=1000, b=0.0001$	1) $n++*m$ 2) $n++<m$ 3) $m-- > m$
5	$\frac{(a-b)^3 - (a^3 - 3a^2b)}{b^3 - 3ab^2},$ при $a=1000, b=0.0001$	1) $- -m-++n$ 2) $m*n<n++$ 3) $n-- > m++$
6	$\frac{(a-b)^3 - (a^3 - 3ab^2)}{b^3 - 3a^2b},$ при $a=1000, b=0.0001$	1) $m-++n$ 2) $++m>--n$ 3) $--n<++m$

7	$\frac{(a-b)^3 - (a^3)}{b^3 - 3ab^2 - 3a^2b},$ <p>при $a=1000, b=0.0001$</p>	1) $m+--n$ 2) $m++<++n$ 3) $n--<--m$
8	$\frac{(a+b)^4 - (a^4 + 4a^3b + 6a^2b^2)}{4ab^3 + b^4},$ <p>при $a=100, b=0.001$</p>	1) $n++-m$ 2) $m-->n$ 3) $n-->m$
9	$\frac{(a+b)^4 - (a^4 + 4a^3b)}{6a^2b^2 + 4ab^3 + b^4},$ <p>при $a=100, b=0.001$</p>	1) $++n^{*}+++m$ 2) $m++<n$ 3) $n++>m$
10	$\frac{(a-b)^4 - (a^4 - 4a^3b + 6a^2b^2)}{b^4 - 4ab^3},$ <p>при $a=100, b=0.001$</p>	1) $n---m$ 2) $m--<n$ 3) $n++>m$
11	$\frac{(a-b)^4 - (a^4 - 4a^3b)}{6a^2b^2 - 4ab^3 + b^4},$ <p>при $a=100, b=0.001$</p>	1) $n++^{*}m$ 2) $n++<m$ 3) $m-->m$
12	$\frac{(a+b)^2 - (a^2 + 2ab)}{b^2},$ <p>при $a=1000, b=0.0001$</p>	1) $--m-++n$ 2) $m^{*}n<n++$ 3) $n-->m++$
13	$\frac{(a-b)^2 - (a^2 - 2ab)}{b^2},$ <p>при $a=1000, b=0.0001$</p>	1) $m-++n$ 2) $++m>--n$ 3) $--n<++m$
14	$\frac{(a+b)^3 - (a^3 + 3a^2b)}{3ab^2 + b^3},$ <p>при $a=1000, b=0.0001$</p>	1) $m+--n$ 2) $m++<++n$ 3) $n--<--m$
15	$\frac{(a+b)^3 - (a^3)}{3ab^2 + b^3 + 3a^2b},$ <p>при $a=1000, b=0.0001$</p>	1) $n++-m$ 2) $m-->n$ 3) $n-->m$
16	$\frac{(a-b)^3 - (a^3 - 3a^2b)}{b^3 - 3ab^2},$ <p>при $a=1000, b=0.0001$</p>	1) $++n^{*}+++m$ 2) $m++<n$ 3) $n++>m$
17	$\frac{(a-b)^3 - (a^3 - 3a^2b)}{b^3 - 3a^2b},$ <p>при $a=1000, b=0.0001$</p>	1) $n---m$ 2) $m--<n$ 3) $n++>m$
18	$\frac{(a-b)^3 - (a^3)}{b^3 - 3ab^2 - 3a^2b},$ <p>при $a=1000, b=0.0001$</p>	1) $n++^{*}m$ 2) $n++<m$ 3) $m-->m$

19	$\frac{(a+b)^4 - (a^4 + 4a^3b + 6a^2b^2)}{4ab^3 + b^4},$ при $a=100, b=0.001$	1) - -m-++n 2) m*n<n++ 3) n-- > m++
20	$\frac{(a+b)^4 - (a^4 + 4a^3b)}{6a^2b^2 + 4ab^3 + b^4},$ при $a=100, b=0.001$	1) m-++n 2) ++m>--n 3) --n<+m
21	$\frac{(a-b)^4 - (a^4 - 4a^3b + 6a^2b^2)}{b^4 - 4ab^3},$ при $a=100, b=0.001$	1) n++-m 2) m-- >n 3) n-- >m
22	$\frac{(a-b)^4 - (a^4 - 4a^3b)}{6a^2b^2 - 4ab^3 + b^4},$ при $a=100, b=0.001$	1) ++n*+m 2) m++<n 3) n++>m
23	$\frac{(a+b)^3 - (a^3 + 3a^2b)}{3ab^2 + b^3},$ при $a=1000, b=0.0001$	1) n---m 2) m--<n 3) n++>m
24	$\frac{(a+b)^3 - (a^3)}{3ab^2 + b^3 + 3a^2b},$ при $a=1000, b=0.0001$	1) n++*m 2) n++<m 3) m-- >m
25	$\frac{(a-b)^3 - (a^3 - 3a^2b)}{b^3 - 3ab^2},$ при $a=1000, b=0.0001$	1) - -m-++n 2) m*n<n++ 3) n-- > m++

4. Методические указания

1. Для ввода и вывода данных использовать операции >> и << и стандартные потоки cin и cout.
2. Для вычисления степени можно использовать функцию pow(x,y) из библиотечного файла math.h.
3. При выполнении задания 1 надо использовать вспомогательные переменные для хранения промежуточных результатов.
Например: `c=pow(a,3);d=3*a*a*b;e=3*a*b*b;f=pow(b,3);`

5. Содержание отчета

1. Постановка задачи.
2. Программа решения задания1.
3. Результаты работы программы для данных типа float.
4. Результаты работы программы для данных типа double.
5. Объяснение результатов.
6. Программа решения задания2.
7. Результаты работы программы.

8. Объяснение результатов.

Лабораторная работа №2.

"Использование основных операторов языка Си"

Цель : Получение навыков в выборе и использовании операторов Си++; знакомство с итерационными процессами.

1. Краткие теоретические сведения

Операторы управления работой программы называют управляющими конструкциями программы. К ним относят:

- составные операторы;
- операторы выбора;
- операторы циклов;
- операторы перехода.

1.1. Составные операторы

К составным операторам относят собственно составные операторы и блоки. В обоих случаях это последовательность операторов, заключенная в фигурные скобки. Блок отличается от составного оператора наличием определений в теле блока. Например:

```
{
n++;
summa+=n;
}
```

это составной оператор

```
{
int n=0;
n++;
summa+=n;
}
```

это блок

1.2. Операторы выбора

Операторы выбора – это условный оператор и переключатель. Условный оператор имеет полную и сокращенную форму.

```
if ( <выражение-условие> ) <оператор>;
//сокращенная форма
```

В качестве <выражения-условия> могут использоваться арифметическое выражение, отношение и логическое выражение. Если значение <выражения-условия> отлично от нуля (т. е. истинно), то выполняется оператор. Например:

```
if (x<y&& x<z) min=x;
if ( <выражение-условие> ) <оператор1>;
//полная форма
else <оператор2>;
```

Если значение <выражения-условия> отлично от нуля, то выполняется оператор1, при нулевом значении <выражения-условия> выполняется оператор2. Например:

```
if (d>=0)
{
```

```

x1=(-b-sqrt(d))/(2*a);
x2=(-b+sqrt(d))/(2*a);
cout<< "\nx1="<<x1<<"x2="<<x2;
}
else cout<<"\nРешения нет";

```

Переключатель определяет множественный выбор.

```

switch (<выражение>)
{
case <константа1> : <оператор1 >;
case <константа2> : <оператор2 >;
. . . . .
default: <операторы>;

```

При выполнении оператора switch, вычисляется выражение, записанное после switch и его значение последовательно сравнивается с константами, которые записаны следом за case. При первом же совпадении выполняются операторы помеченные данной меткой. Если выполненные операторы не содержат оператора перехода, то далее выполняются операторы всех следующих вариантов, пока не появится оператор перехода или не закончится переключатель. Если значение выражения, записанного после switch не совпало ни с одной константой, то выполняются операторы, которые следуют за меткой default. Метка default может отсутствовать.

Пример:

```

switch ( number )
{
case 1 : cout<< "число=1";break;
case 2 : cout<< "2 * 2"<<number * number;
case 3 : cout<< "3 * 3"<<number * number; break;
case 4 : cout<< number<<"- это замечательное число";
break;
default: cout<< "Конец работы программы";
}

```

1.3. Операторы циклов

1. Цикл с предусловием:

```

while (<выражение-условие>)
<тело_цикла> ;

```

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

2. Цикл с постусловием:

```

do
<тело_цикла>;
while (<выражение-условие>);

```

Тело цикла выполняется до тех пор, пока <выражение-условие> истинно.

3. Цикл с параметром:

```

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

```

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

Примеры использования цикла с параметром.

1) Уменьшение параметра:

```
for ( n=10; n>0; n-- )
{ <тело цикла>; }
```

2) Изменение шага корректировки:

```
for ( n=2; n>60; n+=13 )
{ <тело цикла>; }
```

3) Возможность проверять условие отличное от условия, которое налагается на число итераций:

```
for ( num=1; num*num*num<216; num++ )
{ <тело цикла>; }
```

4) Коррекция может осуществляться не только с помощью сложения или вычитания:

```
for ( d=100.0; d<150.0; d*=1.1 )
{ <тело цикла>; }
for ( x=1; y<=75; y=5*(x++)+10 )
{ <тело цикла>; }
```

5) Можно использовать несколько инициализирующих или корректирующих выражений:

```
for ( x=1, y=0; x<10; x++; y+=x );
```

1.4. Операторы перехода

Операторы перехода выполняют безусловную передачу управления.

1) break – оператор прерывания цикла.

```
{
< операторы>
if (<выражение_условие>) break;
<операторы>
}
```

Т. е. оператор break целесообразно использовать, когда условие продолжения итераций надо проверять в середине цикла.

Пример:

```
// ищет сумму чисел вводимых с клавиатуры до тех пор,
пока не будет введено 100 чисел или 0
for(s=0, i=1; i<100; i++)
{
```

```
cin>>x;
if( x==0) break; // если ввели 0, то суммирование
заканчивается
s+=x;
}
```

2) continue - переход к следующей итерации цикла. Он используется, когда тело цикла содержит ветвления.

Пример:

```
//ищет количество и сумму положительных чисел
for( k=0,s=0,x=1;x!=0;)
{
cin>>x;
if (x<=0) continue;
k++;s+=x;
}
```

2. Постановка задачи

Используя оператор цикла, найти сумму элементов, указанных в конкретном варианте. Результат напечатать, снабдив соответствующим заголовком.

3. Варианты

1) Найти сумму целых положительных чисел, кратных 3 и меньших 200.

2) Найти сумму целых положительных четных чисел, меньших 100.

3) Найти сумму целых положительных нечетных чисел, меньших 200.

4) Найти сумму целых положительных чисел, больших 20, меньших 100 и кратных 3

5) Найти сумму ряда с точностью $\varepsilon=10^{-4}$, общий член которого

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

6) Найти сумму ряда с точностью $\varepsilon=10^{-4}$, общий член которого

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

7) Найти сумму ряда с точностью $\varepsilon=10^{-4}$, общий член

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

8) Найти сумму ряда с точностью $\varepsilon=10^{-4}$, общий член которого

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

9) Найти сумму ряда с точностью $\varepsilon=10^{-4}$, общий член которого

$$a_n = \frac{10^n}{n!}$$

10) Найти сумму ряда с точностью $\varepsilon=10^{-4}$, общий член которого

$$a_n = \frac{n!}{(2n)!}$$

11) Найти сумму ряда с точностью $\varepsilon=10^{-4}$, общий член которого

$$a_n = \frac{n!}{n^n}$$

12) Найти сумму ряда с точностью $\varepsilon=10^{-4}$, общий член которого

$$a_n = \frac{2^n n!}{n^n}$$

13) Найти сумму ряда с точностью $\varepsilon=10^{-4}$, общий член которого

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

14) Найти сумму ряда с точностью $\varepsilon=10^{-4}$, общий член которого

$$a_n = \frac{n!}{3n^n}$$

15) Найти сумму ряда с точностью $\varepsilon=10^{-4}$, общий член которого

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

16) Найти сумму ряда с точностью $\varepsilon=10^{-4}$, общий член которого

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

17) Найти сумму ряда с точностью $\varepsilon=10^{-4}$, общий член которого

$$a_n = 10^{-n}(n-1)!$$

18) Найти сумму ряда с точностью $\varepsilon=10^{-4}$, общий член которого

$$a_n = \frac{n^3}{(3n-3)!}$$

19) Найти сумму ряда с точностью $\varepsilon=10^{-4}$, общий член которого

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

20) Найти сумму ряда с точностью $\varepsilon=10^{-4}$, общий член которого

$$a_n = e^n \cdot 100^{-n^2}$$

21) Найти сумму 13 членов ряда, в котором

$$a_n = \frac{\ln(n!)}{n^2}$$

22) Найти сумму 15 членов ряда, в котором

$$a_n = \frac{n^{\ln n}}{(\ln n)^n}$$

23) Найти сумму 10 членов ряда, в котором

$$a_n = \frac{n!}{n^{\sqrt{n}}}$$

24) Найти сумму 9 членов ряда, в котором

$$a_n = e^{-\sqrt{n}}$$

25) Найти сумму 7 членов ряда, в котором

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

3. Содержание отчета

1. Постановка задачи.
2. Текст программы.
3. Результат решения конкретного варианта.

4. Методические указания

1. При определении суммы членов ряда следует использовать рекуррентную формулу для получения следующего члена ряда. Например, требуется найти сумму ряда с точностью $\varepsilon=10^{-4}$,

$$a_n = \frac{2(n!)^2}{(3(2n)!)!}.$$

общий член которого

Для получения рекуррентной формулы вычислим отношение:

$$\frac{a_{n+1}}{a_n} = \frac{2((n+1)!)^2 \cdot 3(2n)!}{3(2n+2)! \cdot 2(n!)^2} = \frac{n+1}{2(2n+1)},$$

откуда:

$$a_{n+1} = a_n \cdot \frac{(n+1)}{2(2n+1)}.$$

2. При составлении программы считать, что точность достигнута, если $a_n < \varepsilon$

Лабораторная работа №3

"Вычисление функций с использованием их разложения в степенной ряд"

Цель: Практика в организации итерационных и арифметических циклов.

1. Краткие теоретические сведения

Действительная функция $f(x)$ называется аналитической в точке ε , если в некоторой окрестности $|x - \varepsilon| < R$ этой точки функция разлагается в степенной ряд (ряд Тейлора):

$$f(x) = f(\varepsilon) + f'(\varepsilon)(x - \varepsilon) + \frac{f''(\varepsilon)}{2!}(x - \varepsilon)^2 + \dots + \frac{f^{(n)}(\varepsilon)}{n!}(x - \varepsilon)^n + \dots \quad (1)$$

При $\varepsilon = 0$ получаем ряд Маклорена:

$$f(x) = f(0) + f'(0)x + \frac{f''(0)}{2!}x^2 + \dots + \frac{f^{(n)}(0)}{n!}x^n + \dots \quad (2)$$

Разность $R_n(x) = f(x) - \sum_{k=0}^n \frac{f^{(k)}(\varepsilon)}{k!}(x - \varepsilon)^k$

(3)

называется остаточным членом и представляет собой ошибку при замене функции $f(x)$ полиномом Тейлора.

Для ряда Маклорена

$$R_n(x) = \frac{f^{(n+1)}(\theta \cdot x)}{(n+1)!} x^{n+1} \quad \text{где } 0 < \theta < 1. \quad (4)$$

Таким образом, вычисление значения функции можно свести к вычислению суммы числового ряда

$$a_1 + a_2 + \dots + a_n + \dots \quad (5)$$

Известно, что числовой ряд называется сходящимся, если существует предел последовательности его частных сумм:

$$S = \lim_{n \rightarrow \infty} S_n, \quad (6)$$

где $S_n = a_1 + a_2 + \dots + a_n + \dots$

Число S называется суммой ряда.

Из формулы (13) получаем $S = S_n + R_n$,

где R_n – остаток ряда, причем $R \rightarrow 0$ при $n \rightarrow \infty$.

Для нахождения суммы S сходящегося ряда (5) с заданной точностью ε нужно выбрать число слагаемых n столь большим, чтобы имело место неравенство $|R_n| < \varepsilon$.

Тогда частная сумма S_n приближенно может быть принята за точную сумму S ряда (5).

Приближенно n выбрать так, чтобы имело место неравенство

$$|S_{n+1} - S_n| < \varepsilon \quad \text{или} \quad a_n < \varepsilon.$$

Задача сводится к замене функции степенным рядом и нахождению суммы некоторого количества слагаемых

$S = \sum a_n(x, n)$ при различных параметрах суммирования x .

Каждое слагаемое суммы зависит от параметра x и номера n , определяющего место этого слагаемого в сумме.

Обычно формула общего члена суммы принадлежит одному из следующих трех типов:

$$а) \frac{x^n}{n!}; \quad (-1)^n \frac{x^{2n+1}}{(2n+1)!}; \quad \frac{x^{2n}}{(2n)!};$$

$$б) \frac{\cos(nx)}{n}; \quad \frac{\sin(2n-1)x}{2n-1}; \quad \frac{\cos(2nx)}{4n^2-1};$$

$$в) \frac{x^{4n+1}}{4n+1}; \quad (-1)^n \frac{\cos(nx)}{n^2}; \quad \frac{n^2+1}{n!} \left(\frac{x}{2}\right)^n.$$

В случае а) для вычисления члена суммы a_n целесообразно использовать рекуррентные соотношения, т. е. выражать последующий член суммы через предыдущий: $a_{n+1} = \psi(x, n) a_n$. Это позволит существенно сократить объем вычислительной работы. Кроме того, вычисление члена суммы по общей формуле в ряде случаев невозможно (например из-за наличия $n!$).

В случае б) применение рекуррентных соотношений нецелесообразно. Вычисления будут наиболее эффективными, если каждый член суммы вычислять по общей формуле $a_n = \phi(x, n)$.

В случае в) член суммы целесообразно представить в виде двух сомножителей, один из которых вычисляется по рекуррентному соотношению, а другой непосредственно $a_n = \phi(x, n) * c_n(x, n)$, где $c_n = c_{n-1} \psi(x, n)$.

2. Постановка задачи

Для x изменяющегося от a до b с шагом $(b-a)/k$, где $(k=10)$, вычислить функцию $f(x)$, используя ее разложение в степенной ряд в двух случаях:

а) для заданного n ;

б) для заданной точности ε ($\varepsilon=0.0001$).

Для сравнения найти точное значение функции.

3. Варианты

№	функция	диапазон изменения аргумента	n	сумма
1	$y = 3^x$	$0,1 \leq x \leq 1$	10	$S = 1 + \frac{\ln 3}{1!} x + \frac{\ln^2 3}{2!} x^2 + \dots + \frac{\ln^n 3}{n!} x^n$

2	$y = -\ln \left 2 \sin \frac{x}{2} \right $	$\frac{\pi}{5} \leq x \leq \frac{9\pi}{5}$	40	$S = \cos x + \frac{\cos 2x}{2} + \dots + \frac{\cos nx}{n}$
3	$y = \sin X$	$0,1 \leq x \leq 1$	10	$S = x - \frac{x^3}{3!} + \dots + (-1)^n \frac{x^{2n+1}}{(2n+1)!}$
4	$y = X \arctg X - \ln \sqrt{1+x^2}$	$0,1 \leq x \leq 0,8$	10	$S = \frac{x^2}{2} - \frac{x^4}{12} + \dots + (-1)^{n+1} \frac{x^{2n}}{2n(2n-1)}$
5	$y = e^x$	$1 \leq x \leq 2$	15	$S = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \dots + \frac{x^n}{n!}$
6	$y = e^{x \cos \frac{\pi}{4}} \cdot \cos(x \sin \frac{\pi}{4})$	$0,1 \leq x \leq 1$	25	$S = 1 + \frac{\cos \frac{\pi}{4}}{1!} x + \dots + \frac{\cos n \frac{\pi}{4}}{n!} x^n$
7	$y = \cos x$	$0,1 \leq x \leq 1$	10	$S = 1 - \frac{x^2}{2!} + \dots + (-1)^n \frac{x^{2n}}{(2n)!}$
8	$y = \frac{x \sin \frac{\pi}{4}}{1 - 2x \cos \frac{\pi}{4} + x^2}$	$0,1 \leq x \leq 0,8$	40	$S = x \sin \frac{\pi}{4} + x^2 \sin 2 \frac{\pi}{4} + \dots + x^n \sin n \frac{\pi}{4}$
9	$y = \frac{1}{4} \ln \frac{1+x}{1-x} + \frac{1}{2} \arctg X$	$0,1 \leq x \leq 0,8$	3	$S = x + \frac{x^5}{5} + \dots + \frac{x^{4n+1}}{4n+1}$
10	$y = e^{\cos x} \cos(\sin x)$	$0,1 \leq x \leq 1$	20	$S = 1 + \frac{\cos x}{1!} + \dots + \frac{\cos nx}{n!}$
11	$y = (1+2x^2)e^{x^2}$	$0,1 \leq x \leq 1$	10	$S = 1 + 3x^2 + \dots + \frac{2n+1}{n!} x^{2n}$
12	$y = -\frac{1}{2} \ln(1 - 2x \cos \frac{\pi}{3} + x^2)$	$0,1 \leq x \leq 0,8$	35	$S = \frac{x \cos \frac{\pi}{3}}{1} + \frac{x^2 \cos 2 \frac{\pi}{3}}{2} + \dots + \frac{x^n \cos n \frac{\pi}{3}}{n}$
13	$y = \frac{1}{2} \ln x$	$0,2 \leq x \leq 1$	10	$S = \frac{x-1}{x+1} + \frac{1}{3} \left(\frac{x-1}{x+1} \right)^3 + \dots + \frac{1}{2n+1} \left(\frac{x-1}{x+1} \right)^{2n+1}$
14	$y = \frac{1}{4} \left(x^2 - \frac{\pi^2}{3} \right)$	$\frac{\pi}{5} \leq x \leq \pi$	20	$S = -\cos x + \frac{\cos 2x}{2^2} + \dots + (-1)^n \frac{\cos nx}{n^2}$
15	$y = \frac{1+x^2}{2} \arctg X - \frac{x}{2}$	$0,1 \leq x \leq 1$	30	$S = \frac{x^3}{3} - \frac{x^5}{15} + \dots + (-1)^{n+1} \frac{x^{2n+1}}{4n^2-1}$
16	$y = \frac{\pi^2}{8} - \frac{\pi}{4} x $	$\frac{\pi}{5} \leq x \leq \pi$	40	$S = \cos x + \frac{\cos 3x}{3^2} + \dots + \frac{\cos(2n-1)x}{(2n-1)^2}$
17	$y = \frac{e^x + e^{-x}}{2}$	$0,1 \leq x \leq 1$	10	$S = 1 + \frac{x^2}{2!} + \dots + \frac{x^{2n}}{(2n)!}$

18	$y = \frac{1}{2} - \frac{\pi}{4} \sin x $	$0,1 \leq x \leq 0,8$	50	$S = \frac{\cos 2x}{3} + \frac{\cos 4x}{15} + \dots + \frac{\cos 2nx}{4n^2 - 1}$
19	$y = e^{2x}$	$0,1 \leq x \leq 1$	20	$S = 1 + \frac{2x}{1!} + \dots + \frac{(2x)^n}{n!}$
20	$y = (\frac{x^2}{4} + \frac{x}{2} + 1)e^{x/2}$	$0,1 \leq x \leq 1$	30	$S = 1 + 2\frac{x}{2} + \dots + \frac{n^2 + 1}{n!} (\frac{x}{2})^n$
21	$y = \arctg X$	$0,1 \leq x \leq 1$	40	$S = x - \frac{x^3}{3} + \dots + (-1)^n \frac{x^{2n+1}}{2n+1}$
22	$y = (1 - \frac{x^2}{2}) \cos x - \frac{x}{2} \sin x$	$0,1 \leq x \leq 1$	35	$S = 1 - \frac{3}{2}x^2 + \dots + (-1)^n \frac{2n^2 + 1}{(2n)!} x^{2n}$
23	$y = 2(\cos^2 x - 1)$	$0,1 \leq x \leq 1$	15	$S = -\frac{(2x)^2}{2} + \frac{(2x)^4}{24} + \dots + (-1)^n \frac{(2x)^{2n}}{(2n)!}$
24	$y = \ln(\frac{1}{2 + 2x + x^2})$	$-2 \leq x \leq -0,4$	40	$S = -(1+x)^2 + \frac{(1+x)^4}{2} + \dots + (-1)^n \frac{(1+x)^{2n}}{n}$
25	$y = \frac{e^x - e^{-x}}{2}$	$0,1 \leq x \leq 1$	20	$S = x + \frac{x^3}{3!} + \dots + \frac{x^{2n+1}}{(2n+1)!}$

4. Методические указания

- Алгоритм решения задачи сводится к трем циклам, причем два из них вложены в третий. Внутренние циклы суммируют слагаемые при фиксированном параметре x , один (арифметический для заданного n), другой (итерационный для заданной точности ε). При организации этих циклов следует обратить внимание на правильный выбор формулы для вычисления элемента ряда a_n и правильное присвоение начальных значений переменным цикла. Внешний цикл организует изменение параметра x .
- Результаты расчетов отпечатать с следующим виде:

Вычисление функции

X=..... SN=..... SE=..... Y=.....

X=..... SN=..... SE=..... Y=.....

.....

X=..... SN=..... SE=..... Y=.....

Здесь X- значение параметра; SN- значение суммы для заданного n ; SE- значение суммы для заданной точности; Y- точное значение функции.

5. Содержание отчета

- Постановка задачи.
- Вариант задания.
- Математическая модель (формулы, по которым выполняются вычисления слагаемых ряда).

4. Программа.
5. Полученные результаты.

Лабораторная работа № 4

"Работа с одномерными массивами"

Цель: Получение навыков обработки одномерных массивов.

1. Краткие теоретические сведения

1.1. Определение массива

Определение массива содержит тип элементов, имя массива и количество элементов в массиве.

```
int mas[10];
```

0	1	2	3	4	5	6	7	8	9

Т. е. индексы элементов в массиве `mas` могут меняться от 0 до 9, всего в массиве 10 элементов.

1.2. Инициализация массива

Инициализация массивов возможна при их определении:

```
double d[] = {1, 2, 3, 4, 5};
```

Длина массива вычисляется компилятором по количеству значений перечисленных в фигурных скобках.

1.3. Указатели

Каждая переменная в программе это объект, имеющий имя и значение по имени можно обратиться к переменной и получить ее значение. Оператор присваивания (`=`) выполняет обратное действие: имени переменной ставится в соответствие значение.

```
a=10;
```

10

↑

&a

a

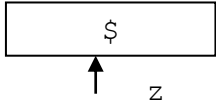
Выражение `&a` позволяет получить адрес участка памяти, выделенного переменной `a`. Операция `&` применима только к объектам имеющим имя и размещенным в памяти.

Имея возможность определить адрес переменной с помощью `&`, надо иметь возможность работать с этим адресом: сохранять его, передавать, преобразовывать. Для этого вводится понятие указателя. Указатель – это переменная, значением которой служит адрес объекта конкретного типа. Нулевой адрес обозначается константой `NULL`, которая определена в заголовочном файле `stdio.h`. Чтобы определить указатель надо сообщить на объект какого типа ссылается этот указатель.

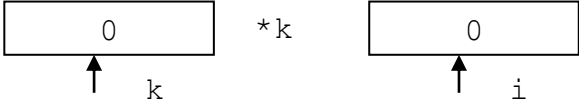
```
char *z;
int *k,*i;
float *f;
```

`*` – это операция разыменования. Операндом этой операции всегда является указатель. Результат операции – это тот объект, который адресуется указателем_операнд.

`*z=' $ ';`



`*k=*i=0;`



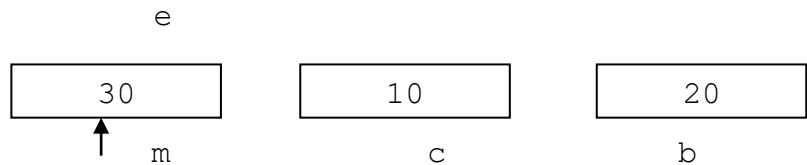
Пример:

`int e, c, b, *m;`

`.`

`m = &e ;`

`*m = c + b ;`



Операции над указателями.

- присваивание (=);
- получение значения объекта, на который ссылается указатель (*);
- получение адреса самого указателя (&).

Пример:

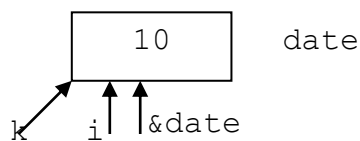
`int date = 10;`

`int *i, *k;`

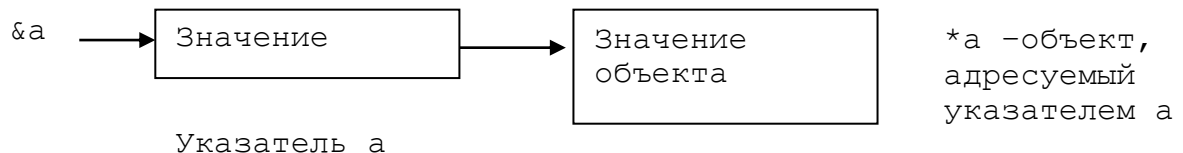
`i = &date;`

`k = i;`

`z = NULL;`



Подобно любым переменным переменная типа указатель имеет имя, адрес в памяти и значение.



С помощью унарных операций ++ и -- числовые значения переменных типа указатель меняются по разному, в зависимости от типа данных, с которым связаны эти переменные.

Пример:

`char *z;`

`int *k,*i;`

`float *f;`

`.`

`z++; // значение изменяется на 1`

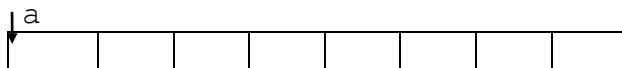
`i++; // значение изменяется на 2`

`f++; // значение изменяется на 4`

Т. е. при изменении указателя на 1, указатель переходит к началу следующего (предыдущего) поля той длины, которая определяется типом объекта, адресуемого указателем.

1.4. Указатели и массивы

Имя массива без индекса является указателем-константой, т. е. адресом первого элемента массива ($a[0]$).



```
*a == a[0] ;
*(a+1) == a[1];
. . . . .
*(a+i) == a[i];
```

В соответствии с синтаксисом в Си существуют только одномерные массивы, но их элементами, в свою очередь, тоже могут быть массивы.

```
int a[5][5];
```

Для двумерного массива:

```
a[m][n] == *(a[m]+n) == (*(a+m)+n);
```

2. Варианты заданий

1.

- 1) Сформировать одномерный массив целых чисел, используя датчик случайных чисел.
- 2) Распечатать полученный массив.
- 3) Удалить элемент с номером K.
- 4) Добавить после каждого четного элемента массива элемент со значением 0.
- 5) Распечатать полученный массив.

2.

- 1) Сформировать одномерный массив целых чисел, используя датчик случайных чисел.
- 2) Распечатать полученный массив.
- 3) Удалить первый элемент равный 0.
- 4) Добавить после каждого четного элемента массива элемент со значением $M[I-1] + 2$.
- 5) Распечатать полученный массив.

3.

- 1) Сформировать одномерный массив целых чисел, используя датчик случайных чисел.
- 2) Распечатать полученный массив.
- 3) Удалить все элементы равные 0.
- 4) Добавить после первого четного элемента массива элемент со значением $M[I-1] + 2$.
- 5) Распечатать полученный массив.

4.

- 1) Сформировать одномерный массив целых чисел, используя датчик случайных чисел.
- 2) Распечатать полученный массив.
- 3) Удалить элементы, индексы которых кратны 3.
- 4) Добавить после каждого отрицательного элемента массива элемент со значением $|M[I-1]|+1$.
- 5) Распечатать полученный массив.

5.

- 1) Сформировать одномерный массив целых чисел, используя датчик случайных чисел.
- 2) Распечатать полученный массив.
- 3) Удалить элементы кратные 7.
- 4) Добавить после каждого нечетного элемента массива элемент со значением 0.
- 5) Распечатать полученный массив.

6.

- 1) Сформировать одномерный массив целых чисел, используя датчик случайных чисел.
- 2) Распечатать полученный массив.
- 3) Удалить элемент с заданным номером.
- 4) Добавить после первого четного элемента массива элемент со значением $M[I-1]+2$.
- 5) Распечатать полученный массив.

7.

- 1) Сформировать одномерный массив целых чисел, используя датчик случайных чисел.
- 2) Распечатать полученный массив.
- 3) Удалить последний элемент равный 0.
- 4) Добавить после элемента массива с заданным индексом элемент со значением 100.
- 5) Распечатать полученный массив.

8.

- 1) Сформировать одномерный массив целых чисел, используя датчик случайных чисел.
- 2) Распечатать полученный массив.
- 3) Удалить все элементы с заданным значением.
- 4) Добавить перед каждым четным элементом массива элемент со значением 0.
- 5) Распечатать полученный массив.

9.

- 1) Сформировать одномерный массив целых чисел, используя датчик случайных чисел.
- 2) Распечатать полученный массив.
- 3) Удалить первый элемент с заданным значением.
- 4) Сдвинуть массив циклически на K элементов вправо.

5) Распечатать полученный массив.

10.

- 1) Сформировать одномерный массив целых чисел, используя датчик случайных чисел.
- 2) Распечатать полученный массив.
- 3) Удалить 5 первых элементов массива.
- 4) Добавить в конец массива 3 новых элемента.
- 5) Распечатать полученный массив.

11.

- 1) Сформировать одномерный массив целых чисел, используя датчик случайных чисел.
- 2) Распечатать полученный массив.
- 3) Удалить 5 последних элементов массива.
- 4) Добавить в начало массива 3 элемента с значением $M[I+1] + 2$.
- 5) Распечатать полученный массив.

12.

- 1) Сформировать одномерный массив целых чисел, используя датчик случайных чисел.
- 2) Распечатать полученный массив.
- 3) Поменять местами минимальный и максимальный элементы массива.
- 4) Удалить из массива все элементы превышающие его среднее значение более, чем на 10%.
- 5) Распечатать полученный массив.

13.

- 1) Сформировать одномерный массив целых чисел, используя датчик случайных чисел.
- 2) Распечатать полученный массив.
- 3) Удалить из массива все элементы совпадающие с его минимальным значением.
- 4) Добавить в начало массива 3 элемента с значением равным среднему арифметическому массива.
- 5) Распечатать полученный массив.

14.

- 1) Сформировать одномерный массив целых чисел, используя датчик случайных чисел.
- 2) Распечатать полученный массив.
- 3) Перевернуть массив и, если число элементов массива нечетное, удалить его средний элемент.
- 4) Добавить в начало массива 3 элемента с значением $M[I+10] - 2$.
- 5) Распечатать полученный массив.

15.

- 1) Реализовать с использованием массива двунаправленное кольцо (просмотр возможен в обе стороны, от последнего элемента можно перейти к первому).
- 2) Распечатать полученный массив, начиная с K -ого элемента и до $K-1$ (по кольцу влево).
- 3) Удалить из кольца первый и последний элементы.
- 4) Распечатать полученный массив, начиная с K -ого элемента (и до $K+1$ по кольцу вправо).

16.

Реализовать с использованием массива очередь (первый пришел, первый ушел), для чего организовать добавление, удаление элементов в массив и печать массива после каждой операции.

17.

- 1) Реализовать с использованием массива двунаправленное кольцо (просмотр возможен в обе стороны, от последнего элемента можно перейти к первому).
- 2) Распечатать полученный массив, начиная с K -ого элемента и до $K-1$ (по кольцу влево).
- 3) Добавить в кольцо первый и последний элементы.
- 4) Распечатать полученный массив, начиная с K -ого элемента (и до $K+1$ по кольцу вправо).

18.

- 1) Реализовать с использованием массива однонаправленное кольцо (просмотр возможен слева направо, от последнего элемента можно перейти к первому).
- 2) Распечатать полученный массив, начиная с K -ого элемента и до $K-1$.
- 3) Добавить в кольцо первый и последний элементы.
- 4) Удалить из кольца четные элементы.
- 5) Распечатать полученный массив, начиная с K -ого элемента и до $K-1$.

19.

- 1) Реализовать с использованием массива однонаправленное кольцо (просмотр возможен справа налево, от первого элемента можно перейти к последнему).
- 2) Распечатать полученный массив, начиная с K -ого элемента и до $K+1$.
- 3) Добавить в кольцо первый и последний элементы.
- 4) Удалить из кольца нечетные элементы.
- 5) Распечатать полученный массив, начиная с K -ого элемента и до $K+1$.

20.

- 1) Реализовать с использованием массива двунаправленное кольцо (просмотр возможен в обе стороны, от последнего элемента можно перейти к первому).
- 2) Распечатать полученный массив, начиная с K -ого элемента и до $K-1$ (по кольцу влево).
- 3) Добавить в кольцо после элементов с индексами кратными 5 элементы равные 0.
- 4) Распечатать полученный массив, начиная с K -ого элемента (и до $K+1$ по кольцу вправо).

21.

- 1) Реализовать с использованием массива двунаправленное кольцо (просмотр возможен в обе стороны, от последнего элемента можно перейти к первому).
- 2) Распечатать полученный массив, начиная с K -ого элемента и до $K-1$ (по кольцу влево).
- 3) Добавить в кольцо первый и 3 последних элемента.
- 4) Распечатать полученный массив, начиная с K -ого элемента (и до $K+1$ по кольцу вправо).

22.

Реализовать с использованием массива стек (первый пришел, последний ушел), для чего организовать добавление, удаление элементов в массив и печать массива после каждой операции.

23.

- 1) Реализовать с использованием массива двунаправленное кольцо (просмотр возможен в обе стороны, от последнего элемента можно перейти к первому).
- 2) Распечатать полученный массив, начиная с K -ого элемента и до $K-1$ (по кольцу влево).
- 3) Удалить из кольца все элементы совпадающие с его максимальным значением.
- 4) Распечатать полученный массив, начиная с K -ого элемента (и до $K+1$ по кольцу вправо).

24.

- 1) Реализовать с использованием массива однонаправленное кольцо (просмотр возможен слева направо, от последнего элемента можно перейти к первому).
- 2) Распечатать полученный массив, начиная с K -ого элемента и до $K-1$.
- 3) Упорядочить элементы по возрастанию
- 4) Удалить из кольца четные элементы.
- 5) Распечатать полученный массив, начиная с K -ого элемента и до $K-1$.

25.

- 1) Реализовать с использованием массива однонаправленное кольцо (просмотр возможен справа налево, от первого элемента можно перейти к последнему).
- 2) Распечатать полученный массив, начиная с K-ого элемента и до K+1.
- 3) Упорядочить элементы по убыванию
- 4) Удалить из кольца нечетные элементы.
- 5) Распечатать полученный массив, начиная с K-ого элемента и до K+1.

3. Методические указания

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

```
int N=100;
int a[N];
```

Затем пользователь вводит реальную длину массива (не больше N) и работает с массивом той длины, которую он сам указал. Остальные элементы (хотя память под них и будет выделена) не рассматриваются.

2) При уменьшении или увеличении длины массива необходимо изменять его реальную длину.

4. Содержание отчета

1. Вариант задания.
2. Текст программы.
3. Результат решения конкретного варианта.

Лабораторная работа №5 "Функции и массивы"

Цель : Организовать обработку массивов с использованием функций, научиться передавать массивы как параметры функций.

1. Краткие теоретические сведения

1.1. Функции

Функцию в Си можно рассматривать:

- как один из производных типов данных (наряду с массивами и указателями);
- как минимальный исполняемый модуль программы (подпрограмму).

Все функции имеют единый формат определения:

<тип><имя_функции>(<список_формальных_параметров>) , где
<тело_функции> , где

<тип> – либо void, если функция не возвращает значения, либо тип возвращаемого функцией значения;

<имя_функции> – либо main для основной функции, либо произвольный идентификатор, не совпадающий со служебными словами и именами других объектов программы;

<список_формальных_параметров> - либо пустой (), либо список, каждый элемент которого имеет вид:

<обозначение_типа><имя_параметра>

Например:

```
(int k )
```

```
(char i, char j, int z)
```

<тело_функции> - это часть определения функции, заключенная в фигурные скобки { }. Тело функции может быть либо составным оператором, либо блоком. Определения функций не могут быть вложенными.

Для передачи результата из функции в вызывающую функцию используется оператор return. Он может использоваться в двух формах:

- 1) return; - завершает функцию гн возвращающую никакого значения (т. е. перед именем функции указан тип void)
- 2) return <выражение>; - возвращает значение выражения, выражение должно иметь тип, указанный перед именем функции.

Если программист не пишет оператор return явно, то компилятор автоматически дописывает return в конец тела функции перед закрывающей фигурной скобкой}.

Пример:

```
int op (char c, int x, int y)
{
switch c
{
case '+' : return x+y;
case '-' : return x-y;
case '*' : return x*y;
case '/' : return x/y;
default: cout<<"\nОперация не определена";return 0;
}
}
```

Вызов функции осуществляется следующим образом:

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

где

<обозначение функции> - либо имя функции, либо указатель на функцию;

<список фактических параметров> - список выражений, количество которых равно числу формальных параметров функции. Между формальными и фактическими параметрами должно быть соответствие по типам.

Например:

```
c = op ( '+', 5 ,4 );
```

Синтаксис Си предусматривает только один способ передачи параметров - передача по значению (т. е. изменить значения параметров внутри функции нельзя). Но существует возможность косвенно изменить значения переменных передаваемых в виде параметров: с помощью указателя в вызываемую функцию можно передать адрес любого объекта из вызывающей программы. Если указатель

разыменовать, то получится значение, записанное по этому адресу.

Пример:

1)

//описание функции для обмена переменных a и b

```
void change (int a,int b)
```

```
{
```

```
int r;
```

```
r = a; a = b; b = r;
```

```
}
```

// вызов функции

```
change(a, b);
```

Обмена не произойдет, т. к. результат не будет передан в вызывающую программу.

2)

```
void change (int *a,int *b)
```

```
{
```

```
int r;
```

```
r = *a; *a = *b; *b = r;
```

```
}
```

// вызов функции

```
change(&a, &b);
```

При вызове передаются адреса, по которым находятся значения и выполняется обмен значений, которые находятся по этим адресам.

1.2. Массивы и строки как параметры функций

Если в качестве параметра функции используется обозначение массива, то на самом деле в функцию передается адрес первого элемента массива.

Пример:

//вычисление суммы элементов массива

//вариант 1

```
int sum (int n, int a[] )
```

```
{
```

```
int i,int s=0;
```

```
for( i=0; i<n; i++ )
```

```
s+=a[i]
```

```
return s;
```

```
}
```

```
void main()
```

```
{
```

```
int a[]={ 3, 5, 7, 9, 11, 13, 15 };
```

```
int s = sum( 7, a );
```

```
cout<<s;
```

```
}
```

//вариант 2

```
int sum (int n, int *a)
```

```
{
```

```
for(int i=0, s=0; i<n; s+=*(a+i),i++ );
```

```
return s;
```

```
}
```

```

void main()
{
int a[]={ 3, 5, 7, 9, 11, 13, 15 };
    int s = sum( 7, a );
cout<<s;
}

```

Строки в качестве фактических параметров могут быть определены либо как одномерные массивы типа `char[]`, либо как указатели типа `char*`. В отличие от обычных массивов в этом случае нет необходимости явно указывать длину строки.

2. Постановка задачи

Используя функции, решить указанную в варианте задачу. Массив должен передаваться в функцию как параметр.

3. Варианты

1. В двумерном массиве записаны слова, представляющие собой последовательность цифр, завершающихся 0. Необходимо распечатать слова через запятую, заключив печатную строку в скобки. Длина печатной строки 60 символов. Извлечение слова оформить в виде функции.
Например:
исходные данные - 123023402303450
234450234567010
234455677670450
результат -
(123,234,23,345) (23445,234567,1) (23445567767,45)
2. Написать функцию для обмена строк двумерного массива с ее помощью отсортировать массив по элементам третьего столбца.
3. Написать процедуру для суммирования матриц. С ее помощью сложить исходную матрицу и транспонированную (т. е. полученную поворотом исходной на 90 °).
4. Написать функцию для удаления строки из двумерного массива. Оставшиеся строки должны быть расположены плотно, недостающие элементы заменяются 0. С помощью разработанных функций исключить из массива строки с номерами от A до B.
5. Определить является ли матрица ортонормированной, т. е. такой, что скалярное произведение каждой пары различных строк равно 0, а скалярное произведение строки самой на себя равно 1.
6. Элемент матрицы является седловой точкой, если он является наименьшим в своей строке и наибольшим в своем столбце (или наоборот: наибольшим в своей строке и наименьшим в своем столбце). Для заданной матрицы определить все седловые точки.

7. Написать процедуру обмена столбца и строки двумерного массива. С ее помощью поменять местами те строки и столбцы, первые элементы которых совпадают.
8. Написать функцию транспонирования квадратной матрицы (т.е. поворота исходной матрицы на 90°). С ее помощью определить является ли заданная матрица симметрической. (Матрица называется симметрической, если транспонированная матрица равна исходной).
9. Написать функцию для вычисления суммы элементов квадратной матрицы, которые расположены ниже главной диагонали. С ее помощью найти максимальное значение такой суммы в n матрицах.
10. Написать функцию, проверяющую есть ли отрицательные элементы в указанной строке двумерного массива. Удалить из массива все строки с отрицательными элементами, удаленная строка заполняется 0 и переносится в конец массива.
11. Написать функцию, проверяющую по возрастанию или убыванию упорядочена указанная строка двумерного массива. Упорядочить по возрастанию все строки двумерного массива, которые неупорядочены по убыванию.
12. Написать функцию, для поиска максимального элемента в указанной строке двумерного массива. Сдвинуть в двумерном массиве все строки циклически вправо на количество элементов равное максимальному элементу в этой строке.
13. Определить можно ли в двумерном массиве найти такой столбец, который разбивает массив на два так, что сумма элементов в первом больше, чем сумма элементов во втором. Сам столбец в разбиваемые части не входит.
14. Вычислить произведение всех столбцов массива, у которых первый элемент больше элементов расположенных на главной и побочной диагонали.
15. Задан двумерный массив. Найти сумму элементов первого столбца без одного последнего элемента, сумму элементов второго столбца без двух последних, сумму элементов третьего столбца без трех последних и т. д. Последний столбец не обрабатывается. Среди найденных сумм найти максимальную.
16. Задан двумерный массив $N \times N$. Разрешается произвольно переставлять элементы внутри любого столбца. Проверить, можно ли выполнив конечное количество перестановок в столбцах, расположить на побочной диагонали элементы так, чтобы он возрастали.
17. Задан двумерный массив $N \times M$. Найти в нем подмассив 3×3 , сумма элементов которого максимальна. N и M могут быть не кратны трем.
18. Задан двумерный массив $N \times N$. Последовательно рассматриваются квадратные подмассивы, правый верхний элемент которых лежит на побочной диагонали. В каждом

таком подмассиве находится максимальный элемент. Путем перестановок строк и столбцов (целиком) элемент надо переместить в правый верхний угол подмассива. Проверить получилась ли на побочной диагонали убывающая последовательность элементов.

19. Задана строка из N^2 цифр. Установить можно ли, разбив строку на подстроки длиной N , записать их в строки двумерного массива $N \times N$ по одной цифре в одном элементе так, чтобы они в первом столбце расположились в порядке возрастания.
20. Найти минимальный из неповторяющихся элементов двумерного массива.
21. Найти максимальный из повторяющихся элементов двумерного массива.
22. В двумерном массиве найти среднее арифметическое первого столбца и количество элементов в каждом из следующих столбцов, превышающих среднее арифметическое предыдущего столбца.
23. Задан одномерный массив состоящий из N целых чисел. Сформировать на его основе двумерный массив $N \times N$ так, чтобы сумма элементов в первом столбце была равна первому элементу одномерного массива, сумма элементов во втором столбце была равна второму элементу одномерного массива и т. д. Нули не использовать.
24. Определить сколько элементов двумерного массива больше любого элемента на главной диагонали.
25. Из двумерного массива в одномерный записали сначала строки в произвольном порядке, затем столбцы в произвольном порядке. Написать программу восстанавливающую исходный двумерный массив по одномерному, если известна размерность двумерного массива и элементы в нем не повторяются.

4. Содержание отчета

1. Постановка задачи.
2. Вариант задания
3. Текст программы.
4. Результат решения конкретного варианта.

Лабораторная работа № 6 **"Строки"**

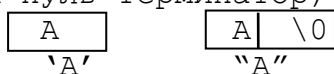
Цель: Изучение символьных и строковых переменных и способов их обработки в языке Си.

1. Краткие теоретические сведения

Для представления символьной (текстовой) информации можно использовать символы, символьные переменные и символьные константы.

Символьная константа представляется последовательностью символов, заключенной в кавычки:

"Начало строки \n". В Си нет отдельного типа для строк. Массив символов – это и есть строка. Количество элементов в таком массиве на один элемент больше, чем изображение строки, т. к. в конец строки добавлен '\0' (нулевой байт или ноль-терминатор).



символ (1 байт) строка (2 байта)

Присвоить значение массиву символов с помощью обычного оператора присваивания нельзя. Поместить строку в массив можно либо при вводе, либо с помощью инициализации:

```
char s[] = "ABCDEF";
```

Для работы со строками существует специальная библиотека string.h. Примеры функций для работы со строками из библиотеки string.h:

Функция	Прототип и краткое описание функции
strcmp	int strcmp(const char *str1, const char *str2); Сравнивает строки str1 и str2. Если str1 < str2, то результат отрицательный, если str1 = str2, то результат равен 0, если str1 > str2, то результат положительный.
strcpy	char* strcpy(char*s1, const char *s2); Копирует байты из строки s1 в строку s2
strdup	char *strdup (const char *str); Выделяет память и переносит в нее копию строки str.
strlen	unsigned strlen (const char *str); Вычисляет длину строки str.
strncat	char *strncat(char *s1, const char *s2, int kol); Приписывает kol символов строки s1 к строке s2.
strncpy	char *strncpy(char *s1, const char *s2, int kol); Копирует kol символов строки s1 в строку s2.
strnset	char *strnset(char *str, int c, int kol); Заменяет первые kol символов строки s1 символом c.

Строки, при передаче в функцию, в качестве фактических параметров могут быть определены либо как одномерные массивы типа char[], либо как указатели типа char*. В отличие от обычных массивов в этом случае нет необходимости явно указывать длину строки.

2. Постановка задачи

Задана строка, состоящая из символов. Символы объединяются в слова. Слова друг от друга отделяются одним или несколькими пробелами. В конце текста ставится

точка. Текст содержит не более 255 символов. Выполнить ввод строки, используя функцию Gets(s) и обработку строки в соответствии со своим вариантом.

3. Варианты

1. Проверить является ли строка палиндромом. (Палиндром – это выражение, которое читается одинаково слева направо и справа налево) .
2. Напечатать самое длинное и самое короткое слово в этой строке.
3. Напечатать все слова, которые не содержат гласных букв.
4. Напечатать все слова, которые содержат по одной цифре.
5. Напечатать все слова, которые совпадают с ее первым словом.
6. Преобразовать строку таким образом, чтобы сначала в ней были напечатаны только буквы, а потом только цифры, не меняя порядка следования символов в строке.
7. Преобразовать строку так, чтобы все буквы в ней были отсортированы по возрастанию.
8. Преобразовать строку так, чтобы все цифры в ней были отсортированы по убыванию.
9. Преобразовать строку так, чтобы все слова в ней стали идентификаторами, слова состоящие только из цифр – удалить.
10. Напечатать все слова-палиндромы, которые есть в этой строке (см 1 вариант) .
11. Преобразовать строку таким образом, чтобы в ее начале были записаны слова, содержащие только цифры, потом слова, содержащие только буквы, а затем слова, которые содержат и буквы и цифры.
12. Преобразовать строку таким образом, чтобы все слова в ней были напечатаны наоборот.
13. Преобразовать строку таким образом, чтобы буквы каждого слова в ней были отсортированы по возрастанию.
14. Преобразовать строку таким образом, чтобы цифры каждого слова в ней были отсортированы по убыванию.
15. Преобразовать строку таким образом, чтобы в ней остались только слова, содержащие буквы и цифры, остальные слова удалить.
16. Определить какое слово встречается в строке чаще всего.
17. Определить какие слова встречаются в строке по одному разу.
18. Все слова строки, которые начинаются с буквы, отсортировать в алфавитном порядке.
19. Все слова строки, которые начинаются с цифры отсортировать по убыванию.

20. Удалить из строки все слова, которые не являются идентификаторами.
21. В. В. Подбельский, С. С. Фомин, «Программирование на языке СИ» стр. 507, Вариант 5.
22. В. В. Подбельский, С. С. Фомин, «Программирование на языке СИ» стр. 514, Вариант 20.
23. В. В. Подбельский, С. С. Фомин, «Программирование на языке СИ» стр. 515, Вариант 21.
24. В. В. Подбельский, С. С. Фомин, «Программирование на языке СИ» стр. 516, Вариант 23.
25. В. В. Подбельский, С. С. Фомин, «Программирование на языке СИ» стр. 518, Вариант 27.

4. Содержание отчета

1. Постановка задачи для конкретного варианта.
2. Исходные данные.
3. Текст программы.
4. Результаты выполнения программы.

Лабораторная работа № 7

7.1. "Перегрузка функций в Си++"

Цель: Знакомство с организацией перегруженных функций в Си++.

1. Краткие теоретические сведения

Цель перегрузки состоит в том, чтобы функция с одним именем по разному выполнялась и возвращала разные значения при обращении к ней с различными типами и различным числом фактических параметров. Для обеспечения перегрузки функций необходимо для каждого имени функции определить сколько различных функций с ним связано.

Пример:

```
#include <iostream.h>
int max_element ( int n, int a[ ])
// находит максимальный элемент для массива типа int
{
    int max=a[0];
    for ( i=1; i<n; i++)
        if (a[i]>max) max=a[i];
    return max;
}

long max_element ( int n, long a[ ])
// находит максимальный элемент для массива типа long
{
    long max=a[0];
    for ( i=1; i<n; i++)
        if (a[i]>max) max=a[i];
    return max;
}
```

```

}
double max_element ( int n, double a[ ])
// находит максимальный элемент для массива типа double
{
double max=a[0];
for ( i=1; i<n; i++)
if (a[i]>max) max=a[i];
return max;
}

float max_element ( int n, float a[ ])
// находит максимальный элемент для массива типа float
{
float max=a[0];
for ( i=1; i<n; i++)
if (a[i]>max) max=a[i];
return max;
}

void main ( )
{
int x[]={10, 20, 30, 40, 50, 60};
long y[]={12L, 44L, 22L, 37L,30L};
. . . . .
int m1=max_element(6, x );
long m2=max_element(5, y);
. . . . .
}

```

2. Постановка задачи

Написать перегруженные функции и основную программу, которая их вызывает.

3. Варианты

1.
 - а) для сложения целых чисел;
 - б) для сложения комплексных чисел.
2.
 - а) для сложения вещественных чисел;
 - б) для сложения комплексных чисел.
3.
 - а) для умножения целых чисел;
 - б) для умножения комплексных чисел.
4.
 - а) для вычитания целых чисел;
 - б) для вычитания комплексных чисел.
5.
 - а) для умножения вещественных чисел;
 - б) для умножения комплексных чисел.
- 6.

- а) для вычитания вещественных чисел;
 - б) для вычитания комплексных чисел.
- 7.
- а) для деления целых чисел;
 - б) для деления комплексных чисел.
- 8.
- а) по номеру года выдает его название по старояпонскому календарю;
 - б) по названию месяца выдает знак Зодиака.
- 9.
- а) для сложения десятичных дробей;
 - б) для сложения обыкновенных дробей.
- 10.
- а) для вычитания десятичных дробей;
 - б) для вычитания обыкновенных дробей.
- 11.
- а) для умножения десятичных дробей;
 - б) для умножения обыкновенных дробей.
- 12.
- а) для деления десятичных дробей;
 - б) для деления обыкновенных дробей.
- 13.
- а) для преобразования десятичной дроби в обыкновенную;
 - б) для преобразования обыкновенной дроби в десятичную.
- 14.
- а) для вычисления натурального логарифма;
 - б) для вычисления десятичного логарифма.
- 15.
- а) целые числа возводит в степень n ;
 - б) из десятичных чисел извлекает корень степени n .
- 16.
- а) для перевода часов и минут в минуты;
 - б) для перевода минут в часы и минуты.
- 17.
- а) для массива целых чисел находит среднее арифметическое;
 - б) для строки находит количество букв, содержащихся в ней.
- 18.
- а) для массива целых чисел находит максимальный элемент;
 - б) для строки находит длину самого длинного слова .
- 19.
- а) для массива целых чисел находит минимальный элемент;
 - б) для строки находит длину самого короткого слова .
- 20.
- а) для массива целых чисел находит количество четных элементов;
 - б) для строки находит количество слов, начинающихся на букву «а» .
- 21.

- а) для массива целых чисел находит количество отрицательных элементов;
- б) для строки находит количество слов, заканчивающихся и начинающихся на одну и ту же букву.

22.

- а) для массива целых чисел находит количество нечетных элементов;
- б) для строки находит количество слов в ней.

23.

- а) для массива начинающегося на четное число выполняет циклический сдвиг влево на количество элементов равное первому элементу массива.
- б) для массива начинающегося на нечетное число выполняет циклический сдвиг вправо на количество элементов равное последнему элементу массива.

24.

- а) для массива целых чисел удаляет все четные элементы из массива;
- б) для строки удаляет все четные слова.

25.

- а) для двумерного массива удаляет все четные строки;
- б) для одномерного массива удаляет все элементы, заключенные между двумя нулевыми элементами.

4. Содержание отчета

1. Постановка задачи для конкретного варианта.
2. Исходные данные.
3. Текст программы.
4. Результаты выполнения программы.

7.2. "Функции с переменным числом параметров"

Цель: Знакомство с организацией функций с переменным числом параметров.

1. Краткие теоретические сведения

В Си допустимы функции, у которых при компиляции не фиксируется число параметров, кроме того, может быть неизвестен и тип параметров. Количество и тип параметров становится известным только в момент вызова, когда явно задан список фактических параметров. Каждая функция с переменным числом параметров должна иметь хотя бы один обязательный параметр.

Определение функции с переменным числом параметров:

<тип><имя>(<явные параметры>, . . .)

После списка явных параметров ставится запятая, а затем многоточие, которое показывает, что дальнейший контроль соответствия количества и типов параметров при обработке вызова функции производить не нужно. Сложность заключается в определении начала и конца переменного

списка параметров, поэтому каждая функция должна иметь механизм определения их количества и типов. Существует два подхода:

- 1) известен признак конца списка переменных параметров;
- 2) известно количество параметров, которое передается как обязательный параметр.

Пример:

```
#include <iostream.h>
int sum (int k, . . .)
{
    int *p = &k; //настроили указатель на параметр k
    int s=0;
    for ( ; k!=0;k--)
        s+=*(++p);
    return s;
}
void main( )
{
    cout<<"\nСумма(2,4,6)= "<<sum(2,4,6); //находит сумму 4+6
    cout<<"\nСумма(4,1,2,3,4)= "<<sum(4,1,2,3,4); //находит
    сумму 1+2+3+4
}
```

Для доступа к списку параметров используется указатель *p типа int. Он устанавливается на начало списка параметров в памяти, а затем p перемещается по адресам фактических параметров (++p).

2. Постановка задачи

Решить указанную в варианте задачу, используя функции с переменным числом параметров.

3. Варианты

1. Написать функцию sum с переменным числом параметров, которая находит сумму чисел типа int. Написать вызывающую функцию main, которая обращается к функции sum не менее трех раз с количеством параметров 3, 7, 11.
2. Написать функцию mult с переменным числом параметров, которая находит произведение чисел типа float. Написать вызывающую функцию main, которая обращается к функции mult не менее трех раз с количеством параметров 3, 7, 11.
3. Написать функцию sum с переменным числом параметров, которая находит сумму чисел типа int по формуле:

$$S=a_1*a_2+a_2*a_3+a_3*a_4+.$$
 Написать вызывающую функцию main, которая обращается к функции sum не менее трех раз с количеством параметров 5, 10, 12.
4. Написать функцию sum с переменным числом параметров, которая находит сумму чисел типа int по формуле:

$$S=a_1*a_2+a_3*a_4+a_5*a_6+.$$

Написать вызывающую функцию `main`, которая обращается к функции `sum` не менее трех раз с количеством параметров 8, 10, 12.

5. Написать функцию `sum` с переменным числом параметров, которая находит сумму чисел типа `int` по формуле:
 $S = a_1 \cdot a_2 - a_2 \cdot a_3 + a_3 \cdot a_4 - \dots$

Написать вызывающую функцию `main`, которая обращается к функции `sum` не менее трех раз с количеством параметров 5, 10, 12.

6. Написать функцию `min` с переменным числом параметров, которая находит минимальное из чисел типа `int`. Написать вызывающую функцию `main`, которая обращается к функции `min` не менее трех раз с количеством параметров 5, 10, 12.

7. Написать функцию `min` с переменным числом параметров, которая находит минимальное из чисел типа `int` или из чисел типа `double`, тип параметров определяется с помощью первого параметра функции. Написать вызывающую функцию `main`, которая обращается к функции `min` не менее трех раз с количеством параметров 5, 10, 12.

8. Написать функцию `max` с переменным числом параметров, которая находит минимальное из чисел типа `int`. Написать вызывающую функцию `main`, которая обращается к функции `min` не менее трех раз с количеством параметров 5, 10, 12.

9. Написать функцию `max` с переменным числом параметров, которая находит минимальное из чисел типа `int` или из чисел типа `double`, тип параметров определяется с помощью первого параметра функции. Написать вызывающую функцию `main`, которая обращается к функции `min` не менее трех раз с количеством параметров 5, 10, 12.

10. Написать функцию `kvadr` с переменным числом параметров, которая определяет количество чисел, являющихся точными квадратами (2, 4, 9, 16, ...) типа `int`. Написать вызывающую функцию `main`, которая обращается к функции `kvadr` не менее трех раз с количеством параметров 3, 7, 11.

11. Написать функцию `sum` с переменным числом параметров, которая находит сумму заданных обыкновенных дробей. Написать вызывающую функцию `main`, которая обращается к функции `sum` не менее трех раз с количеством параметров 5, 10, 12.

12. Написать функцию с переменным числом параметров для перевода чисел из десятичной системы счисления в восьмеричную. Написать вызывающую функцию `main`, которая обращается к этой функции не менее трех раз с количеством параметров 3, 5, 8.

13. Написать функцию с переменным числом параметров для перевода чисел из десятичной системы счисления в троичную. Написать вызывающую функцию `main`, которая

обращается к этой функции не менее трех раз с количеством параметров 3, 4, 7.

14. Написать функцию с переменным числом параметров для перевода чисел из двоичной системы счисления в троичную. Написать вызывающую функцию `main`, которая обращается к этой функции не менее трех раз с количеством параметров 3, 6, 7.
15. Написать функцию с переменным числом параметров для перевода чисел из восьмеричной системы счисления в десятичную. Написать вызывающую
16. Написать функцию `days` с переменным числом параметров, которая находит количество дней, прошедших между двумя датами (параметрами функции являются даты в формате «дд.мм.гг»). Написать вызывающую функцию `main`, которая обращается к функции `days` не менее трех раз с количеством параметров 3, 5, 8.
17. Написать функцию `prost` с переменным числом параметров, которая находит все простые числа из нескольких интервалов. Интервалы задаются границами `A` и `B`. Написать вызывающую функцию `main`, которая обращается к функции `prost` не менее трех раз с количеством параметров 3, 5, 6.
18. Написать функцию `nok` с переменным числом параметров, которая находит наименьшее общее кратное для нескольких чисел.

$$\text{НОК}(a,b) = \frac{a \cdot b}{\text{НОД}(a,b)} \quad (\text{НОД} - \text{наибольший общий делитель})$$

Написать вызывающую функцию `main`, которая обращается к функции `nok` не менее трех раз с количеством параметров 3, 5, 6.

19. Написать функцию (или макроопределение), которая определяет принадлежит ли точка с координатами (`x`, `y`) окружности с заданным радиусом `R`. Написать функцию `belong` с переменным числом параметров, которая определяет сколько точек с координатами (`x`, `y`) принадлежат заданной окружности. Написать вызывающую функцию `main`, которая обращается к функции `belong` не менее трех раз с количеством параметров 3, 9, 11.
20. Написать функцию (или макроопределение), которая определяет можно ли из чисел `x`, `y`, `z` построить треугольник. Написать функцию `triangle` с переменным числом параметров, которая определяет сколько троек рядом расположенных чисел типа `int` могут быть длинами сторон треугольника. Написать вызывающую функцию `main`, которая обращается к функции `triangle` не менее трех раз с количеством параметров 3, 9, 11.
21. Написать функцию (или макроопределение), которая находит угол треугольника по его сторонам. Написать

- функцию `angles` с переменным числом параметров, которая находит углы n -угольника по заданным сторонам. Написать вызывающую функцию `main`, которая обращается к функции `angle` не менее трех раз с количеством параметров 3, 9, 11.
22. Написать функцию (или макроопределение), которая находит площадь треугольника по его сторонам. Написать функцию `square` с переменным числом параметров, которая находит площадь n -угольника по заданным сторонам. Написать вызывающую функцию `main`, которая обращается к функции `square` не менее трех раз с количеством параметров 3, 5, 8.
23. Написать функцию (или макроопределение), которая находит длину стороны по координатам его точек.. Написать функцию `belong`, которая определяет принадлежит ли точка M с координатами (x, y) треугольнику, заданному координатами вершин. Написать функцию с переменным числом параметров, которая определяет принадлежит ли точка M выпуклому многоугольнику, заданному координатами своих вершин.
24. Написать функцию (или макроопределение), которая находит длину стороны по координатам его точек.. Написать функцию `square`, которая вычисляет площадь треугольника, заданного координатами вершин. Написать функцию `squaren` с переменным числом параметров, которая определяет площадь выпуклого многоугольника, заданного координатами своих вершин.
25. Написать функцию (или макроопределение), которая находит длину стороны по координатам его точек.. Написать функцию `square`, которая вычисляет площадь треугольника, заданного координатами вершин. Написать функцию `square1` с переменным числом параметров, которая определяет площадь треугольника, содержащего диагональ наибольшей длины выпуклого многоугольника, заданного координатами своих вершин.

4. Содержание отчета

1. Постановка задачи для конкретного варианта.
2. Исходные данные.
3. Текст программы.
4. Результаты выполнения программы.

Лабораторная работа №8 "Блоковый ввод-вывод"

Цель: Работа с двоичными файлами, организация ввода-вывода структурированной информации и ее хранение на внешних носителях.

1. Краткие теоретические сведения

1. 1. Ввод и вывод в Си

Особенностью Си является отсутствие в этом языке структурированных файлов. Все файлы рассматриваются как не структурированная последовательность байтов. При таком подходе понятие файла распространяется и на различные устройства.

В Си отсутствуют средства ввода-вывода. Все операции ввода-вывода реализуются с помощью функций, которые находятся в библиотеке Си. Библиотека Си поддерживает три уровня ввода-вывода:

- потоковый ввод-вывод;
- ввод-вывод нижнего уровня;
- ввод-вывод для консоли и портов (зависит от ОС).

1.2. Потоковый ввод-вывод

На уровне потокового ввода-вывода обмен данными производится побайтно, т. е. за одно обращение к устройству (файлу) производится считывание или запись фиксированной порции данных (512 или 1024 байта). При вводе с диска или при считывании из файла данные помещаются в буфер ОС, затем побайтно или порциями передаются в программе пользователя. При выводе в файл данные накапливаются в буфере, а при заполнении буфера записываются в виде единого блока на диск. Буферы ОС реализуются в виде участков основной памяти. Функции библиотеки Си, поддерживающие обмен, с данными на уровне потока позволяют обрабатывать данные различных размеров и форматов.

Поток – это файл вместе с предоставленными средствами буферизации. При работе с потоком можно:

- 1) Открывать и закрывать потоки (связывать указатели на поток с конкретными файлами);
- 2) вводит и выводит строку, символ, форматированные данные, порцию данных произвольной длины;
- 3) анализировать ошибки ввода-вывода и достижения конца файла;
- 4) управлять буферизацией потока и размером буфера;
- 5) получать и устанавливать указатель текущей позиции в файле;

Функции библиотеки ввода-вывода находятся в заголовочном файле `<stdio.h>`.

1.3. Открытие и закрытие потока

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

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

```
.....
```

```
FILE *fp;
```

```
.....
```

```
fp= fopen( "t.txt", "r");
```

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

Существуют следующие режимы для открытия файла:

"w" - открыть файл для записи, если файл существует, то он стирается;

"r" - открыть файл для чтения;

"a" - открыть файл для добавления, если файл существует, то он не стирается и можно писать в конец файла;

"w+" - открыть файл для записи и исправления, если файл существует, то он стирается, а далее можно и читать, и писать, размеры файла можно увеличивать;

"r+" - открыть файл для чтения и записи, но увеличить размер файла нельзя;

"a+" - открыть файл для добавления, т. е. можно и читать и писать, в том числе и в конец файла.

Поток можно открыть в текстовом (t) или двоичном (b) режиме. По умолчанию - текстовый режим. В явном виде режим указывается следующим образом: "r+b"или "rb" - двоичный (бинарный) режим.

Пример:

```
if ((fp=fopen("t.txt", "w")==NULL)
```

```
{
```

```
perror("\ношибка при открытии файла"); // выводит строку  
символов с сообщением // об
```

```
ошибке
```

```
exit(0);
```

```
}
```

После работы с файлом, его надо закрыть

```
fclose(<указатель_на_поток>);
```

1.4. Блочный ввод-вывод

Для блочного ввода и вывода используются функции :

1) int fread(void *ptr, int size, int n, FILE *fp) , где void *ptr - указатель на область памяти, в которой размещаются считываемые из файла данные;

int size - размер одного считываемого элемента;

int n - количество считываемых элементов;

FILE *fp - указатель на файл, из которого производится считывание.

В случае успешного считывания информации функция возвращает число прочитанных элементов (а не байтов), иначе возвращает EOF.

2) int fwrite(void *ptr, int size, int n, FILE *fp) ,

где

void *ptr - указатель на область памяти, в которой размещаются записываемые в файл данные;
 int size - размер одного записываемого элемента;
 int n - количество записываемых элементов;
 FILE *fp - указатель на файл, в который производится запись.

В случае успешной записи информации функция возвращает число записанных элементов, иначе возвращает EOF.

Пример:

```
. . . . .
typedef STRUCT
{
char name [40];
char post [40];
float rate;
}EMPLOYEE;
void main ()
{
FILE *f; // указатель связанный с файлом
EMPLOYEE e; // переменная
EMPLOYEE mas[10] //массив
//открываем файл
if ((f=fopen("f.dat", "wb")==NULL) exit(1); // если при
открытии файла возникает
//ошибка, то выходим из
функции

int i;
for(i=1; i<=10;i++)
{
//формируем запись e
printf("name="); scanf("%s",&e.name);
printf("post="); scanf("%s",&e.post);
printf("rate="); scanf("%f",e.rate);
// записываем запись e в файл
fwrite(&e, sizeof(EMPLOYEE),1,f);
if (ferror(f)==NULL) exit(2);
}
fclose(f);
//чтение записей из файла
if ((f=fopen("f.dat", "rb")==NULL) exit(3); // если при
открытии файла возникает
//ошибка, то выходим из
функции

i=0;
while(!feof(f)&&i<=10)
{
fread(&mas[i], sizeof(EMPLOYEE),1,f);
i++;
}
fclose(f);
}
```

2. Постановка задачи

Сформировать двоичный файл из элементов, заданной в варианте структуры, распечатать его содержимое, выполнить удаление и добавление элементов в соответствии со своим вариантом, используя для поиска удаляемых или добавляемых элементов функцию. Формирование, печать, добавление и удаление элементов оформить в виде функций. Предусмотреть сообщения об ошибках при открытии файла и выполнении операций ввода/вывода.

3. Варианты

1. Структура "Абитуриент":

- фамилия, имя, отчество;
- год рождения;
- оценки вступительных экзаменов (3);
- средний балл аттестата.

Удалить элемент с указанным номером, добавить элемент после элемента с указанной фамилией.

2. Структура "Сотрудник":

- фамилия, имя, отчество;
- должность
- год рождения;
- заработная плата.

Удалить элемент с указанной фамилией, добавить элемент после элемента с указанным номером.

3. Структура "Государство":

- название;
- столица;
- численность населения;
- занимаемая площадь.

Удалить все элементы, у которых численность меньше заданной, добавить элемент после элемента с указанным номером.

4. Структура "Человек":

- фамилия, имя, отчество;
- домашний адрес;
- номер телефона;
- возраст.

Удалить все элементы с заданным возрастом, добавить элемент после элемента с заданным номером.

5. Структура "Человек":

- фамилия, имя, отчество;
- год рождения;
- рост;
- вес.

Удалить все элемент с указанным ростом и весом,
добавить элемент после элемента с указанной фамилией.

6. Структура "Школьник":

- фамилия, имя, отчество;
- класс;
- номер телефона;
- оценки по предметам (математика, физика, русский язык, литература).

Удалить все элементы, у которых есть 2 хотя бы по одному предмету, добавить элемент в начало файла.

7. Структура "Студент":

- фамилия, имя, отчество;
- домашний адрес;
- группа;
- рейтинг.

Удалить все элементы, у которых рейтинг меньше заданного, добавить 1 элемент в конец файла.

8. Структура "Покупатель":

- фамилия, имя, отчество;
- домашний адрес;
- номер телефона;
- номер кредитной карточки.

Удалить 3 элемента из начала файла, добавить 3 элемента в конец файла.

9. Структура "Пациент":

- фамилия, имя, отчество;
- домашний адрес;
- номер медицинской карты;
- номер страхового полиса.

Удалить элемент с заданным номером медицинской карты, добавить 2 элемента в начало файла.

10. Структура "Информация":

- носитель;
- объем;
- название;
- автор.

Удалить первый элемент с заданным объемом информации, добавить элемент перед элементом с указанным номером.

11. Структура "Видеокассета":

- название фильма;
- режиссер;
- продолжительность;
- цена.

Удалить все элементы с ценой выше заданной, добавить 3 элемента в конец файла.

12. Структура "Музыкальный диск":

- название;
- автор;
- продолжительность;
- цена.

Удалить первый элемент с заданной продолжительностью, добавить 2 элемента после элемента с заданным номером.

13. Структура "Спортивная команда":

- название;
- город;
- количество игроков;
- количество набранных очков.

Удалить все элементы с количеством очков меньше заданного, добавить 2 элемента в начало файла.

14. Структура "Стадион":

- название;
- адрес;
- вместимость;
- виды спорта.

Удалить элемент с заданным названием, добавить 2 элемента после элемента с указанным номером.

15. Структура "Автомобиль":

- марка;
- год выпуска;
- цена;
- цвет.

Удалить все элементы, у которых год выпуска меньше заданного, добавить элемент в начало файла.

16. Структура "Владелец автомобиля":

- фамилия, имя, отчество;
- номер автомобиля;
- телефон;
- номер техпаспорта.

Удалить элемент с заданным номером, добавить 2 элемента перед элементом с заданной фамилией.

17. Структура "Фильм":

- название;
- режиссер;
- год выпуска;
- стоимость.

Удалить все элементы, у которых стоимость превышает заданную, добавить элемент в начало файла.

18. Структура "Книга":

- название;
- автор;
- год издания;
- количество страниц.

Удалить 3 элемента из начала файла, добавить элемент перед элементом с указанным названием.

19. Структура "Фильм":

- название;
- режиссер;
- страна;
- приносимая прибыль.

Удалить 2 элемента из конца файла, добавить элемент после элемента с указанным названием.

20. Структура "Государство":

- название;
- государственный язык;
- денежная единица;
- курс валюты относительно \$.

Удалить элемент с указанным названием, добавить 2 элемента в конец файла.

21. Структура "Автомобиль":

- марка;
- серийный номер;
- регистрационный номер;
- год выпуска.

Удалить 3 элемента из начала файла, добавить элемент поле элемента с указанным регистрационным номером.

22. Структура "Владелец автомобиля":

- фамилия, имя, отчество;
- номер автомобиля;
- номер техпаспорта;
- отделение регистрации ГАИ.

Удалить элемент с заданным номером, добавить 2 элемента перед элементом с заданной фамилией.

23. Структура "Стадион":

- название;
- год постройки;
- количество площадок;
- виды спорта.

Удалить все элементы, у которых год постройки меньше заданного, добавить 2 элемента перед элементом с указанным номером.

24. Структура "Студент":

- фамилия, имя, отчество;
- номер телефона;
- группа;
- оценки по 3 основным предметам.

Удалить все элементы из группы с указанным номером, у которых среднее арифметическое оценок меньше заданного, добавить элемент после элемента с заданной фамилией.

25. Структура "Студент":

- фамилия, имя, отчество;
- дата рождения;
- домашний адрес;
- рейтинг.

Удалить элементы, у которых даты рождения совпадают, добавить элемент перед элементом с заданной фамилией.

4. Методические указания

1. Для заполнения файла можно использовать функцию, формирующую одну структуру, указанного в варианте типа. Значения элементов структуры вводятся с клавиатуры. Для ввода можно использовать операцию >> и функцию gets().
2. При вводе структур можно реализовать один из следующих механизмов:
 - ввод заранее выбранного количества структур (не менее 5);
 - ввод до появления структуры с заданным количеством признаков;
 - диалог с пользователем о необходимости продолжать ввод.
3. Для записи структуры в файл и чтения структуры из файла использовать функции блочного ввода/вывода fread и fwrite.
4. Для удаления/добавления элементов в файл использовать вспомогательный файл.

5. Содержание отчета

1. Постановка задачи.
2. Описание используемых типов данных.
3. Текст функций для:
 - формирования файла,
 - печати файла,
 - добавления записи в файл,
 - удаления записи из файла
 - поиска структуры для удаления.
4. Результат решения конкретного варианта.

Лабораторная работа № 9

"Строковый ввод-вывод"

Цель: Работа с текстовыми файлами, ввод-вывод текстовой информации и ее хранение на внешних носителях.

1. Краткие теоретические сведения

Для построчного ввода – вывода используются следующие функции;

1) `char *fgets(char *s, int n, FILE *F)`, где
`char *s` – адрес, по которому размещаются считанные байты;
`int n` – количество считываемых байтов;
`FILE *fr` – указатель на файл, из которого производится считывание.

Прием символов заканчивается после передачи `n` байтов или при получении `"\n"`. Управляющий символ `"\n"` тоже передается в принимающую строку. В любом случае строка заканчивается `"\0"`. При успешном завершении считывания, функция возвращает указатель на прочитанную строку, иначе возвращает `NULL`.

2) `char *fputs(char *s, FILE *F)`, где
`char *s` – адрес, из которого берутся записываемые в файл байты;
`FILE *fr` – указатель на файл, в который производится запись.

Пример:

```
int MAXLINE=255; //максимальная длина строки
FILE      *in,   //исходный файл
          *out;   //принимающий файл
char buf[MAXLINE]; //строка, с помощью которой
                    //выполняется копирование
//копирование строк одного файла в другой
while (fgets (buf, MAXLINE, in)!=NULL)
    fputs (buf,out);
```

2. Постановка задачи

1. Создать текстовый файл F1 не менее, чем из 10 строк и записать в него информацию
2. Выполнить задание.

3. Варианты

1.
 - 1) Скопировать в файл F2 только четные строки из F1.
 - 2) Подсчитать размер файлов F1 и F2 (в байтах).
2.
 - 1) Скопировать в файл F2 только те строки из F1, которые начинаются с буквы «А».
 - 2) Подсчитать количество слов в F2.

3.

- 1) Скопировать в файл F2 только те строки из F1, которые начинаются и заканчиваются на одну и ту же букву.
- 2) Подсчитать количество символов в F2.

4.

- 1) Скопировать из файла F1 в файл F2 строки, начиная с 4.
- 2) Подсчитать количество символов в последнем слове F2.

5.

- 1) Скопировать из файла F1 в файл F2 строки, начиная с K до K+5.
- 2) Подсчитать количество гласных букв в файле F2.

6.

- 1) Скопировать из файла F1 в файл F2 строки, начиная с N до K.
- 2) Подсчитать количество согласных букв в файле F2.

7.

- 1) Скопировать из файла F1 в файл F2 все строки, кроме тех, что начинаются на букву А.
- 2) Подсчитать количество символов в первом слове F2.

8.

- 1) Скопировать из файла F1 в файл F2 все строки, которые не содержат цифры.
- 2) Подсчитать количество строк, которые начинаются на букву «А» в файле F2.

9

- 1) Скопировать из файла F1 в файл F2 все строки, которые содержат только одно слово.
- 2) Найти самое длинное слово в файле F2.

10.

- 1) Скопировать из файла F1 в файл F2 все строки, которые не содержат слова, начинающиеся на одну букву.
- 2) Найти самое короткое слово в файле F2.

11.

- 1) Скопировать из файла F1 в файл F2 все строки, кроме той строки, которая содержит самое короткое слово.
- 2) Напечатать номер этой строки.

12.

- 1) Скопировать из файла F1 в файл F2 все строки, кроме той строки, в которой больше всего гласных букв.
- 2) Напечатать номер этой строки.

13.

- 1) Скопировать из файла F1 в файл F2 все строки, начинающиеся на букву «А» и расположенные между строками с номерами N1 и N2.
- 2) Определить номер той строки, в которой больше всего согласных букв, файла F2.

14.

- 1) Скопировать из файла F1 в файл F2 все строки, не содержащие букву «А» и расположенные между строками с номерами N1 и N2.
- 2) Определить номер той строки, в которой больше всего гласных букв, файла F2.

15.

- 1) Скопировать из файла F1 в файл F2 все строки, заканчивающиеся на букву «А» и расположенные между строками с номерами N1 и N2.
- 2) Определить номер той строки, в которой больше всего букв «А», файла F2.

16.

- 1) Скопировать из файла F1 в файл F2 все строки, начинающиеся на букву «А» и заканчивающиеся на букву «С», расположенные между строками с номерами N1 и N2.
- 2) Определить количество слов в первой строке файла F2.

17.

- 1) Скопировать из файла F1 в файл F2 все строки, начинающиеся на букву «А» расположенные между строками с номерами N1 и N2, а затем все строки от N2+3 и до последней.
- 2) Определить количество слов в последней строке файла F2.

18.

- 1) Скопировать из файла F1 в файл F2 все строки, в которых нет одинаковых слов.
- 2) Определить количество гласных букв в первой строке файла F2.

19.

- 1) Скопировать из файла F1 в файл F2 все строки, в которых нет слов, совпадающих с первым словом.

- 2) Определить количество согласных букв в первой строке файла F2.

20.

- 1) Скопировать из файла F1 в файл F2 все строки, в которых есть одинаковые слова.
- 2) Определить количество гласных букв в последней строке файла F2.

21.

- 1) Скопировать из файла F1 в файл F2 все строки, в которых есть слова, совпадающие с первым словом.
- 2) Определить количество согласных букв в последней строке файла F2.

22.

- 1) Скопировать из файла F1 в файл F2 все строки, в которых более 2 слов.
- 2) Определить номер слова, в котором больше всего гласных букв.

23.

- 1) Скопировать из файла F1 в файл F2 все строки, в которых содержится только одно слово.
- 2) Определить номер слова, в котором больше всего согласных букв.

24.

- 1) Скопировать из файла F1 в файл F2 все строки, в которых содержится два одинаковых слова.
- 2) Определить номер слова, в котором больше всего букв «А».

25.

- 1) Скопировать из файла F1 в файл F2 все строки, в которых содержится не менее двух одинаковых слов.
- 2) Определить номер слова, в котором больше всего цифр.

4. Содержание отчета

1. Постановка задачи.
2. Описание используемых типов данных.
3. Текст функций для:
 - формирования файла,
 - печати файла,
 - копирования файлов,
 - выполнения задания.
4. Результат решения конкретного варианта.

Лабораторная работа № 10

"Динамические массивы"

Цель: Организация динамических массивов.

1. Краткие теоретические сведения

При традиционном определении массива:
тип имя_массива [количество_элементов];
общее количество памяти, выделяемой под массив, задается определением и равно количество_элементов * sizeof(тип). Но иногда бывает нужно чтобы память под массив выделялась для решения конкретной задачи, причем ее размеры заранее не известны и не могут быть фиксированы.

Формирование массивов с переменными размерами можно организовать с помощью указателей и средств динамического распределения памяти двумя способами:

- 1) с использованием библиотечных функций, описанных в заголовочных файлах alloc.h и stdlib.h (стандартный Си);
- 2) с использованием операций new и delete (Си++).

1.1. Формирование динамических массивов с использованием библиотечных функций

Для выделения и освобождения динамической памяти используются функции

Функция	Прототип и краткое описание
malloc	void * malloc(unsigned s) Возвращает указатель на начало области динамической памяти длиной в s байт, при неудачном завершении возвращает NULL
calloc	void * calloc(unsigned n, unsigned m) Возвращает указатель на начало области динамической памяти для размещения n элементов длиной по m байт каждый, при неудачном завершении возвращает NULL
realloc	void * realloc(void * p, unsigned s) Изменяет размер блока ранее выделенной динамической памяти до размера s байт, p – адрес начала изменяемого блока, при неудачном завершении возвращает NULL
free	void * free(void p) Освобождает ранее выделенный участок динамической памяти, p – адрес первого байта

Пример:

Функция для формирования одномерного динамического массива

```
int * make_mas(int n)
(
int *mas;
mas=(int*)malloc(n*sizeof(int));
for(int i=0;i<n;i++)
mas[i]=random(10);
```

```
return mas;
}
```

Для выделения памяти используется функция `malloc`, параметром которой является размер выделяемого участка памяти равный `n*sizeof(int)`. Так как функция `malloc` возвращает нетипизированный указатель `void*`, то необходимо выполнить преобразование полученного нетипизированного указателя в указатель `int*`.

Освободить выделенную память можно функцией `free(mas)`.

1.2. Формирование динамических массивов с использованием операций `new` и `delete`

Для динамического распределения памяти используются операции `new` и `delete`. Операция

```
new имя_типа
```

или

```
new имя_типа инициализатор
```

позволяет выделить и сделать доступным свободный участок памяти, размеры которого соответствуют типу данных, определяемому именем типа. В выделенный участок заносится значение, определяемое инициализатором, который не является обязательным параметром. В случае успешного выделения памяти операция возвращает адрес начала выделенного участка памяти, если участок не может быть выделен, то возвращается `NULL`.

Примеры:

```
1) int *i;
   i=new int(10);
2) float *f;
   f=new float;
3) int *mas=new[5];
```

В примерах 1, 2 показано как выделить память под скалярные переменные, пример 3 показывает выделение памяти под массив переменных.

Операция `delete` указатель освобождает участок памяти ранее выделенный операцией `new`.

Пример:

Функция для формирования двумерного динамического массива

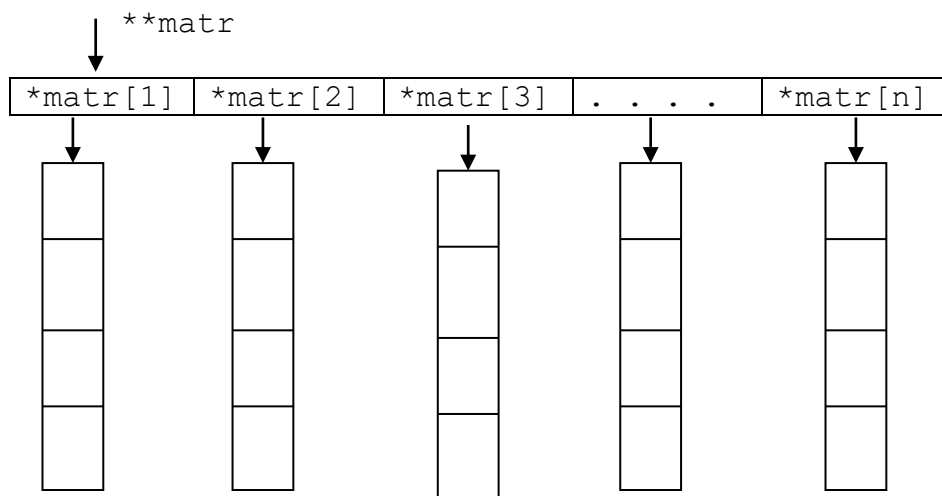
```
int ** make_matr(int n)
{
int **matr;
int i,j;
matr=new int*[n];
for (i=0;i<n;i++)
{
matr[i]=new int[n];
for (j=0;j<n;j++)
matr[i][j]=random(10);
}
return matr;
```

```

}

```

При формировании матрицы сначала выделяется память для массива указателей на одномерные массивы, а затем в цикле с параметром выделяется память под n одномерных массивов.



Чтобы освободить память необходимо выполнить цикл для освобождения одномерных массивов

```

for(int i=0;i<n;i++)
    delete matr[i];

```

После этого освобождаем память на которую указывает указатель `matr`

```

delete [] matr;

```

2. Постановка задачи

Написать программу, в которой создаются динамические массивы и выполняются их обработки в соответствии со своим вариантом.

3. Порядок выполнения работы

1. Ввести размер массива;
2. Сформировать массив с помощью операции `new` или библиотечных функций `malloc` (`calloc`);
3. Заполнить массив (можно с помощью датчика случайных чисел);
4. Выполнить задание варианта, сформировать новый массив (ы) - результат (ы);
5. Напечатать массив (ы) - результат (ы);
6. Удалить динамические массивы с помощью операции `delete` или библиотечной функции `free`.

4. Варианты заданий

1. Сформировать одномерный массив. Удалить из него элемент с заданным номером, добавить элемент с заданным номером;
2. Сформировать одномерный массив. Удалить из него элемент с заданным ключом, добавить элемент с заданным ключом;

3. Сформировать одномерный массив. Удалить из него K элементов, начиная с заданного номера, добавить элемент с заданным ключом;
4. Сформировать одномерный массив. Удалить из него элемент с заданным номером, добавить K элементов, начиная с заданного номера;
5. Сформировать одномерный массив. Удалить из него K элементов, начиная с заданного номера, добавить K элементов, начиная с заданного номера;
6. Сформировать двумерный массив. Удалить из него строку с заданным номером;
7. Сформировать двумерный массив. Удалить из него столбец с заданным номером;
8. Сформировать двумерный массив. Добавить в него строку с заданным номером;
9. Сформировать двумерный массив. Добавить в него столбец с заданным номером;
10. Сформировать двумерный массив. Удалить из него строку и столбец с заданным номером.
11. Сформировать двумерный массив. Добавить в него строку и столбец с заданным номером.
12. Сформировать двумерный массив. Удалить из него все строки, в которых встречается заданное число.
13. Сформировать двумерный массив. Удалить из него все столбцы, в которых встречается заданное число.
14. Сформировать двумерный массив. Удалить из него строку и столбец, на пересечении которых находится минимальный элемент.
15. Сформировать двумерный массив. Удалить из него строку и столбец, на пересечении которых находится максимальный элемент.
16. Сформировать массив строк. Удалить из него самую короткую строку.
17. Сформировать массив строк. Удалить из него самую длинную строку.
18. Сформировать массив строк. Удалить из него строку, начинающуюся на букву "a".
19. Сформировать массив строк. Удалить из него строку, начинающуюся и заканчивающуюся на букву "a".
20. Сформировать массив строк. Удалить из него строку, начинающуюся и заканчивающуюся на одну и ту же букву.
21. Сформировать массив строк. Удалить из него строку с заданным номером.
22. Сформировать массив строк. Удалить из него K строк, начиная со строки с заданным номером.
23. Сформировать массив строк. Удалить из него одинаковые строки. Сформировать массив строк. Удалить из него K последних строк.
24. Сформировать массив строк. Удалить из него K первых строк.

25. Сформировать массив строк. Добавить в него строку с заданным номером.

5. Содержание отчета

1. Постановка задачи.
2. Функции для формирования массива, печати массива, преобразования массива, удаления массива.
3. Результаты выполнения работы.

Лабораторная работа № 11

"Информационные динамические структуры"

Цель: Знакомство с динамическими информационными структурами на примере одно- и двунаправленных списков.

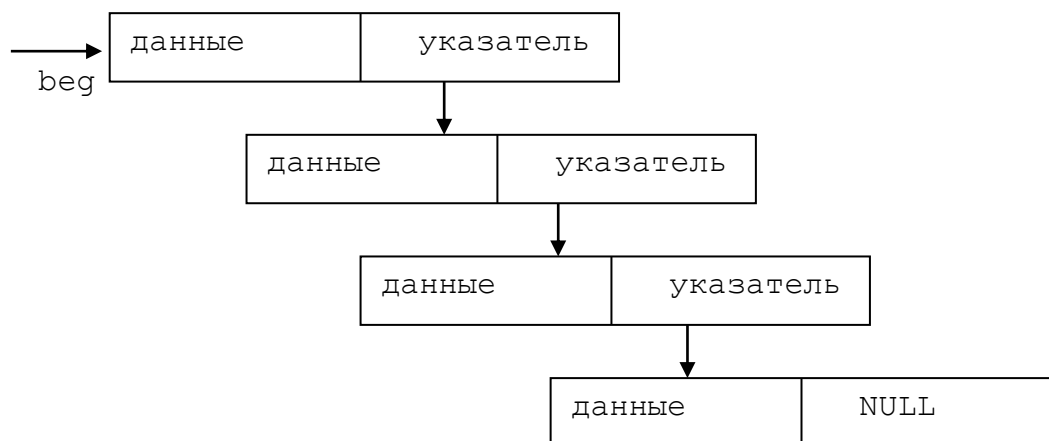
1. Краткие теоретические сведения

Во многих задачах требуется использовать данные у которых конфигурация, размеры, состав могут изменяться в процессе выполнения программы. Для их представления используют динамические информационные структуры.

Наиболее простая информационная структура – это односвязный список, элементами которого служат объекты структурного типа. Например

```
struct имя_структурного_типа
{
    элементы_структуры;
    struct имя_структурного_типа *указатель;
}
```

В каждую структуру такого типа входит указатель на объект того же типа, что и определяемая структура.



Примеры:

1. Описание структуры

```
struct point
{int key;
point* next;
};
```

Поле `key` содержит информационную часть структуры `point`, а поле `next` содержит адрес следующего элемента списка.

2. Функция для формирования однонаправленного списка

```
point* make_point( int n)
```

```
{
point *first, *p;
first=NULL;
for (int i=n;i>0;i--)
{
p=new(point);
p->key=i;
p->next=first;
first=p;
}
return first;
}
```

В качестве параметра в функцию передается количество элементов в списке, а результатом является указатель на первый элемент этого списка. Указатель `p` указывает на вновь создаваемый элемент. Для обращения к полям используется операция доступа к элементу структуры, с которой связан указатель `->`. Существует вторая возможность обращения к полю динамической структуры: `(*p).key` или `(*p).next`. В информационное поле `key` заносится порядковый номер элемента в списке. Добавление новых элементов осуществляется в начало списка.

3. Функция для печати однонаправленного списка

```
point* print_point(point*first)
```

```
{
if (first==NULL)return NULL;
point*p=first;
while(p!=NULL)
{
cout<<p->key<<" ";
p=p->next;
}
return first;
}
```

При печати сформированного списка осуществляется проход по списку с помощью вспомогательной переменной `p` до тех пор, пока она не станет равна `NULL`.

2. Постановка задачи

Написать программу, в которой создаются динамические структуры и выполняются их обработки в соответствии со своим вариантом.

Для каждого варианта разработать следующие функции:

1. Создание списка.
2. Добавление элемента в список (в соответствии со своим вариантом).

3. Удаление элемента из списка (в соответствии со своим вариантом).
4. Печать списка.
5. Запись списка в файл.
6. Уничтожение списка.
7. Восстановление списка из файла.

3. Порядок выполнения работы

1. Написать функцию для создания списка. Функция может создавать пустой список, а затем добавлять в него элементы.
2. Написать функцию для печати списка. Функция должна предусматривать вывод сообщения, если список пустой.
3. Написать функции для удаления и добавления элементов списка в соответствии со своим вариантом.
4. Выполнить изменения в списке и печать списка после каждого изменения.
5. Написать функцию для записи списка в файл.
6. Написать функцию для уничтожения списка.
7. Записать список в файл, уничтожить его и выполнить печать (при печати должно быть выдано сообщение "Список пустой").
8. Написать функцию для восстановления списка из файла.
9. Восстановить список и распечатать его.
10. Уничтожить список.

4. Варианты заданий

1. Записи в линейном списке содержат ключевое поле типа `int`. Сформировать однонаправленный список. Удалить из него элемент с заданным номером, добавить элемент с заданным номером;
2. Записи в линейном списке содержат ключевое поле типа `int`. Сформировать однонаправленный список. Удалить из него элемент с заданным ключом, добавить элемент перед элементом с заданным ключом;
3. Записи в линейном списке содержат ключевое поле типа `int`. Сформировать однонаправленный список. Удалить из него K элементов, начиная с заданного номера, добавить элемент перед элементом с заданным ключом;
4. Записи в линейном списке содержат ключевое поле типа `int`. Сформировать однонаправленный список. Удалить из него элемент с заданным номером, добавить K элементов, начиная с заданного номера;
5. Записи в линейном списке содержат ключевое поле типа `int`. Сформировать однонаправленный список. Удалить из него K элементов, начиная с заданного номера, добавить K элементов, начиная с заданного номера;
6. Записи в линейном списке содержат ключевое поле типа `int`. Сформировать двунаправленный список. Удалить из него элемент с заданным номером, добавить элемент в начало списка.

7. Сформировать двунаправленный список. Удалить из него первый элемент, добавить элемент в конец списка.
8. Записи в линейном списке содержат ключевое поле типа `int`. Сформировать двунаправленный список. Удалить из него элемент после элемента с заданным номером, добавить `K` элементов в начало списка.
9. Записи в линейном списке содержат ключевое поле типа `int`. Сформировать двунаправленный список. Удалить из него `K` элементов перед элементом с заданным номером, добавить `K` элементов в конец списка.
10. Записи в линейном списке содержат ключевое поле типа `int`. Сформировать двунаправленный список. Добавить в него элемент с заданным номером, удалить `K` элементов из конца списка.
11. Записи в линейном списке содержат ключевое поле типа `*char`(строка символов). Сформировать двунаправленный список. Удалить из него элемент с заданным ключом, добавить элемент с указанным номером.
12. Записи в линейном списке содержат ключевое поле типа `*char`(строка символов). Сформировать двунаправленный список. Удалить из него Элементы, с одинаковыми ключевыми полями. Добавить элемент после элемента с заданным ключевым полем.
13. Записи в линейном списке содержат ключевое поле типа `*char`(строка символов). Сформировать двунаправленный список. Удалить из него `K` первых элементов. Добавить элемент после элемента, начинающегося с указанного символа.
14. Записи в линейном списке содержат ключевое поле типа `*char`(строка символов). Сформировать двунаправленный список. Удалить из него `K` элементов с указанными номерами. Добавить `K` элементов с указанными номерами.
15. Записи в линейном списке содержат ключевое поле типа `*char`(строка символов). Сформировать двунаправленный список. Удалить `K` элементов из конца списка. Добавить элемент после элемента с заданным ключом.
16. Записи в линейном списке содержат ключевое поле типа `*char`(строка символов). Сформировать двунаправленный список. Удалить элемент с заданным ключом. Добавить `K` элементов в конец списка.
17. Записи в линейном списке содержат ключевое поле типа `*char`(строка символов). Сформировать двунаправленный список. Удалить элемент с заданным номером. Добавить `K` элементов в начало списка.
18. Записи в линейном списке содержат ключевое поле типа `*char`(строка символов). Сформировать двунаправленный список. Удалить элемент с заданным ключом. Добавить `K` элементов в начало списка.
19. Записи в линейном списке содержат ключевое поле типа `*char`(строка символов). Сформировать двунаправленный

- список. Удалить К элементов с заданными номерами. Добавить К элементов в начало списка.
20. Записи в линейном списке содержат ключевое поле типа *char(строка символов). Сформировать двунаправленный список. Удалить элемент с заданным ключом. Добавить по К элементов в начало и в конец списка.
 21. Записи в линейном списке содержат ключевое поле типа *char(строка символов). Сформировать двунаправленный список. Удалить элементы перед и после элемента с заданным ключом. Добавить по К элементов в начало и в конец списка.
 22. Записи в линейном списке содержат ключевое поле типа *char(строка символов). Сформировать двунаправленный список. Удалить элемент с заданным ключом. Добавить К элементов перед элементом с заданным ключом.
 23. Записи в линейном списке содержат ключевое поле типа *char(строка символов). Сформировать двунаправленный список. Удалить элемент с заданным ключом. Добавить К элементов после элемента с заданным ключом.
 24. Записи в линейном списке содержат ключевое поле типа *char(строка символов). Сформировать двунаправленный список. Удалить элемент с заданным номером. Добавить по К элементов перед и после элемента с заданным ключом.
 25. Записи в линейном списке содержат ключевое поле типа *char(строка символов). Сформировать двунаправленный список. Удалить элемент с заданным ключом. Добавить К элементов перед элементом с заданным номером.

5.Содержание отчета

4. Постановка задачи.
5. Функции для работы со списком.
6. Функция main().
7. Результаты выполнения работы.

Библиографический список

1. Викентьева О.Л., Гусин А.Н., Полякова О.А. Проектирование программ и программирование на С++. Часть I. Структурное программирование: Учеб. пособие. – Пермь:Изд-во Перм. нац. исслед. политехн. ун-та, 2012. – 139 с.
2. Павловская Т.А. С/С++. Программирование на языке высокого уровня: Учеб.пособие. – СПб.:Питер, 2007. – 461 с.
3. Павловская Т.А., Щупак Ю.А. С/С++. Программирование на языке высокого уровня. Структурное программирование: Практикум. – СПб.:Питер, 2003. – 240 с.
4. Гладков В. П. Курс лабораторных работ по программированию : Учебное пособие для специальностей

- электротехнического факультета ПГТУ/ Перм. гос. техн. ун-т.-Пермь, 1998. -153с.
5. Подбельский В. В., Фомин С. С. Программирование на языке Си: Учеб. пособие. -М.:Финансы и статистика, 1998.-600с.
 6. Подбельский В. В. Язык Си++: Учеб. пособие.-М.:Финансы и статистика, 1996.-560с.
 7. Страуструп Б. Язык программирования Си++: Пер. с англ. - М.: Радио и связь, 1991.-352с.

ОГЛАВЛЕНИЕ

ЛАБОРАТОРНАЯ РАБОТА №1. "ЗНАКОМСТВО С СИ++. ВЫПОЛНЕНИЕ ПРОГРАММЫ ПРОСТОЙ СТРУКТУРЫ"	2
ЛАБОРАТОРНАЯ РАБОТА №2. "ИСПОЛЬЗОВАНИЕ ОСНОВНЫХ ОПЕРАТОРОВ ЯЗЫКА СИ"	12
ЛАБОРАТОРНАЯ РАБОТА №3 "ВЫЧИСЛЕНИЕ ФУНКЦИЙ С ИСПОЛЬЗОВАНИЕМ ИХ РАЗЛОЖЕНИЯ В СТЕПЕННОЙ РЯД"	18
ЛАБОРАТОРНАЯ РАБОТА № 4 "РАБОТА С ОДНОМЕРНЫМИ МАССИВАМИ"	22
ЛАБОРАТОРНАЯ РАБОТА №5 "ФУНКЦИИ И МАССИВЫ"	29
ЛАБОРАТОРНАЯ РАБОТА № 6 "СТРОКИ"	34
ЛАБОРАТОРНАЯ РАБОТА № 7 7.1. "ПЕРЕГРУЗКА ФУНКЦИЙ В СИ++"	37
7.2. "ФУНКЦИИ С ПЕРЕМЕННЫМ ЧИСЛОМ ПАРАМЕТРОВ"	40
ЛАБОРАТОРНАЯ РАБОТА №8 "БЛОКОВЫЙ ВВОД-ВЫВОД"	44
ЛАБОРАТОРНАЯ РАБОТА № 9 "СТРОКОВЫЙ ВВОД-ВЫВОД"	53
ЛАБОРАТОРНАЯ РАБОТА № 10 "ДИНАМИЧЕСКИЕ МАССИВЫ"	57
ЛАБОРАТОРНАЯ РАБОТА № 11 "ИНФОРМАЦИОННЫЕ ДИНАМИЧЕСКИЕ СТРУКТУРЫ"	61
БИБЛИОГРАФИЧЕСКИЙ СПИСОК	65