МЕТОДИЧЕСКИЕ УКАЗАНИЯ

к лабораторным работам по предмету

"СИСТЕМНОЕ ПРОГРАММИРОВАНИЕ"

УДК 681.3

Методические указания к лабораторным работам по предмету "Системное программирование"

**Оглавление**

[**ВВЕДЕНИЕ 4**](#_heading=h.gjdgxs)

[**ЛАБОРАТОРНАЯ РАБОТА №1. « ПРОГРАММИРОВАНИЕ ЛИНЕЙНЫХ АЛГОРИТМОВ » 7**](#_heading=h.30j0zll)

[**ЛАБОРАТОРНАЯ РАБОТА №2. «ПРОГРАММИРОВАНИЕ РАЗВЕТВЛЯЮЩИХСЯ АЛГОРИТМОВ » 12**](#_heading=h.3znysh7)

[**ЛАБОРАТОРНАЯ РАБОТА №3. «ПРОГРАММИРОВАНИЕ ЦИКЛИЧЕСКИХ АЛГОРИТМОВ » 16**](#_heading=h.qsh70q)

[**ЛАБОРАТОРНАЯ РАБОТА №4. «МАССИВЫ» 22**](#_heading=h.147n2zr)

[**ЛАБОРАТОРНАЯ РАБОТА №5. «СИМВОЛЬНЫЕ СТРОКИ» 28**](#_heading=h.2iq8gzs)

[**ЛАБОРАТОРНАЯ РАБОТА №6. «ПРОГРАММИРОВАНИЕ С ПОМОЩЬЮ СТРУКТУРНЫХ ЗАПИСЕЙ» 35**](#_heading=h.xvir7l)

[**ЛАБОРАТОРНАЯ РАБОТА №7. «ПРОГРАММИРОВАНИЕ ФУНКЦИЙ» 44**](#_heading=h.4kx3h1s)

[**ЛАБОРАТОРНАЯ РАБОТА №8. «ПРОГРАММИРОВАНИЕ С ИСПОЛЬЗОВАНИЕМ ФАЙЛОВ» 50**](#_heading=h.44bvf6o)

[**ЛАБОРАТОРНАЯ РАБОТА №9. «УКАЗАТЕЛИ» 58**](#_heading=h.2jh5peh)

[**СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ 66**](#_heading=h.ymfzma)

**Введение**

**Разработка программ**

Создание программ зачастую оказывается довольно сложной задачей. Не существует ни полного набора правил, ни алгоритма, указывающего, как написать программу. Это процесс творческий. Тем не менее существует некий схематический план (он приведён ниже), который может помочь в этом нелёгком деле. Из него видно, что весь процесс создания программы можно разбить на две фазы – *фазу решения задачи* и *фазу реализации*. Результатом первой фазы является алгоритм решения задачи, написанный на естественном языке, а результатом второй – получение из алгоритма конечной программы на конкретном языке программирования.



Не стоит легкомысленно относиться к каждому из этапов реализации программы. Особенно тщательно советуем подходить к разработке алгоритма (последовательности точных инструкций, выполнение которых приводит к решению задачи). Правильный, грамотно составленный алгоритм упрощает дальнейший процесс перевода поставленной задачи на язык программирования, уменьшает риск появления ошибок.

**Стиль программирования**

*В важных вопросах главное не искренность, а стиль.*

*Оскар Уайльд*

Программа с тщательно выдержанным стилем выглядит эстетично, к тому же её легче читать, а также вносить в неё исправления и изменения. Поэтому привёдём несколько полезных рекомендаций:

* для объявления основных частей программы или неясных мест используйте комментарии (// и /\*\*/);
* для выделения логических частей программы используйте отступы в начале строк либо вставляйте пустые строки;
* в каждой строке программы должно быть не более одного оператора; сложные операторы советуем оформлять в несколько строк;
* в тех местах программы, где пользователь должен ввести данные с клавиатуры, всегда вставляйте строки с выводом соответствующего приглашения; всегда делайте повторный вывод входных данных;
* используйте осмысленные имена переменных, констант, функций;
* избегайте любого дублирования имён идентификаторов в программе;
* переменные, использующиеся только в определённой функции, должны быть объявлены локальными переменными этой функции, а не внешними переменными;
* включайте в имена переменных указателей префикс **ptr**; такие имена подскажут вам, что эти переменные являются указателями и должны обрабатываться соответствующим образом;
* ссылайтесь на элементы массива при помощи индексов;
* давайте каждой создаваемой структуре имя-этикетку; оно позволит объявлять далее в программе новые переменные, принадлежащие к типу этой структуры;
* явно закрывайте каждый файл, как только выяснится, что программа больше не будет больше обращаться к файлу;
* открывайте файлы только для чтения, если записанные в файле данные не должны изменяться;
* используйте в объявлении класса каждый из спецификаторов доступа (**private, public, protected**) только один раз из соображений ясности и удобочитаемости программы;
* несмотря на то, что тип доступа **private** принят по умолчанию, явно используйте эту метку при перечислении закрытых элементов в объявлении класса (если вы хотите их объявить до **public**);
* хотя в программах на С++ может применяться как память, выделяемая с помощью **malloc** и освобождаться с помощью **free**, так и объекты, создаваемые с помощью **new** и удаляемые с помощью **delete**, всё же лучше использовать только **new** и **delete**;
* при вводе/выводе данных предпочтительнее пользоваться **cin** и **cout**;
* когда в производном классе наследуются ненужные свойства, маскируйте их, переопределяя функции.

**Лабораторная работа №1**

**« Программирование линейных алгоритмов »**

**Цель:**

* научиться писать простейшие программы на С++
* научиться писать простые операторы ввода и вывода
* научиться использовать арифметические операции, изучить их приоритеты
* научиться писать простейшие операторы принятия решений
* познакомиться с часто используемыми математическими функциями стандартной библиотеки С++

Любой конкретный алгоритм может быть записан на языке программирования, использующем только три управляющие структуры: последовательное выполнение *(следование), ветвление и повторение.* В настоящее время это утверждение лежит в основе *структурного программирования*.

Последовательное выполнение столь обычно, что мы редко о нем вспоминаем, как об управляющей структуре. Последовательность операторов выполняется в порядке их естественного расположения в программе, с возможным отклонением для вызова внешнего фрагмента (функции), но с обязательным возвратом в точку вызова.

Язык С++ располагает двумя видами операторов:

* оператор присваивания
* математические операторы

В С++ существуют 5 арифметических операторов: +, -, \*, / и вычисление остатка от деления целых чисел – %. Есть унарная операция - (унарный минус). При делении целых чисел дробная часть отбрасывается. Порядок выполнения операций совпадает с общепринятыми: операции + и - имеют одинаковый приоритет, причем он ниже приоритета операций \*, / и %. Самый высокий приоритет имеет унарный минус. Элементы, заключённые в круглые скобки, имеют более высокий приоритет, чем любые другие арифметические операторы.

Оператор присваивания (=) позволяет заменить значение операнда, расположенного с левой стороны от знака равенства, значением, вычисляемым с правой стороны от него.

**Синтаксис:**

*Переменная = Выражение;*

**Пример:**

х = а + с;

х = 5;

х = у;

либо

*Переменная Оператор= Выражение*

**Пример: Эквивалент:**

x += 2; x = x + 2;

x \*= a + c; x = x\*(a + c);

С++ обладает рядом библиотечных функций: арифметических, тригонометрических, строковых, функций ввода-вывода, работы с символами и т.д.

В данной лабораторной работе вам понадобятся некоторые из них.

|  |  |  |
| --- | --- | --- |
| **Прототип** | **Описание** | **Заголовочный**  **файл** |
| int abs (int) | Абсолютное значение | cstdlib |
| double fabs (double) | Абсолютное значение | cmath |
| double sqrt (double) | Квадратный корень | cmath |
| double pow (double, double) | Возвращает первый аргумент, возведённый в степень, равную значению второго аргумента | cmath |
| double exp (double) | Возвращает е в степени, равной значению аргумента | cmath |
| double log (double) | Натуральный логарифм (ln) | cmath |
| double log10 (double) | Логарифм по основанию 10 | cmath |
| double acos (double) | Арккосинус | cmath |
| double asin (double) | Арксинус | cmath |
| double atan (double) | Арктангенс | cmath |
| double cos (double) | Косинус | cmath |
| double sin (double) | Синус | cmath |
| double tan(double) | Тангенс | cmath |

Если в программе используются библиотечные функции, то в ней должна быть директива с именем соответствующей библиотеки. Например :

*# include <cmath>*

Программа на С++ может осуществлять ввод-вывод данных несколькими способами. Приведём один из них.

Выводить данные и информацию можно с помощью инструкции **cout <<**, а вводить с клавиатуры **– cin >>**.

**Пример:**

cout << “Введите числа х и у: \n”;

cin >> x >> y;

При использовании потоков **cout** и **cin** в начале программы следует подключить следующую библиотеку:

*# include <iostream>*

Приведём пример простой программы, которая возводит число в степень и дадим некоторые пояснения.

//возведение х в степень у

# include <iostream>

# include <cmath>

using namespase std;

int main ()

{

double result, x = 3.0, y = 2.0;

cout << “х в степени у равно: \n”;

result = pow (x,y);

cout << result;

return 0;

}

**Замечание:** Если программа не компилируется и не запускается, то строки

# include <iostream>

# include <cmath>

using namespase std;

можно попробовать заменить строками

# include <iostream.h>

# include <cmath.h>

(если используется старый компилятор).

**Задание №1**

Разработать программу для вычисления арифметического выражения (табл. 1) и вывода полученного результата. Ввод соответствующих исходных данных организовать с помощью оператора присваивания.

**Табл. 1. Варианты выражений для расчетов**

|  |  |  |
| --- | --- | --- |
| **Номер**  **варианта** | **Выражение** | **Исходные**  **данные** |
| 1 | 2 | 3 |
| 1 |  |  |
| 2 |  |  |
| 3 |  |  |
| 4 |  |  |
| 5 |  |  |
| 6 |  |  |
| 7 |  |  |
| 8 |  |  |
| 9 |  |  |
| 10 |  |  |
| 11 |  |  |
| 12 |  |  |
| 13 |  |  |
| 14 |  |  |
| 15 |  |  |
| 16 |  |  |
| 17 |  |  |
| 18 |  |  |
| 19 |  |  |
| 20 |  |  |

**Лабораторная работа №2**

**«Программирование разветвляющихся алгоритмов »**

**Цель:**

* научиться использованию структур выбора *if* и *if-else* для принятия решений
* изучить структуру множественного выбора *switch*
* освоить использование логических операций

Иногда программе необходимо предоставить возможность выбора одного из двух альтернативных вариантов. В С++ это ветвление описывается с помощью условного оператора *if-else*:

**Синтаксис:**

*if (Логическое\_выражение)*

*оператор \_1;*

*else*

*оператор \_2;*

где часть *else* может и отсутствовать.

Сначала вычисляется *"Логическое\_выражение"* в скобках; если оно истинно (отлично от нуля), то выполняется *оператор\_1*. Если *"Логическое\_выражение"* ложно (равно нулю) и *else* присутствует, то выполняется *оператор\_2*, а *оператор\_1* пропускается. Если на месте условно выполняемых операторов должна располагаться группа из нескольких операторов языка, то они заключаются в фигурные скобки({}).

Часто *"Логическое\_выражение"* в скобках представляет условие, заданное с помощью операций *отношений и логических операций.* Операции отношения обозначаются в языке С++ следующим образом:

***==*** *равно;*

***!=*** *не равно;*

***<*** *меньше;*

***>*** *больше;*

***<=*** *меньше или равно;*

***>=*** *больше или равно.*

Символ ! в языке Си обозначает логическое отрицание.

Будем использовать еще две логические операции: **||** означает *логическое ИЛИ*, а *&&* - *логическое И*.

if ((x==1) || (x==y))

cout << “x равно 1 или y. \n”;

else

cout << “x не равно ни 1, ни y. \n”;

Операции отношения имеют приоритет ниже арифметических операций, так что выражение вида *k* *>* *n % i* вычисляется как *k* *>* *(n % i**).* Приоритет *&&* выше, чем у **||**, но обе логические операции выполняются после операций отношения и арифметических операций. В сомнительных случаях лучше расставлять скобки.

В языке С++ также предусмотрен оператор *switch*. Основным его отличием от оператора *if*является то, что он позволяет проверять сразу несколько условий, в результате чего ветвление программы организуется более эффективно.

**Синтаксис:**

*switch (выражение)*

*{*

*case Первое\_значение: выражение;*

*case Второе\_значение: выражение;*

*………………………………………*

*case n-ое\_значение: выражение;*

*default: выражение;*

*}*

Если ни одно из значений операторов *case* не совпадает с возвращённым значением, то выполняются строки программы, стоящие после оператора *default***,** в случае же отсутствия этого оператора в теле блока *switch*управление будет передано следующей за этим блоком строке программы.

**Задание №2**

Разработать программу вычисления выражения (табл. 2) и вывода полученного результата (на экран). Соответствующие исходные данные ввести с клавиатуры.

**Табл. 2. Варианты выражений для расчетов**

|  |  |  |
| --- | --- | --- |
| **Номер**  **варианта** | **Выражение** | **Исходные**  **данные** |
| 1 | 2 | 3 |
| 1 |  |  |
| 2 |  |  |
| 3 |  |  |
| 4 |  |  |
| 5 |  |  |
| 6 |  |  |
| 7 |  |  |
| 8 |  |  |
| 9 |  |  |
| 10 |  |  |
| 11 |  |  |
| 12 |  |  |
| 13 |  |  |
| 14 |  |  |
| 15 |  |  |
| 16 |  |  |
| 17 |  |  |
| 18 |  |  |
| 19 |  |  |
| 20 |  |  |

**Лабораторная работа №3**

**«Программирование циклических алгоритмов »**

**Цель:**

* научиться использованию структуры для многократного исполнения операторов программы
* изучить методики повторения, управляемого счётчиком, и повторения, управляемого контрольным значением
* научиться применению операторов управления **break** и **continue**

В большинстве программ встречаются действия, которые многократно повторяются. Часть программы, в которой повторно исполняется инструкция или группа инструкций, называется **циклом**. В языке С++ цикл можно создать несколькими способами.

**Цикл с предусловием while (пока)**

**Синтаксис:**

*while (условие)*

*оператор;*

Условие обязательно заключено в скобки. Оно понимается в широком смысле и в общем случае может быть произвольным выражением*.* Оператор *while* повторяет выполнение оператора, следующего после условия, до тех пор, пока это условие истинно. Если это условие не истинно с самого начала или становится не истинным, заданный оператор не выполняется и управление передается первому оператору, следующему за оператором цикла.

Если повторяемая часть оператора (тело цикла) содержит не один, а несколько операторов, то вся повторяемая группа должны быть заключена в фигурные скобки:

*while* *(условие)*

*{*

*оператор\_ 1;*

*оператор\_2;*

*……………*

*оператор\_n;*

*}*

Для описания условия в операторе *while* используются те же операции отношений и логические операции, что и для *if* - оператора.

**Цикл с постусловием**

**Синтаксис:**

*do*

*оператор;*

*while (выражение);*

Выполняется «*оператор*», а затем вычисляется «*выражение*». Если оно истинно, то снова выполняется «*оператор*» и т.д. Если «*выражение*» становится ложным, циклический процесс заканчивается.

Тело цикла *do-while*всегда выполняется по крайней мере один раз.

**Оператор for** - это наиболее общий способ организации цикла.

**Синтаксис:**

*for ( выражение\_ 1 ; выражение\_ 2 ; выражение\_ 3 )*

*тело;*

*Выражение\_1* обычно используется для установления начального значения переменных, управляющих циклом. *Выражение\_2* - это выражение, определяющее условие, при котором тело цикла будет выполняться. *Выражение\_3* определяет изменение переменных, управляющих циклом после каждого выполнения тела цикла.

Схема выполнения оператора *for*:

1. Вычисляется *выражение\_1*.

2. Вычисляется *выражение\_2*.

3. Если значения *выражения\_2* отлично от нуля (истина), выполняется тело цикла, вычисляется *выражение\_3* и осуществляется переход к пункту 2, если *выражение\_2* равно нулю (ложь), то управление передается на оператор, следующий за оператором *for*.

Существенно то, что проверка условия всегда выполняется в начале цикла. Это значит, что тело цикла может ни разу не выполниться, если условие выполнения сразу будет ложным.

Любой параметра цикла *for* может быть опущен.

**Пример:** вычисление квадратов чисел от 1 до 9.

int main()

{

int i,b;

for (i=1; i<10; i++)

b = i\*i;

return 0;

}

**Операторы break и continue**. Часто при возникновении некоторого события удобно иметь возможность досрочно завершить цикл. Используемый для этой цели оператор **break (разрыв)** вызывает немедленный выход из циклов, организуемых с помощью операторов **for, while, do-while**, а также прекращение оператора **switch**. Оператор **continue** тоже предназначен для прерывания циклического процесса, организуемого операторами **for, while, do-while**. Но в отличие от оператора **break**, он не прекращает дальнейшее выполнение цикла, а только немедленно переходит к следующей итерации того цикла, в теле которого он оказался. Он как бы имитирует безусловный переход на конечный оператор цикла, но не за пределы самого цикла.

**Задание №3**

1.Вычислить число сочетаний из по по формуле



,



где , целые числа , ввести с клавиатуры.



2.Вычислить значение выражения

,



где ввести с клавиатуры.



3.Вычислить число размещений из по по формуле



,



где целые числа и ввести с клавиатуры .



4.Вычислить приближенно значение бесконечной суммы с точностью до :



Значение и точность расчетов ввести с клавиатуры.



**Примечание.** Считать, что требуемая точность достигнута, если очередное слагаемое оказалось по модулю меньше, чем .



5.Вычислить приближенно значение бесконечной суммы с точностью до 0,0005:



Значение и точность расчетов ввести как константы (см. примечание к варианту 4).



6.Дано натуральное , определить количество цифр в числе и сумму всех его цифр. Значение ввести с клавиатуры.



7.Вывести на печать таблицу значений функции при изменении от до с шагом . Коэффициенты ,,, границы интервала , и число ввести с клавиатуры.



8.Вычислить значение функции для .



Вычисления производить до тех пор, пока . Исходные данные , и ввести с клавиатуры.



9.Найти приближенно, с точностью до 0,01, минимум функции на отрезке . Значения , и ,, ввести как константы.



10.Пусть ; ; ; , . Даны действительные ,, натуральное . Найти .



11.Вычислить бесконечную сумму с точностью .



См. примечание к варианту 4.

12.Вычислить бесконечную сумму с точностью .



См. примечание к варианту 4.

13.Вычислить и вывести на экран таблицу значений функции , больших , если .Исходные данные: .



14.Вычислить и вывести на экран таблицу всех 20 значений функции для . Исходные данные: ; ; ; ; ; ;; ; .



15.Вычислить значение выражения

.



Число ввести с клавиатуры.



16.Составить программу для вычисления и вывода на экран таблицы значений сумм для , изменяющегося с шагом . Исходные данные: ; ; .



17.Вычислить и вывести на экран таблицу значений функции для . Исходные данные (в экспоненциальной форме записи): ; ; .



18.Составить программу вычисления и вывода на экран таблицы сумм , где изменяется в пределах с шагом . Исходные данные: ; ; .



19.Вычислить и вывести на экран таблицу значений функции , для с шагом . Исходные данные: ; ; ; ; ; .



20.Найти наибольшее значение функции при изменении от до с шагом . Исходные данные: ; ; ; ; ; .



**Лабораторная работа №4**

**«Массивы»**

**Цель:**

* познакомиться со структурой данных, называемой массивом
* изучить применение массивов для хранения, сортировки данных и поиска в списках и таблицах значений
* научиться объявлять массив, инициализировать его и ссылаться на отдельные элементы массива
* научиться объявлять массив с несколькими индексами и работать с ним

**Массив -** это конечная совокупность данных одного типа. Можно говорить о массивах целых чисел, массивах символов и т.д. Мы можем даже определить массив, элементы которого - массивы (массив массивов), определяя, таким образом, многомерные массивы.

Любой массив в программе должен быть описан: после имени массива добавляют квадратные скобки [ ], внутри которых обычно стоит число, показывающее количество элементов массива. Например, запись **int** **х[10];** определяет **х** как массив из 10 целых чисел. В случае многомерных массивов указывают столько пар скобок, какова размерность массива, а число внутри скобок показывает размер массива по данному измерению. Например, описание двумерного массива выглядит так : **int** **а[2][5];**. Такое описание можно трактовать как матрицу из 2 строк и 5 столбцов.

Для обращения к некоторому элементу массива указывают его имя и индекс, заключенный в квадратные скобки (для многомерного массива несколько индексов, заключенных в отдельные квадратные скобки):

а[1][3]

х[i]

a[0][k+2]

int а [6]={5,0,4,-17,49,1};

int matr[2][5] = {{ 3,4,0,1,2 }, { 6,5,1,4,6}};

Матрица хранится в памяти построчно, т.е. самый правый индекс в наборе индексов массива меняется наиболее быстро.

Индексы массива в С++ всегда начинаются с 0, а не с 1. Индекс может быть не только целой константой или целой переменной, но и любым выражением целого типа. Переменная с индексами в программе используется наравне с простой переменной (например, в операторе присваивания, в функциях ввода-вывода).

**Пример:**

/\* в массиве найти разность  
мин. и макс. элементов \*/  
int fmax(int x[ ], int n);

{

int max,i=0;

max=x[0];

while (i < n)

{  
 if (x[i] **>** max)

max=x[i];  
 i++;  
 }

return(max);  
 }

#include <stdio.h>  
 main()  
 {

int max,min,i;  
 int a[10] = {1,-2,3,-4,5,-6,7,-8,9,-13};  
 max = fmax(a, 10);  
 i = 0;  
 while (i < 10)  
 {  
 a[i]=-a[i];  
 i++;

}

min=fmax(a,10);  
 printf("мин-мах=%d\n", max+min);

**Задание №4**

1.Задан двухмерный массив (матрица) вещественных чисел. Найти наибольший и наименьший элементы массива. Размерность массива (число столбцов, число строк) и значения его элементов ввести с клавиатуры.

2.Задан двухмерный массив вещественных чисел размерности . Найти номер строки и номер столбца, в которых находится наименьший элемент. Массив описать как типизированную константу.



3.Задан двухмерный массив вещественных чисел. Необходимо каждый элемент соответствующей строки разделить на сумму элементов этой строки. Размерность массива (число столбцов, число строк) и значения его элементов ввести с клавиатуры.

4.Задан двухмерный массив вещественных чисел. Заменить все отрицательные элементы нулями и подсчитать сумму положительных элементов в каждом столбце. Размерность массива (число столбцов, число строк) и значения его элементов ввести с клавиатуры.

5.Заданы натуральное число и последовательность вещественных чисел . Сформировать одномерный массив такой, что , ; . Значение и последовательность чисел ввести с клавиатуры.



6.Заданы натуральное число и последовательность вещественных чисел (результаты экспериментов) . Рассчитать их среднее арифметическое значение по формуле и выборочную дисперсию по формуле . Исходные данные ввести с клавиатуры.



7.Задан двухмерный массив размерности . Получить новый массив с именем путем удаления из строки и столбца, в которых содержится минимальный элемент. Описать массив как типизированную константу.



8.Задан двухмерный массив размерности *.* Дополнить его -й строкой и -м столбцом, в которых записать суммы элементов соответствующих строк или столбцов исходного массива *.* В элементе должна храниться сумма элементов первоначального массива. Исходные элементы массива , и необходимо ввести с клавиатуры.



9.Задана матрица (двухмерный массив) размерности *.* Необходимо, не используя другого вспомогательного массива, транспонировать данную матрицу. Исходные данные ввести с клавиатуры дисплея.



10.Заданы два одномерных массива и размерности соответственно и *.* Сформировать двухмерный массив размерности , каждый элемент которого ; ; , и обнулить отрицательные элементы *.*



11.Задана целочисленная квадратная матрица порядка 4. Из этой матрицы выбрать столбец, который обладает наибольшей суммой модулей элементов. Если таких столбцов несколько, то взять первый из них. Далее в данном столбце найти наименьшее из значений его элементов.

12.Задана вещественная матрица размерности , все элементы которой различны. В каждой строке выбрать элемент с наименьшим значением, затем среди этих чисел выбрать наибольшее. Указать индексы (номер строки и номер столбца) элемента с найденным значением.



13.Заданы целые положительные числа , и *.* Определить и вывестинапечать вектор , элементы которого , где , .



Исходные данные: , , .



14.Задана перестановка из элементов множества и матрица размерности **.** Определить значение функции и вывести его на печать.



Исходные данные: , , т.е. , , , и, например, для (элемент в первой строке и втором столбце): .



15.Даны натуральное число , целочисленная квадратнаяматрица порядка *.* Получить , где – наименьшее из значений элементов (т.е. элементов, находящихся в начале -й строки матрицы до элемента, принадлежащего главной диагонали, включительно).



16.Задана перестановка из элементов множества и матрица размерности . Определить значение функции , и вывести его на печать (см. задание 14). Исходные данные:, ; .



17.Задана перестановка из элементов множества и матрицы и размерности . Определить и вывести на печать значение функции (см. задание 14).



Исходные данные: ; ; и взять из вариантов 14и 16.



18.Задан массив целых чисел следующего вида:



.



Вывести на экран значения его элементов по столбцам в следующем виде:

.



19.Заданы матрицы размерности и размерности . Определить матрицу размерности , равную произведению матриц и , т.е. для ; . Исходные данные: , , .



20.Имеется деталей, которые необходимо обработать на станках. Последовательность запуска деталей в обработку задана перестановкой , где – номер детали (целое число из диапазона ), запускаемой *-*й по порядку. Известно время обработки -й детали на -м станке (задается в виде матрицы) для всех , . Определить общее время обработки всех деталей на основе рекуррентной формулы:



, ; ,



где для всех и . Полученная в результате величина и определит искомоевремя обработки деталей (элемент матрицы , находящийся в -й строке и -м столбце, см. задание 14).



**Лабораторная работа №5**

**«Символьные строки»**

**Цель:**

* изучить функции библиотеки для работы с символами (**cctype**)
* научиться использовать функции из стандартной библиотеки ввода/вывода (**stdio**)
* научиться использовать функции преобразования строк из библиотеки утилит общего назначения (c**stdlib**)
* научиться применять функции обработки строк из библиотеки для работы со строками (c**string**)

**Символы** – это фундаментальные «кирпичики» исходного текста программы.

**Символьная константа** – это величина типа **int**, представляемая в виде символа, заключённого в одинарные кавычки.

**Строкой** называется последовательность символов, с которой обращаются как с одним элементом. Строка может содержать буквы, цифры и различные специальные символы. В С++ строки-константы заключаются в двойные кавычки.

Строка является массивом символов, который заканчивается нулевым символом (‘\0’). Доступ к строке осуществляется через указатель, ссылающийся на первый символ строки.

**Символьные библиотечные функции**

При использовании этих функций необходимо включить в программу директиву *#include <cctype>*.

|  |  |
| --- | --- |
| **Функция** | **Описание** |
| toupper (char\_exp) | Возвращает прописной вариант значения выражения char\_exp |
| tolower (char\_exp) | Возвращает строчный вариант значения выражения char\_exp |
| isupper (char\_exp) | Если значением выражения char\_exp является прописная буква, возвращает *true*, в противном случае *false* |
| islower (char\_exp) | Если значением выражения char\_exp является строчная буква, возвращает *true*, в противном случае *false* |
| isalpha (char\_exp) | Если значением выражения char\_exp является буква, возвращает *true*, в противном случае *false* |
| isdigit (char\_exp) | Если значением выражения char\_exp является одна из цифр от ‘0’ до ’9’, возвращает *true*, в противном случае *false* |
| isspase (char\_exp) | Если значением выражения char\_exp является пробел, символ новой строки или табуляции, возвращает *true*, в противном случае *false* |
| isxdigit (char\_exp) | Возвращает значение *true*, если char\_exp является одним из символов шестнадцатеричного формата, и *false* в других случаях |

**Функции стандартной библиотеки ввода/вывода символов**

Если вы вызываете функции из стандартной библиотеки ввода/вывода, подключите заголовочный файл *<stdio.h>*.

|  |  |
| --- | --- |
| **Функция** | **Описание** |
| getchar (void) | Вводит следующий символ со стандартного устройства ввода и возвращает его в формате целого |
| putchar (char\_exp) | Печать символа, хранящегося в char\_exp |

Приведём пример программы, которая принимает символ с клавиатуры и выводит его на экран, пока не обнаружен признак конца файла.

#include <stdio.h>

main()

{

char c;

while((c=getchar())!==EOF)

//EOF (End Of File) – признак конца файла

putchar(c);

}

**Строковые библиотечные функции**

Строковые значения и строковые переменные не похожи на значения и переменные других типов данных, и многие обычные операции с ними не работают (н-р, = или = =). Эти операции заменяют некоторые библиотечные функции.

|  |  |
| --- | --- |
| **Функция** | **Описание** |
| strcpy (цел\_строка, копир\_строка) | Копирует строковое значение цел\_строка в строковую переменную копир\_строка |
| strcat (цел\_строка, присоед\_строка) | Присоединяет строковое значение присоед\_строка к окончанию значения строковой переменной цел\_строка |
| strlen (измеряемая\_строка) | Возвращает целое число, равное длине строки измеряемая\_строка без учёта нуль-символа |
| strcmp (строка1, строка2) | Если строки строка1 и строка2 совпадают, возвращает 0; Если строка1 меньше, чем строка2, возвращает отрицательное число, в противном случае – положительное число |
| strstr (строка1, подстрока1) | Возвращает указатель на первую встреченную в строке1 подстроку1. Если подстрока1 не найдена, возвращает NULL |
| strchr (строка1, символ1) | Возвращает указатель на первый встреченный в строке1 символ1. Если символ1 не найден, возвращает NULL |
| strrchr (строка1, символ1) | Возвращает указатель на последний встреченный в строке1 символ1. Если символ1 не найден, возвращает NULL |
| Вышеуказанные функции находятся в заголовочном файле *<cstring>* | |
| atoi (const char a[]) | Преобразует строку символов в число типа **int** |
| atof (const char a[]) | Преобразует строку символов в число типа **double** |
| Вышеуказанные функции находятся в заголовочном файле *<cstdlib>* | |

Для чтения вводимой строки и присвоения символов этой строки строковой переменной может использоваться функция вида:

**getline (строк\_перем, макс\_длина +1)**

**Задание №5**

1.Дана строка символов. Преобразовать данную строку, удалив из нее каждую пару символов и повторив (вставив еще раз) каждую пару символов . После преобразования полученную строку вывести на печать.



Примечание. Удалить символ – не значит заменить его пробелом, так как в данном случае пробел тоже символ.

2.Дана строка символов. Исключить из этой строки группы символов, расположенные между скобками [, ]. Сами скобки тоже должны быть исключены. Предполагается, что внутри каждой пары скобок нет других скобок.

3.Дана строка символов , в которой могут встречаться цифры, пробелы, буква и знаки +, –. Известно, что первый символ является цифрой. Из данной строки выделить подстроку ( заранее не известно) символов, предшествующих первому пробелу. Требуется определить, является ли эта подстрока числом. Если да, то выяснить: целое или вещественное число, положительное или отрицательное.



4.Задан одномерный массив, каждым элементом которого является строка символов, состоящая из одних цифр. Рассматривая каждую строку как число, определить сумму четных и нечетных значений элементов массива.

5.Задан одномерный массив, каждым элементом которого является строка символов, состоящая из одних цифр. Упорядочить элементы массива по возрастанию их числовых значений и вывести на экран. От максимального элемента отнять значение минимального и вывести разность на экран. Подсчитать среднее значение всех элементов.

6.Дана строка символов, состоящая из нулей, единиц и пробелов. Группы нулей и единиц, разделенные пробелами (одним или несколькими) и не содержащие пробелов внутри себя, будем называть словами. Требуется найти самое длинное и самое короткое слово в строке и, рассматривая эти слова как числа, определить их сумму.

7.Дана строка символов, состоящая из нулей, единиц и пробелов. Группы нулей и единиц, разделенные пробелами (одним или несколькими) и не содержащие пробелов внутри себя, будем называть словами. Требуется подсчитать количество слов в данной строке. Рассматривая слова как числа, определить количество слов, делящихся на 5 без остатка.

8.На складе хранится ряд деталей, наименования которых представляют некоторую последовательность символов (например, Д21А, Д52Н и т.д.). Подсчитать количество наименований деталей, которые начинаются с Д2. Все имеющиеся наименования вводить с клавиатуры; поскольку число деталей заранее не известно, заканчивать ввод данных следует вводом пустой строки (строки, не содержащей ни одного символа), т.е. нажатием клавиши <ВВОД> без ввода наименования детали.

9.Разработать программу, которая работает следующим образом. Пользователь вводит свою фамилию, год рождения и место рождения (город). Программа подсчитывает, сколько ему лет, формирует строку символов вида

фамилия – количество\_лет – место\_рождения’

и запоминает ее. Следующий пользователь вводит аналогичные данные о себе и т. д. Ввод данных заканчивается вводом пустой строки. По окончании ввода программа выводит на экран сведения о пользователях.

10.Разработать программу, которая проверяла бы орфографию (правильность написания) слов в некоторой строке, сравнивая их со словами из словаря. Использовать при этом в качестве словаря одномерный массив слов (описать в программе в виде типизированной константы).

11.Дана строка символов, содержащая некоторый текст на русском языке. Разработать программу форматирования этого текста, т.е. его разбиения на отдельные строки (по 30 символов в каждой строке) и выравнивания по правой границе путем вставки между отдельными словами необходимого количества пробелов. Вывести на экран отформатированный текст.

12.Дана строка символов, содержащая некоторый текст. Разработать программу, которая определяет, является ли данный текст палиндромом, т. е. читается ли он слева направо так же, как и справа налево (например, “А роза упала на лапу Азора”).

13.Дан массив строк, каждый элемент которого (строка длиной 4 символа) может быть интерпретирован как целое двоичное число без знака. Разработать программу преобразования представления значений элементов массива в шестнадцатеричную систему счисления. Не использовать для этих целей вспомогательных массивов. Распечатать значение элементов массива до и после преобразования.

14.Дан массив строк, каждая из которых может рассматриваться как целое шестнадцатеричное число без знака. Разработать программу преобразования представления строк массива из шестнадцатеричной системы счисления в двоичную. Распечатать введенные шестнадцатеричные значения и полученный массив.

15.Составить программу, которая читает произвольные строки длиной 80 символов, разбивает их по словам (подстрока между двумя пробелами), находит максимальное слово и выводит его на печать. Ввод строк заканчивается вводом символа ’!’, который не является элементом строки.

16.Составить программу, которая читает построчно текст другой программы (ввести его с клавиатуры в размере 10 строк), обнаруживает комментарии и выводит их на печать.

17.Составить программу, которая читает произвольный текст (5 строк по 40 символов) и распечатывает в алфавитном порядке все латинские буквы, входящие в этот текст.

18.Составить программу, которая читает построчно (строка – 80 символов) текст другой программы (ввести его с клавиатуры в размере 10 строк), считает число ключевых слов ’begin’ и число ключевых слов ’end’, сравнивает их и выводит на печать сообщение об ошибке, если они не равны между собой.

19.Разработать программу, которая предназначена для зашифровки текстов. Принять следующий тривиальный алгоритм шифрования. Все буквы в исходном тексте заменяются на , буквы – на , буквы – на и т.д. (по вашему усмотрению). Вывести на печать исходный текст и результат шифрования. Текст читать построчно (строка – 80 символов), хранить текст в массиве строк.



20.Составить программу, которая предназначена для ввода вещественных чисел с фиксированной точкой и проверки, является ли введенное число палиндромом. Палиндром принимает одно и то же значение при чтении его как справа налево, так и слева направо. Программа должна быть рассчитана на ввод десяти чисел.

**Лабораторная работа №6**

**«Программирование с помощью структурных записей»**

**Цель:**

* научиться создавать и использовать структуры, объединения и перечисления
* изучить передачу структур в функции по значению и по ссылке
* научиться работе с данными с помощью поразрядных операций

**Структура** - это совокупность логически связанных переменных, возможно, различных типов, сгруппированных под одним именем для удобства дальнейшей обработки.

Традиционным примером структуры служит учетная карточка работающего: служащий предприятия описывается набором атрибутов, таких, как табельный номер, имя, дата рождения, пол, адрес, номер цеха (отдела), зарплата и т.д. В свою очередь, некоторые из этих атрибутов сами могут оказаться структурами. Таковы, например: имя, дата рождения, адрес, имеющие несколько компонент.

Элементы структуры обозначаются идентификаторами, с которыми можно связывать описатели типов. В результате структуру можно рассматривать как единое целое и осуществлять при необходимости выбор составляющих ее элементов.

Для образования структуры присваивают имена каждому из включаемых элементов и структуре в целом. Затем посредством специального описания задают иерархию, порядок следования и типы элементов, включаемых в структуру. Так, для рассмотренного выше примера с анкетой служащего можно выбрать имена:

tab\_nom - табельный номер;

fio - фамилия, имя, отчество;

ро1 - пол;

adres - адрес;

summa - зарплата.

Все эти понятия можно объединить в такую, например, структуру:

struct anketa

{

int tab\_nom;

char fio [30];

char data [10];

int pol;

char adres [40];

float summa;

};

Эта запись называется описанием структуры. Она начинается с ключевого слова **struct** и состоит из заключенного в фигурные скобки списка описаний. За словом **struct** может следовать необязательное имя, которое называется **именем типа структуры** (иногда его называют тэгом или ярлыком структуры). Этот ярлык именует структуру и в дальнейшем может использоваться для сокращения подробного описания. Переменные, упоминающиеся в записи, называют **элементами**. Следом за правой фигурной скобкой, заканчивающей список элементов, может следовать список переменных, так же, как и в случае базисных типов. Вот почему в приведенном выше описании структуры после закрывающей фигурной скобки стоит точка с запятой: она завершает пустой список. Описание struct { ... } р1, р2, р3; синтаксически аналогичной р1, р2, р3; в том смысле, что каждый из операторов описывает р1, р2, р3 как переменные соответствующего типа и приводит к выделению для них памяти. Описание же структуры без последующего списка переменных не выделяет никакой памяти. Оно только определяет форму структуры и действует как шаблон. Если такое описание снабжено ярлыком (именем типа), то его можно позже использовать при определении фактических экземпляров структуры. Например, используя указанное выше описание **anketa**, можно с помощью строки

struct anketa а0, а1, a2

описать структурные переменные **а0**, **а1**, **a2**, каждая из которых строится по шаблону, введенному структурой anketa. Любая из переменных **а0**, **а1**, **a2** содержит в строго определенном порядке элементы **tab\_nom****, fio, data****, pol****, adres****, summa**. Эти переменные, как и все остальные переменные языка, получают места в памяти.

Внешние и статические структуры можно инициировать, помещая следом за определением список начальных значений элементов:

struct anketa а0 = { 1086, "Иванов И.И.", "8.14.1969", 0,"Луганск, Советская, 13/6", 215.00 };

Каждой структурной переменной в нашем случае могут быть присвоены шесть значений, имеющих соответствующие базовые типы. Доступ к этим значениям осуществляется с помощью следующей конструкции:

*имя\_структуры**.* *имя\_элемента*

Операция выделения элемента . (точка) связывает имя структуры и имя элемента. Например, мы можем с учетом введенных обозначений написать:

a0.data ="8.14.1969";

al.summa = 0.0;

if (a2.pol == 1)

man = man + 1 ;

Действия над структурами, в общем, ограничены. Все, что можно делать со структурой, - это взять ее адрес с помощью операции **&** и обращаться к ее элементам, как показано выше. Записи нельзя копировать или присваивать как единое целое; их нельзя передавать в функцию или получать оттуда целиком. Однако к указателям на структуры это замечание не относится.

На практике структурные переменные обычно появляются в виде массива или списка. Нетрудно видеть, что наши три переменные **а0**, **а1**, **а2** будет проще использовать, если их объединить в массив, состоящий из элементов типа **struct** **anketa**. Применив в программе описание

struct anketa a[3];

мы можем употреблять в ней, например, такие операторы:

а[0].fio = "Иванов И.И." ;

if (a[i]l.tab\_nom > a[i+l].tab\_nom)

{

p = a[i].tab\_nom;

a[i].tab\_nom=a[i+l].tab\_nom;

a[i+l].tab\_nom = p;

}

**Задание №6**

1. Дана ведомость абитуриентов, сдавших вступительные экзамены в институт. В каждой строке данной ведомости записана фамилия абитуриента, его постоянное местожительство и полученные им оценки по отдельным дисциплинам (например, физике, математике, литературе). Необходимо определить количество абитуриентов, проживающих в городе Минске и сдавших экзамены со средним баллом не ниже 4, распечатать их фамилии в алфавитном порядке.

2. В справочной аэропорта хранится расписание вылета самолетов на следующие сутки. Для каждого рейса указаны его номер, тип самолета, пункт назначения, время вылета. Определить все номера рейсов, типы самолетов и времена их вылета для заданного пункта назначения.

3. Разработать программу ("электронную сваху") для службы знакомств. Имеется два списка: список женихов и список невест. В каждом списке кандидат (жених или невеста) характеризуется записью вида:

struct INFORMATION

{

int Nom;

struct MyData

{

int Vz, Rst, Vs;

};

struct Critery

{

int VzMin, VzMax;

int RstMin, RstMax;

int VsMin, VsMax;

};

};

где Nom – порядковый номер кандидата; MyData – данные о кандидате (возраст, рост, вес); Critery – аналогичная запись, содержащая требования к партнеру (в виде диапазона: Min – Max). Требуется объединить эти списки в список пар (без повторений женихов и невест) с учетом требований кандидатов.

4. Дан список очередности запуска деталей в обработку. Из данного списка определить общее количество деталей, тип которых начинается с Д2 и объем партии запуска не меньше 15. Исходные данные:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Тип детали | Д11 | Д21 | Д31 | Д22 | Д41 | Д23 | Д32 | Д25 |
| Размер партии | 20 | 10 | 15 | 30 | 25 | 6 | 22 | 30 |

5. Дана ведомость абитуриентов, сдавших вступительные экзамены в институт. В каждой строке данной ведомости записана фамилия абитуриента и полученные им оценки по отдельным дисциплинам (например, физике, математике, литературе). Необходимо определить средний балл по институту и распечатать список абитуриентов, средний балл которых выше среднего балла по институту. Первыми в списке должны идти студенты, сдавшие все экзамены на 5.

6. В радиоателье хранятся квитанции о сданной в ремонт радиоаппаратуре. Каждая квитанция содержит следующую информацию: наименование группы изделий (телевизор, радиоприемник и т.п.), марка изделия, дата приемки в ремонт, состояние готовности заказа (выполнен, не выполнен). Необходимо выдать информацию начальнику ателье о состоянии заказов на текущие сутки по группам изделий.

7. Написать программу формирования ведомости об успеваемости студентов. Каждая запись этой ведомости должна содержать номер группы, фамилию студента, средний балл за последнюю сессию. Необходимо распечатать списки студентов по группам. В каждой группе фамилии студентов разместить в порядке убывания среднего балла.

8. На складе комплектующих деталей хранятся N деталей, каждая из которых характеризуется записью следующего вида:

struct DETAL

{

char DTL [10];

int PR1;

boolean PR2, PR3;

};

Здесь DTL – наименование детали; PR1, PR2, PR3 – некоторые признаки детали. Необходимо подготовить детали к запуску в сборочный цех. Для этого их следует разбить на пары (Z1, Z2), где Zl, Z2 – переменные типа DETAL, таким образом, чтобы разность |Z1.PR1-Z2.PR1| была как можно меньшей для каждой пары (Z1, Z2). Вывести на печать все наименования пар запуска.

9. На складе комплектующих изделий хранятся N деталей (см. задание 8). Необходимо рассортировать детали по группам с учетом признаков PR2 и PR1. Первая группа – это детали, для которых PR2=false и PR3=false, для второй группы PR2=false, PR3=true и т. д. Вывести на печать перечень наименований деталей по каждой группе.

10. В магазине имеется список лиц, записавшихся на покупку мебельного гарнитура. Каждая запись этого списка содержит порядковый номер, фамилию, домашний адрес покупателя и дату постановки на учет. Предполагается, что список упорядочен в порядке возрастания даты постановки на учет. Требуется удалить в нем все повторные записи (проверку осуществлять по фамилии и домашнему адресу).

11. В больнице имеется общий список больных, каждый из которых характеризуется записью

struct PACIENT

{

char FIO [25];

int NP;

char DGN [20];

};

где FIO – фамилия и инициалы больного, NP – номер палаты, DGN – диагноз. Требуется вывести на печать перечень больных по каждой палате и отдельно по каждому диагнозу. Разместить больных по палатам так, чтобы больные с одинаковым диагнозом располагались по возможности вместе.

12. Имеется список учета нуждающихся в улучшении жилищных условий. Каждая запись этого списка содержит фамилию, имя, отчество и дату постановки на учет. Список упорядочен по дате постановки на учет. Известно число квартир, выделяемых по данному списку в течение года. Рассчитать, какое количество лет в среднем необходимо ожидать получения квартиры, и вывести на экран весь список с указанием ожидаемого года получения квартиры.

13. В предвыборной кампании проводится регистрация кандидатов в депутаты. Каждый кандидат, подавая заявление на регистрацию, указывает номер округа, в котором он собирается баллотироваться, наименование партии, которую он представляет, свой возраст и профессию. Пресс-служба центральной избирательной комиссии выдает информационный бюллетень, в котором приводит следующую информацию: число поданных заявлений на регистрацию кандидатов каждой политической партии; средний возраст кандидатов от каждой политической партии; наиболее часто встречающаяся профессия для кандидатов по каждой партии. Написать соответствующую программу.

14. Имеется N различных деталей, которые необходимо обработать на одном станке. Каждая деталь характеризуется записью вида

struct DETAL

{

int I, DN, DL;

};

где I – номер детали; DN – длительность наладки станка (в часах) для обработки 1-й детали; DL – длительность обработки I-й детали на станке (в минутах). Очередность обработки всех N деталей на данном станке хранится в виде массива SP типа array [1..N] of integer. Сначала обрабатывается деталь с номером SP[1], затем SP[2] и т.д. Суммарное время обработки каждой детали включает время наладки станка и непосредственно время ее обработки. Определить время обработки всей последовательности деталей.

15. У администратора дорожных касс хранится информация о свободных местах в поездах по всем направлениям на ближайшую неделю. Данная информация представлена в следующем виде: дата выезда, конечный пункт назначения, время отправления, число свободных купейных мест, число свободных плацкартных мест. Оргкомитет международной конференции обращается к администратору с просьбой зарезервировать 50 купейных мест до Берлина на субботу. При этом время отправления поезда должно быть не позднее 10 часов вечера. Выдать на печать время отправления или сообщение о невозможности выполнить заказ в полном объеме.

16. В библиотеке имеется список книг. Каждая запись этого списка содержит фамилии авторов, название книги, год издания. Определить, имеются ли в данном списке книги, в названии которых встречается некоторое ключевое слово (например, "ПЭВМ"). Если имеются, то выдать на печать фамилии авторов, название и год издания всех таких книг. Ключевое слово ввести с клавиатуры.

17. В магазине имеется список поступивших в продажу автомобилей. Каждая запись этого списка содержит марку автомобиля и его параметры: стоимость, расход бензина на 100 км, надежность (число лет безотказной работы), комфортность (отличная, хорошая, удовлетворительная). Покупатель в свою очередь имеет ряд требований по каждому из этих параметров. Эти требования задаются в виде некоторого интервала (например, стоимость – 10...30 тыс. долл.; расход бензина – 4...10 л на 100 км). Вывести на печать перечень автомобилей, удовлетворяющих требованиям покупателя. Требования покупателя ввести с клавиатуры.

18. В бюро по занятости населения (трудовой бирже) ведется список вакантных рабочих мест на предприятиях города. Каждая запись такого списка содержит следующую информацию: наименование организации, местоположение организации (расстояние в км от центра города), наименование должности, требуемая квалификация (разряд или образование), требуемый стаж работы по специальности, заработная плата в месяц, наличие социального страхования (да или нет), продолжительность ежегодного оплачиваемого отпуска. Клиент бюро вводит информацию о своей квалификации и требованиях (например, максимальная удаленность от центра города). Написать программу, которая бы распечатывала для каждого клиента список рабочих мест в соответствии с его требованиями.

19. Дан одномерный массив из N элементов, каждый из которых представляет собой запись вида

struct ELEMENT

{

char St [10];

int Dt;

};

Предполагается, что все элементы массива упорядочены в порядке возрастания Dt. Разработать программу, которая по заданному значению параметра St находит соответствующий элемент массива, выводит его на печать (значения St, Dt) и выдаёт запрос на ввод нового значения Dt. После ввода Dt необходимо скорректировать очередность следования массива так, чтобы упорядоченность оставалась прежней (в порядке возрастания Dt).

**Лабораторная работа №7**

**«Программирование функций»**

**Цель:**

* понять принципы модульного построения программ из небольших блоков, называемых функциями
* научиться созданию новых функций
* понять механизмы обмена информацией между функциями

**Функция** - это именованная часть программы, к которой можно обращаться из других частей программы столько раз, сколько потребуется.

Описание функции состоит из двух частей, которые называются *прототипом* и *определением* функции.

В определении функции описано, как именно функция должна вычислять возвращаемое значение. Определение функции состоит из двух частей: *заголовка* и *тела*. Заголовок определяет имя функции, ее тип и формальные параметры, тело определяет действия над данными, выполняемые функцией. Возвращаемое функцией значение передается в вызывающую программу оператором **return**(выражение). Значение "выражения" есть результат функции.

Будем пока предполагать, что и вызывающая и вызываемая функции находятся в одном файле. Вообще говоря, там они могут располагаться в любом порядке. Но если в нашей операционной системе функция физически следует за вызывающей ее программой (**main**), то надо в последней объявить функцию внешней с помощью описателя **extern** : extern int fun(); либо проще int fun(). В противном случае при компиляции будет выдано сообщение об ошибке.

**Синтаксис функции:**

*[тип] имя ([список формальных параметров])*

*{*

*описания переменных;*

*операторы;*

*return**(выражение);*

*}*

Здесь *список формальных параметров* **–** список разделённых запятыми параметров:

*Тип 1 Формальный параметр1;*

*…………………………………*

*Тип 2 Формальный параметр2;*

Квадратные скобки, как обычно, указывают, что заключенная в них конструкция может отсутствовать. По умолчанию тип функции целый. Описание формальных параметров расположено между списком параметров и левой скобкой. Каждое описание заканчивается точкой с запятой. Формальные параметры функции полностью локализованы в ней и недоступны для любых других функций. Аргументы передаются функции **по значению,** т.е. вызываемая функция получает копии значений фактических параметров, а не их адреса.

Прототип функции представляет собой информацию, необходимую для того, чтобы вызвать эту функцию. В коде прототип должен располагаться до вызова функции, если только она не была определена до этого. Обычно прототипы функций размещают перед функцией **main** программы. Прототип имеет следующий вид:

*[тип] имя ([список формальных параметров]);*

Приведём пример функции, вычисляющей факториал.

int factorial (int n);

//Возвращает факториал n

int main ()

//Алгоритм главной программы мы опустим

{}

int factorial (int n)

{

int product = 1;

while (n>0)

{

product = n \* product;

n--;

}

return product;

}

Функции, которые не возвращают ни одного значения, называются **void-функциями** (от **void** - пустота, пустое место). Например, одной из типичных подзадач программы является вывод на экран результатов вычислений.

**Синтаксис функции:**

*void имя ([список формальных параметров])*

*{*

*описания переменных;*

*операторы;*

*return;*

*}*

**Задание №7**

1. Написать подпрограмму, которая выводит на печать элементы одномерного массива в порядке возрастания их значений. В головной программе вызвать эту подпрограмму для различных массивов.

2. Задана матрица размерности . Получить матрицу .



3. Написать подпрограмму для вычисления суммы с точностью до 0,001, где передать в качестве параметра.



4. Написать подпрограмму для вычисления суммы с точностью до 0,001, где передать в качестве параметра.



5. Написать подпрограмму (процедуру или функцию) для вычисления суммы , когда вид функции заранее не известен. Для этого имя функции передать подпрограмме в виде параметра процедурного типа. В головной программе вызвать подпрограмму для следующих функций : ; ; , где коэффициенты определить в самой функции посредством типизированных констант.



6. Задано множество из целых чисел и его подмножество . При этом заранее не известно. Составить подпрограмму для вычисления функции по заданному значению . Здесь – мощность (число элементов) множества . Запись означает, что сумма берется лишь по тем , которые принадлежат . Вид функции заранее не известен. Функция равна единице при и нулю в противном случае. Совместно с бригадой, выполняющей задание № 7, составить головную программу, в которой следует обратиться к данной программе несколько раз для различных значений и функции , а также – к подпрограмме из задания № 7.



7. Задано множество из целых чисел и его подмножество . При этом заранее не известно. Составить подпрограмму для вычисления функции по заданному значению . Здесь использованы те же обозначения, что и в задании № 6. Вид функции заранее не известен. Совместно с бригадой, выполняющей задание № 6, разработать головную программу с вызовом реализованных подпрограмм (см. задание № 6). В качестве функции выбрать .



8. Даны натуральное и целочисленная матрица . Получить , где – наименьшее из значений элементов , .



9. Составить подпрограмму, которая из произвольной строки, содержащей некоторый текст, выделяет все слова и печатает их в алфавитном порядке (по первой букве). Исходную строку ввести в головной программе.

10. Составить подпрограмму, которая в произвольной строке символов находит наиболее часто повторяющийся символ. В головной программе вызвать эту подпрограмму для различных значений строк.

11. Составить подпрограмму, которая из произвольной строки символов удаляет все повторно встречающиеся символы. Исходную строку описать в головной программе как типизированную константу.

12. Составить подпрограмму, которая печатает все натуральные числа, меньшие , являющиеся палиндромом. Число называется палиндромом, если оно читается одинаково как с начала, так и с конца (например, 383, 22). Число передать подпрограмме как параметр.



13. Даны действительные числа . Найти сумму для следующих функций : , .



14. Составить подпрограмму вычисления сумм для с шагом . Значения передать в качестве параметров.



15. Вычислить значения функции с заданной точностью для каждого . Считать, что требуемая точность достигнута, если вычислена сумма нескольких первых слагаемых и очередное слагаемое оказалось по модулю меньше, чем (это и все последующие слагаемые можно не учитывать).



16. Дана целочисленная матрица размерности . Найти номера строк, все элементы которых делятся на 3 без остатка.



17. В пункте получения багажа аэропорта имеется общий список пассажиров и их багажа. Каждая запись этого списка содержит информацию о фамилии пассажира и его багаже (количество мест, их вес). Составить подпрограмму для печати сведений о пассажирах, багаж которых состоит из одного места весом не менее 25 кг. Совместно с бригадой, выполняющей задание № 18, составить головную программу, в которой вызвать данную или другую подпрограмму (см. задание № 18).

18. По условиям задания № 17 составить подпрограмму для печати сведений о пассажире, багаж которого превышает багаж всех других пассажиров и по числу мест, и по их весу. Совместно с бригадой, выполняющей задание № 17, составить головную программу, в которой поочередно вызывать данные подпрограммы.

**Лабораторная работа №8**

**«Программирование с использованием файлов»**

**Цель:**

* научиться создавать, читать, записывать и модифицировать файлы
* познакомиться с обработкой файлов последовательного доступа
* познакомиться с обработкой файлов произвольного доступа

**Файл –** это группа связанных записей.

Для облегчения поиска в файле заданных записей, по крайней мере, одно из полей в каждой записи выбирается в качестве *ключа записи (record key)*.

Файл рассматривается как последовательность байтов. Каждый файл завершается *маркером конца файла (end-of-line marker)* или указанным числом байтов, записанным в служебную структуру данных поддерживающей системой. Когда файл открывается, то создаётся объект и с этим объектом связывается поток. Потоки, связанные с этими объектами, обеспечивают каналы связи между программой и отдельными файлами или устройствами.

Для обработки файлов в С++ должны быть включены заголовочные файлы *<iostream>*, *<fstream>*. Файл *<fstream>* включает определения классов потоков **ifstream**(для ввода из файла), **ofstream**(для вывода в файл) и **fstream**(для ввода-вывода файлов). Файлы открываются путём создания объектов этих классов потоков. Эти классы потоков являются производными (т.е. наследуют функциональные возможности) от классов **istream**, **ostream**, **iostream** соответственно.

Существует множество способов организации записей в файле. Наиболее распространённый тип организации записей в файле называется *последовательным файлом*, в котором записи обычно хранятся в последовательности, соответствующей ключевому полю.

Файлы открываются на вывод путём создания объекта класса **ofstream**. Этому объекту передаются два аргумента: имя файла и режим открытия файла. Для объекта класса **ofstream** режим открытия файла может быть либо **ios::out** – для вывода данных в файл, либо **ios::app** – для присоединения данных в конец файла. Существующие файлы, открываемые в режиме **ios::out,** усекаются – вся информация в них теряется.

//создание объекта **outClientFile** класса **ofstream**, связанного с файлом **clients.dat**,

//который открывается для вывода

ofstream outClientFile (“clients.dat”, ios::out);

Файлы открываются для ввода путём создания объекта класса **ifstream**. Объекту передаются два аргумента – имя файла и режим открытия файла.

ifstream inClientFile (“clients.dat”, ios::in);

Для последовательного поиска данных в файле программа обычно начинает считывание данных с начала файла и читает все данные последовательно до тех пор, пока не будут найдены требуемые данные. Классы **istream** и **ostream** содержат функции-члены для позиционирования указателя позиции файла(это порядковый номер следующего байта в файле, который должен быть считан или записан). Этими функциями-членами являются **seekg** (позиционировать для извлечения потока) для класса **istream** и **seekp** (позиционировать для помещения в поток) для класса **ostream.** Любой объект класса **istream** имеет так называемый указатель «**get**», который показывает номер в файле очередного вводимого байта; а любой объект класса **ostream** имеет указатель «**put**», который показывает номер в файле очередного выводимого байта.

//позиционирование **inClientFile** на **n** байтов вперёд

inClientFile.seekg(n, ios::cur);

где **n** – смещение от начала файла (целое число), а **ios::cur** – направление позиционирования (может отсутствовать). Можно использовать следующие направления позиционирования: **ios::beg** (по умолчанию) для поциционирования относительно начала потока, **ios::cur** – относительно текущей позиции в потоке, **ios::end** – относительно конца потока.

Функции-члены **tellp** и **tellg** возвращают текущие позиции соответственно указателей «**put**» и «**get**».

Удобным средством при разработке файлов произвольного доступа является применение записей только фиксированной длины. Используя этот метод, программа может быстро определить точное местоположение записи относительно начала файла.

Данные могут быть помещены в файл произвольного доступа без разрушения других данных в файле. Данные могут быть обновлены или удалены без перезаписи всего файла.

Функция-член **write** класса **ostream** выводит в указанный поток некоторое число байтов, начиная с заданной позиции в файле. Когда поток связан с файлом, то данные записываются, начиная с позиции, определённой указателем «**put**».

outFile.write (reinterpret\_cast<const char\*>(&number), sizeof(number));

где оператор **reinterpret\_cast<const char\*>** используется для преобразования адреса **number** в указатель **const char\***, а **sizeof(number)** определяет число байтов, которые должны быть записаны.

Функция-член **read** класса **istream** вводит некоторое число байтов из указанного потока в область памяти, начиная с заданного адреса. Байты берутся из потока, начиная с позиции, определённой указателем «**get**». Функция **read** требует, чтобы первый аргумент имел тип **char\***.

inCredit.read (reinterpret\_cast<char\*(&client), sizeof(clientData));

Функция-член **eof** класса **ios** определяет, достигнут ли признак конца файла в заданном потоке. Признак конца файла устанавливается, когда попытка считывания данных оканчивается неудачей.

**Задание №8**

1. Разработать программу, которая формирует файл F1, содержащие целые числа, и переписывает этот файл в другой файл – F2, помещая в него из F1 только положительные числа.

2. Разработать программу, переписывающую в текстовый файл Т2 содержимое текстового файла Т1, но без строк, содержащих цифры.

3. Разработать программу сортировки (упорядочивания по возрастанию значений элементов) файла, содержащего целые числа.

4. Разработать программу, которая формирует на основе внешнего текстового файла Т1 файл Т2, разбивая Т1 на строки так, чтобы каждая строка оканчивалась либо точкой с запятой, либо содержала 30 любых литер, кроме точки с запятой. В качестве Т1 можно использовать файл с исходным текстом разработанной программы.

5. Разработать программу формирования файла, содержащего сведения о студентах. Каждый элемент этого файла должен содержать следующие данные: номер группы; номер в группе по списку; фамилию, имя, отчество; год рождения; оценки за последнюю сессию.

6. Разработать программу перекодировки текстового файла, заменив в нем все заглавные буквы строчными.

7. Разработать программу формирования и корректировки (отдельных элементов) файла, в котором хранятся сведения о товарах, находящихся на складе. Каждый элемент этого файла должен содержать следующую информацию: наименование товара; объем партии; дату поступления на склад; стоимость единицы товара.

8. Разработать программу вывода на печать (или экран) текстового файла. Печать следует организовать таким образом, чтобы выполнялось выравнивание по правой границе путем вставки между отдельными словами необходимого количества пробелов. В качестве исходного файла можно использовать файл с исходным текстом разработанной программы.

9. Разработать программу слияния двух отсортированных по убыванию значений элементов файлов F1 и F2. Результатом слияния должен быть файл F3, элементы которого упорядочены по возрастанию.

10. Разработать программу создания и корректировки (добавление новых элементов) файла, содержащего сведения о книгах, находящихся в читальном зале библиотеки. Каждый элемент этого файла должен содержать следующую информацию: фамилию и инициалы автора; название книги; наименование издательства; год издания; количество страниц.

11. Задана ведомость абитуриентов, сдавших вступительные экзамены в институт. В каждой строке данной ведомости записана фамилия абитуриента, специальность, на которую он поступает, и полученные им оценки по отдельным дисциплинам (например, физике, математике, литературе). Написать программу для хранения указанной информации в файле на дискете. Программа должна предусматривать ввод, корректировку, дополнение ведомости, а также формирование выходных текстовых файлов по каждой специальности, содержащих фамилии и инициалы абитуриентов и суммарный балл для каждого из них. Записи в выходных файлах должны быть упорядочены по первой букве фамилии в алфавитном порядке.

12. В справочной аэропорта хранится расписание вылета самолетов на следующие сутки. Для каждого рейса указаны номер рейса, тип самолета, пункт назначения, время вылета. Имеются справочники по расстояниям между всеми возможными пунктами назначения и по расходу горючего на тысячу километров для каждого типа самолета. Написать программу для хранения указанной информации в файлах на дискете. Данная программа должна, кроме того, формировать в виде текстового файла заявку на горючее на следующий день (количество в тоннах).

13. У администратора железнодорожных касс хранится информация о свободных местах в поездах по всем направлениям на ближайшую неделю. Данная информация представлена в следующем виде: дата выезда, номер рейса, конечный пункт назначения, время отправления, число свободных купейных мест, число свободных плацкартных мест. Разработать программу для хранения данной информации в файле. Программа должна позволять корректировать записи, выдавать информацию об имеющихся местах по каждому рейсу и каждому типу мест (купейные или плацкартные).

14. В радиоателье хранятся квитанции о сданной в ремонт радиоаппаратуре. Каждая квитанция содержит следующую информацию: наименование группы изделий (телевизор, радиоприемник и т.п.), марку изделия, дату приемки в ремонт, состояние готовности заказа (выполнен, не выполнен). Разработать программу для ведения архива квитанций на персональной ЭВМ. В архиве должны в форме файла храниться квитанции за последний квартал. Необходимо выдавать на основании анализа архива информацию начальнику ателье о числе и характере заказов на текущие сутки и объеме выполненных услуг за текущий квартал.

15. Написать программу для хранения информации об успеваемости студентов. Необходимо хранить номер группы, фамилию студента, оценки за последнюю сессию. Программа должна позволять хранить эту информацию в виде файла на дискете, корректировать указанную информацию и распечатывать списки студентов по группам с указанием среднего балла каждого за последнюю сессию. В каждой группе фамилии студентов в распечатке размешать в порядке убывания среднего балла.

16. На складе комплектующих хранятся детали, каждая из которых характеризуется записью следующего вида:

struct DETAL

{

string[10] DTL;

integer PR1;

boolean PR2, PR3;

};

Здесь DTL – наименование детали; PR1, PR2, PR3 – некоторые признаки детали. Написать программу для ведения справочной информации о деталях на складе. Программа должна хранить указанную информацию в виде файла на дискете и выдавать по запросу пользователя информацию о количестве имеющихся деталей по каждому наименованию, о количестве деталей указанного наименования с заданными признаками, о деталях различных наименований, имеющих одинаковые один, два или три признака.

17. В магазине имеется список лиц, записавшихся на покупку мебельного гарнитура. Каждая запись этого списка содержит порядковый номер, фамилию, домашний адрес покупателя и дату постановки на учет. Написать программу для хранения этого списка в виде файла на дискете. Программа должна по запросу пользователя выдавать информацию об общем числе записавшихся в каждом микрорайоне города (определять по названию улицы), формировать текстовый файл, записи которого содержат фамилии и инициалы очередников и упорядочены по дате постановки на учет.

18. В больнице имеется общий список больных, каждый из которых характеризуется записью

struct PACIENT

{

char FIO [25];

int NP;

char SEX [3];

char DGN [20];

};

де FIO – фамилия и инициалы больного, NP – номер палаты, SEX – пол (мужской или женский), DGN – диагноз. Написать программу для хранения и корректировки списка больных. Программа должна позволять вводить данные о новых больных, размещать их по палатам с учетом диагноза, выводить по запросам пользователя информацию о больных по палатам, находить номер палаты, в которой лежит нужный больной. По запросу пользователя файл, в котором хранится вся эта информация, должен обновляться путем удаления информации об уже выписавшихся больных.

19. Разработать программу ("электронную сваху") для поиска женихов и невест. Имеется два списка: список женихов и список невест. В каждом списке кандидат (жених или невеста) характеризуется записью вида

struct INFORMATION

{

int Nom;

struct MyData

{

int Vz, Rst, Vs;

};

struct Critery

{

int VzMin, VzMax;

int RstMin, RstMax;

int VsMin, VsMax;

};

};

где Nom – порядковый номер кандидата; MyData – данные о кандидате (возраст, рост, вес); Critery – аналогичная запись, содержащая требования к партнеру (минимальное и максимальное значения соответствующего параметра). Написать программу для ведения таких списков на персональной ЭВМ. Программа должна позволять вводить данные о новых кандидатах, делать выборку с учетом заданного ограничения на какую-то характеристику кандидата, формировать текстовый файл, содержащий список пар, с учетом требований кандидатов.

20. В бюро по занятости населения (трудовой бирже) ведется список вакантных рабочих мест на предприятиях города. Каждая запись такого списка содержит следующую информацию: наименование организации, местоположение организации (в километрах от центра города), наименование должности, требуемая квалификация (разряд или образование), требуемый стаж работы по специальности, заработная плата в месяц, наличие социального страхования (да или нет), продолжительность ежегодного оплачиваемого отпуска. Клиент бюро вводит информацию о своей квалификации и требованиях (например, максимальная удаленность от центра города). Написать программу, которая бы позволяла хранить информацию указанной структуры, вносить сведения о вновь появившихся свободных местах, удалять информацию об уже занятых местах, распечатывать для каждого клиента список рабочих мест в соответствии с его требованиями.

**Лабораторная работа №9**

**«Указатели»**

**Цель:**

* понять концепцию указателей
* изучить применение указателей для передачи аргументов в вызове функции по ссылке
* понять связь между указателями, массивами и строками
* научиться использовать и обновлять массивы строк

**Указатель** представляет собой адрес переменной. Он идентифицирует переменную, говоря не о её имени, а о том, где она находится в памяти.

**Синтаксис:**

*Имя\_типа \*Имя\_переменной;*

int \*pointer\_1;

// pointer\_1 – указатель на переменную типа int

pointer\_1 = &v;

//& – оператор получения адреса

Оператор **new** создаёт новую безымянную переменную определённого типа и возвращает указатель на эту новую переменную. Созданные таким образом переменные называются **динамическими**, поскольку создаются и уничтожаются в процессе работы программы.

int \*p;

p = new int;

Оператор **delete** удаляет динамическую переменную и возвращает используемую ею память в кучу для использования другими динамическими переменными.

delete p;

//удаление динамической переменной, на которую указывает p

С помощью ключевого слова **typedef** можно присвоить имя некоторому определению типа и затем использовать это имя для объявления переменных.

typedef int\* IntPtr;

IntPtr pointer\_1;

В С++ переменная-массив представляет собой переменную-указатель, которая указывает на первый элемент массива. Например:

int a[10];

typedef int\* IntPtr;

IntPtr p;

p = a;

Здесь **р** указывает на то же место в памяти, что и **а**.

Для доступа к элементам массива существует два различных способа. Первый способ связан с использованием обычных индексных выражений в квадратных скобках, например, **array[16]=3** или **array[i+2]=7**. Второй способ доступа к элементам массива связан с использованием адресных выражений и операции разадресации в форме

\*(array+16)=3 или \*(array+i+2)=7

Для доступа к элементам двумерного массива чисел типа **int**, например к arr2[1][2], можно использовать следующие выражения:

\*(\*(arr2+1)+2) и (\*(arr2+1))[2]

Над указателями можно выполнять унарные операции: инкремент и декремент. При выполнении операций ++ и --значение указателя увеличивается или уменьшается

В бинарных операциях сложения и вычитания могут участвовать указатель и величина типа **int**. При этом результатом операции будет указатель на исходный тип, а его значение будет на указанное число элементов больше или меньше исходного.

В операции вычитания могут участвовать два указателя на один и тот же тип. Результат такой операции имеет тип **int** и равен числу элементов исходного типа между уменьшаемым и вычитаемым, причем если первый адрес младше, то результат имеет отрицательное значение.

Значения двух указателей на одинаковые типы можно сравнивать в операциях ==, !=, <, <=",">, >= при этом значения указателей рассматриваются просто как целые числа, а результат сравнения равен 0 (ложь) или 1 (истина).

Если размер массива заранее неизвестен, а определяется в процессе работы программы, то такой массив называется **динамическим**.

Динамические массивы, как и переменные-указатели, создаются с использованием оператора **new**, а уничтожаются с использованием оператора **delete [ ]**.

Ещё одним способом применения указателей являются связанные и двусвязные списки.

При простом связанном хранении каждый элемент списка представляет собой структуру **nd**, состоящую из двух элементов: **val** - предназначен для хранения элемента списка, **n** - для указателя на структуру, содержащую следующий элемент списка. На первый элемент списка указывает указатель **dl**. Для всех операций над списком используется описание:

typedef struct nd

{

float val;

struct nd \* n;

} ND;

int i, j;

ND \* dl, \* r, \* p;

Для реализации операций могут использоваться следующие фрагменты программ:

1) печать обоих соседей узла(элемента), определяемого указателем p (см. рис.1)

  
**Рис. 1. Схема выбора соседних элементов**

if((r=p->n)==NULL) printf("\n нет соседа справа");

else printf("\n сосед справа %f", r->val);

if(dl==p) printf("\n нет соседа слева" );

else { r=dl;

while( r->n!=p ) r=r->n;

printf("\n левый сосед %f", r->val);

}

2) удаление элемента, следующего за узлом, на который указывает р (см. рис.2)

  
**Рис. 2. Схема удаления элемента из списка**

if ((r=p->n)==NULL)

printf("\n нет следующего");

p->n=r->n;

free(r->n);

3) вставка нового узла со значением new за элементом, определенным указателем р (см. рис.3)

  
**Рис. 3. Схема вставки элемента в список**

r=malloc(1,sizeof(ND));

r->n=p->n;

r->val=new;

p->n=r;

Связанное хранение линейного списка называется **списком с двумя связями** или **двусвязным списком**, если каждый элемент хранения имеет два компонента указателя (ссылки на предыдущий и последующий элементы линейного списка).

В программе двусвязный список можно реализовать с помощью описаний:

typedef struct ndd

{

float val; /\* значение элемента \*/

struct ndd \* n; /\* указатель на следующий элемент \*/

struct ndd \* m; /\* указатель на предыдующий элемент \*/

} NDD;

NDD \* dl, \* p, \* r;

1) вставка нового узла со значением **new** за элементом, определяемым указателем p, осуществляется при помощи операторов:

r=malloc(NDD);

r->val=new;

r->n=p->n;

(p->n)->m=r;

p->=r;

2) Удаление элемента, следующего за узлом, на который указывает p

p->n=r;

p->n=(p->n)->n;

( (p->n)->n )->m=p;

free(r);

**Стек** - это конечная последовательность некоторых однотипных элементов - скалярных переменных, массивов, структур или объединений. Стек представляет динамическую структуру данных; ее количество элементов заранее не указывается и в процессе работы, как правило изменяется. Операции добавления и удаления элемента, а также доступа к элементу выполняются только в конце списка.

Реализация стеков в программе может быть выполнена в виде последовательного или связанного хранения.

**Задание №9**

1. Разработать программу перемножения двух матриц и размерности *.* Обе матрицы размещаются в оперативной памяти динамически, а значение вводится по запросу с клавиатуры.



2. Разработать программу сортировки (упорядочивания) матрицы размерности так, чтобы элементы в каждой строке отсортированной матрицы располагались по возрастанию и ни один элемент в -й строке не был больше любого элемента в -й строке. Сортировку выполнять над одномерным массивом из элементов, который "накладывается" на исходную матрицу.



3. Разработать программу, которая в матрице размерности меняет местами строку, содержащую элемент с наибольшим значением, со столбцом, содержащим элемент с наименьшим значением. Матрица должна размешаться в оперативной памяти динамически, а значение вводиться по запросу с клавиатуры.



4. Разработать программу вычисления значения многочлена



в целочисленной точке *.* При этом значения коэффициентов вводятся с клавиатуры и динамически размещаются в памяти либо в форме массива, либо в форме стека.



5. Разработать программу формирования стека, содержащего целые числа, и упорядочивания по возрастанию элементов в этом стеке. В процессе упорядочивания элементы стека перемещаться не должны.

6. Разработать программу формирования стека, куда помещается последовательность символов в виде отдельных слов, вводимых с клавиатуры. Каждое слово, помещенное в стек, следует вывести на экран терминала; при этом порядок вывода символов в каждом слове должен быть обратным по сравнению с последовательностью их ввода.

7. Разработать программу формирования и вывода на печать двунаправленного списка. Двунаправленный список является динамической структурой данных и отличается от однонаправленного списка тем, что каждый его элемент хранит не одну ссылку (указатель на следующий элемент), а две. Из них одна указывает на предыдущий элемент, другая – на следующий элемент списка.

8. Разработать программу формирования стека, содержащего целые положительные числа, и его преобразования путем удаления из него всех четных чисел (в процессе преобразования стека его элементы в оперативной памяти перемещаться не должны).

9. Разработать программу, формирующую динамическую структуру данных для хранения генеалогического дерева. Каждая вершина дерева должна содержать следующую информацию: имя и год рождения.

10. Разработать программу вычисления суммы элементов массива, состоящего из вещественных чисел. Массив должен быть размещен в памяти динамически, а значение вводиться с клавиатуры.



11. Разработать программу, определяющую симметричность произвольного текста любой длины. Текст всегда должен оканчиваться точкой. Эту задачу рекомендуется решать с помощью двух стеков. В первый стек следует поместить весь текст, затем во второй стек перенести его половину так, чтобы последний символ текста находился на дне стека. Далее путем поэлементного сравнения этих стеков получить ответ на вопрос о симметричности текста.

12. Разработать программу формирования стека, куда помещается последовательность символов, вводимых с клавиатуры. Процесс ввода символов должен прекращаться, как только среди вводимых символов появляется точка. После этого программа должна реверсировать стек. Под реверсированием здесь понимается изменение направления ссылок в стеке на противоположное, т.е. после реверсирования вершина и дно стека меняются местами.

13. Разработать программу выявления седловой точки в матрице размерности . Матрица размещается в оперативной памяти в форме двухмерного массива динамически (значение вводится по запросу с клавиатуры). Седловой точкой в матрице называют элемент, одновременно наибольший в своей строке и наименьший в своем столбце.



14. Разработать программу вычисления значения выражения следующего вида: .



При этом значения вводятся с клавиатуры и динамически размещаются в оперативной памяти либо в форме массива, либо в форме стека (или двух стеков, один из которых реверсирован по отношению к другому – см. задание 12).



15. Разработать программу слияния двух стеков, содержащих возрастающую последовательность целых положительных чисел, в третий стек так, чтобы его элементы располагались также в порядке возрастания.

16. Разработать программу формирования стека, куда помешаются целые положительные числа, вводимые с клавиатуры. Процесс ввода должен прекращаться, как только среди вводимых чисел появляется отрицательное число. После этого программа должна вывести на экран терминала содержимое стека, при этом порядок выводимых чисел должен быть обратным по сравнению с последовательностью их ввода.

17. Разработать программу добавления к стеку, содержащему возрастающую последовательность целых положительных чисел, нового элемента так, чтобы порядок возрастания в стеке не изменялся. При добавлении нового элемента в стек другие его элементы перемещаться не должны.

18. Разработать программу вывода на экран таблицы кодов ПЭВМ в следующем формате:

код – символ.

Вывод таблицы следует организовать с помощью цикла, в котором управляющая переменная (типа byte) интерпретируется как код символа и изменяется от 30 до 255 (значения кода до 30 на экране не отображаются). Символ следует выводить на экран с помощью переменной типа char, на которую накладывается управляющая переменная цикла (см. оператор @).

19. Разработать программу формирования стека с последующим его преобразованием в двунаправленную очередь. Двунаправленная очередь является динамической структурой данных, каждый элемент которой хранит не одну ссылку (указатель на следующий элемент, как в стеке), а две. Из них одна указывает на предыдущий элемент, другая – на следующий элемент очереди.

20. Разработать программу нормирования матрицы размерности *,* которое заключается в том, что каждый элемент в этой матрице вычисляется на основании исходной матрицы как отношение суммы всех других элементов в его строке к сумме всех других элементов в его столбце. Матрица должна размещаться в оперативной памяти динамически в форме массива, а значения и вводиться с клавиатуры по запросу.



**Список используемой литературы**

1. Дейтел Харви, Дейтел Пол. Как программировать на С++. – М., 2003.
2. Либерти Джесс. Освой самостоятельно С++ за 21 день.
3. Пол Айра. Объектно-ориентированное программирование С++. – М., 1999.
4. Савитч Уолтер. Язык С++. Курс объектно-ориентированного программирования. – М., 2001.

Учебное издание

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

к лабораторным работам по предмету

"СИСТЕМНОЕ ПРОГРАММИРОВАНИЕ"

(для студентов специальности "Прикладная математика")

Составители:

Юрий Владимирович КУЧМА

Татьяна Николаевна ФЕСЕНКО

Редактор

Техн. Редактор

Оригинал-макет

Подписано в печать   14.05.01  
Формат 60x84 . Бумага офсетная.  Гарнитура Times.  
Печать офсетная. Усл. печ. л.  . Усл. кр.-от.   . Уч.-изд. л.  .  
Тираж  100 экз. Издат. № \_\_\_\_. Заказ № \_\_\_\_. Цена договорная.



Издательство Восточноукраинского национального университета им. В.Даля

91034, г. Луганск

кв. Молодежный, 20а

**Адрес редакции:** 91034, г. Луганск, кв. Молодежный, 20а  
**Телефон:** 8 (0642) 46-11-24. **Факс:** 8 (0642) 46-13-64  
**E-mail:** uni@snu.edu.ua **http://**www.snu.edu.ua