ЫСЫК-КУЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
им. К.ТЫНЫСТАНОВА

Кафедра информационных технологий и программирования

Эркинбаев М., Мукамбетова С., Иманканова К.

МЕТОДИЧЕСКОЕ УКАЗАНИЕ

к выполнению лабораторных работ по языку программирования Си для студентов специальностей:

552801.04 «Программное обеспечение вычислительной техники и автоматизированных систем»
552801.02 «Автоматизированные системы обработки информации и управления»
510201 «Прикладная математика и информатика»

Каракол, 2010

УДК 004
ББК 32.973-01
М 54

Рекомандовано к изданию решением Учебно–методического совета (протокол № 7 от 26.03.2010. г.) и методическим советом факультета (протокол № 5 от 25.02.2009 г.) Ысык-Кульского государственного университета им. К.Тыныстанова

Рецензент: . к.ф.м.н. Тултуков Б.Т.

Составители: Эркинбаев М., Мукамбетова С., Иманканова К.

М 54 Методическое указание к выполнению лабораторных работ по языкам программирования Си. /Сост.: Эркинбаев М., Мукамбетова С., Иманканова К. /ЫГУ им. К. Тыныстанова. -Каракол, 2010. — 76 стр.

ISBN 978-9967-431-81-2

Методические указания разработаны на кафедре информационных технологий и программирования.

Могут быть использованы для подготовки к выполнению лабораторных заданий по курсу «Языки программирования и методы трансляции» для специальности 510201 Прикладной математики и информатики и «Программирование на языках высокого уровня» для специальности 552801.04 Программное обеспечение вычислительной техники и автоматизированных систем.

Рассматриваются алгоритмы и программы обработки данных на языке программирования Си. Методическое указание предназначено студентам вузов технических специальностей, а также школьникам старших классов, учащимся колледжей, студентам техникумов. Данное пособие может оказаться полезным для студентов, самостоятельно изучающих язык программирования Си.

М 2404090000-09
ISBN 978-9967-431-81-2

УДК 004
ББК 32.973-01

Введение

Язык Си, созданный Денисом Ритчи в начале 70-х годов в Bell Laboratory американской корпорации AT&T, является одним из универсальных языков программирования. Язык Си считается языком системного программирования, хотя он удобен и для написания прикладных программ. Среди преимуществ языка Си следует отметить переносимость программ на компьютеры различной архитектуры и из одной операционной системы в другую, лаконичность записи алгоритмов, логическую стройность программ, а также возможность получить программный код, сравнимый по скорости выполнения с программами, написанными на языке ассемблера. Последнее связано с тем, что хотя Си является языком высокого уровня, имеющим полный набор конструкций структурного программирования, он также обладает набором низкоуровневых средств, обеспечивающих доступ к аппаратным средствам компьютера. С 1989 года язык Си регламентируется стандартом Американского института национальных стандартов ANSI C. В настоящее время, кроме стандарта ANSI C разработан международный стандарт ISO C (International Standard Organization C).

Отчет по лабораторной работе должен содержать следующие элементы:

- титульный лист, где следует обязательно указать фамилию и группу автора работы; название дисциплины, в рамках которой выполнена работа; название и номер работы.
- формулировка задания к работе.
- алгоритм решения задачи (в виде структурной схемы, а для некоторых лабораторных работ возможно словесное описание алгоритма).
- текст программы с комментариями (обязательные комментарии: заголовок для каждой функции, в котором описывается назначение этой функции; описание всех используемых переменных и констант; заголовки отдельных функциональных блоков программы).
- тесты, иллюстрирующие все основные варианты работы программы.

Лабораторная работа на тему:
Программирование линейных структур

Цель работы: приобретение практических навыков записи арифметических выражений.

А также использования в программе оператора присваивание и функции ввода и вывода.

1.1 Методические указания к выполнению работы

Программа на Си – это набор функции. Главная функция программы должна иметь имя main (главный) и именно с нее начинается выполнение программы. В программе на Си должна быть только одна функции с именем main.

Простейшая структура программы:

```
#include <stdio.h>
main()
{
<внутренние описание>;
<действие>;
}
```

Первая программа, которую мы рассмотрим, — это «Привет!» — программа, которая выведет на экран строчку «Привет!» и закончит своё выполнение.

```
#include <stdio.h>
int main (void) {
printf ("Привет!\n");
return 0;
}
```

Посмотрим на неё внимательно. Первая строчка

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

означает «включи файл stdio.h». В этом файле определяются функции, связанные с вводом и выводом данных. Аббревиатура STDIO означает «STanDard Input/Output Library». Буква «h» после точки означает «header», то есть заголовочный файл. В заголовочных файлах описано, какие функции предоставляет соответствующая им библиотека. Далее идёт функция main. Она начинается с объявления

```
int main(void)
```

что значит: «функция с именем main, которая возвращает целое число (число типа int от англ. *integer number*) и у которой нет аргументов (void)». Слово void можно переводить как ничто. Далее открываются фигурные скобки и идёт описание этой функции, в конце фигурные скобки

закрываются. Функция main — эта главная функция вашей программы, именно она начинает выполняться, когда ваша программа запускается. Между фигурных скобок находится *тело функции*, в котором описана последовательность действий, производимых данной функцией — логика функции. Наша функция производит одно единственное действие:

```
printf ("Привет!\n");
```

Это действие, в свою очередь, есть вызов функции printf из библиотеки stdio. В результате выполнения этой функции, на экран печатается текст Привет!. Обратите внимание на комбинацию "\n" — она задаёт специальный символ, который в действительности является командой текстовому терминалу: «перейти на следующую строку». Таких специальных символов несколько, все они записываются с помощью символа \ (символ backslash) (см. Базовые понятия языка Си/Специальные символы, Язык Си в примерах/ASCII коды символов). Затем идёт команда return 0;, которая завершает выполнение функции и возвращает значение 0. Функция main должна возвращать 0, если выполнение прошло успешно.

Примечания

В действительности, #include<...> есть директива препроцессора, то есть команда, которая выполняется до начала компиляции файла. Смысл этой директивы очень прост и заключается в том, чтобы на место, где указана эта директива, вставить содержимое файла, имя которого указано в угловых скобках. Обычно заголовочные файлы содержат только прототипы функций, то есть просто список функций с указанием аргументов и типа возвращаемого значения.

Переменные объявляются в начале тела функции main — после открывающей фигурной скобки. Объявление начинается со слова, обозначающего тип переменных, имена которых перечисляются через запятую после обозначения типа.

```
int a, b;
```

В языке Си есть несколько типов данных. Они делятся на две группы: целые типы и типы с плавающей точкой.

К первому типу относятся

char,
short,
int,
long,
unsigned char,
unsigned int,
unsigned long .

Ко второму —

float, double.

Линейным называется вычислительный процесс, структурная схема которого не содержит разветвлений, В алгоритме линейной структуры все

действия выполняются последовательно одно за другим. Для реализации алгоритмов линейной структуры обычно используются функции ввода-вывода и операторы присваивания.

Оператор присваивания в Си имеет вид:

<идентификатор> = <выражение>;

При записи выражения в операторе присваивания используются стандартные арифметические операции "-*-". "-", "*", "/".

Разнообразные вычисления — моделирование, решение алгебраических и дифференциальных уравнений — это то, для чего и создавались первые компьютеры. Давайте и мы научимся использовать компьютер для вычислений. Начнём со сложения двух чисел.

```
#include <stdio.h>
int main ()
{
int a, b;
printf ("Введите два числа: ");
scanf ("%d%d", &a, &b);
printf ("%d\n", a + b);
return 0;
}
```

В нашей программе будут две целочисленные переменные: a и b. Это две ячейки памяти, в которых могут хранятся целые числа из определенного диапазона значений (в 32-разрядной архитектуре от -2^{31} до $2^{31}-1$).

Функция scanf, также как и printf, определена в библиотеке stdio. Эта функция считывает данные, которые пользователь (тот, кто запустит вашу программу) вводит с клавиатуры. Слово scan означает «считывать данные», а print — «печатать данные». Буква «f» в конце соответствует первой букве английского слова «formated», то есть scanf и printf есть функции для форматированного ввода и вывода данных.

Первый аргумент у функции scanf — это "%d%d" (то, что стоит между открывающей скобкой и первой запятой). Первый аргумент является описанием формата входных данных, то есть описание типа данных, которые (как мы ожидаем) введёт пользователь. В этой программе мы ожидаем, что пользователь введет два целых числа. Символ % служебный, с него начинается описание формата. Обычно, после него идет один или два символа, определяющих тип входных данных. Формат "%d" соответствует целому числу в десятичной системе счисления (decimal integer). Если вы напишете "%x", то функция будет ожидать ввода целого числа, записанного в шестнадцатиричной системе счисления.

Контрольные вопросы

1. Структура и основные элементы Си-программы
2. Препроцессор языка Си
3. Основные команды препроцессора define и include
4. Константы в языке Си
5. Основные типы в Си
6. Правила для автоматического преобразования типов в выражении.
7.Стандартные функции ввода в Си
8. Стандартные функции вывода в Си
9. Роль ключевого слова unsigned при задании типа.
10.Явные преобразование типов.

1.2 Варианты заданий

ЗАДАНИЕ. Написать программу для вычисления заданной величины по определенной формуле.

№ 1. Вычислить площадь треугольника по формуле:

S= $\frac{1}{2}$ *b*h, где b-основание треугольника, h-высота, опущенная на это основание.

№ 2. Вычислить площадь равнобедренного треугольника по формуле

S= $\frac{1}{2}$ a*$\sqrt{(b^2-\frac{a^2}{4})}$, где a- основание треугольника, b-боковая сторона.

№ 3. Вычислить площадь равностороннего треугольника по формуле:

S= $\frac{1}{4}$ a^2 *$\sqrt{3}$, где a- сторона треугольника.

№4. Вычислить площадь ромба по формуле

S= $\frac{1}{2}$ d_1*d_2, где d_1,d_2 - диагоналы ромба.

№5. Вычислить площадь трапеции по формуле S= c*h, c= (a+b)/2, где h-высота трапеции, c- средняя линия трапеции: c= $\frac{a+b}{2}$, где a,b-основания трапеции.

№6. Вычислить площадь параллелограмма по формуле
S=b*h, где b-основание параллелограмма,h-высота параллелограмма.

№7. Вычислить площадь правильного шестиугольника по формуле

S= $\frac{3}{2}$ $\sqrt{3}$ *a^2, где a- сторона шестиугольника.

№8. Вычислить медиану треугольника, соединяющую вершину A с серединой противоположной стороны a, по формуле

m= $\frac{1}{2}$ $\sqrt{2b^2+2c^2-a^2}$, где a,b,c- стороны треугольника.

№9. Вычислить длину окружности l и площадь круга Sпо формулам: $l = 2\pi R$, $S = \pi R^2$, где R-радиус.

№10. Вычислить радиус r вписанного в треугольник круга по формуле $r = \sqrt{\frac{(p-a)(p-b)(p-c)}{p}}$, где a,b,c - стороны треугольника, p - полупериметр треугольника: $p = \frac{a+b+c}{2}$

№11. Вычислить радиус R круга, описанного около треугольника по формуле:

$R = \frac{a+b+c}{4 \cdot \sqrt{p(p-a)(p-b)(p-c)}}$, где a,b,c-стороны треугольника, p- полупериметр треугольника: $p = \frac{a+b+c}{2}$

№12. Вычислить высоту треугольника, опущенную на сторону a, по формуле $h = \frac{2 \cdot \sqrt{p(p-a)(p-b)(p-c)}}{a}$, где a,b,c - стороны треугольника, p - полупериметр треугольника: $p = \frac{a+b+c}{2}$

№13. Вычислить площадь между окружностью радиуса R и заключенной внутри нее окружностью радиуса r по формуле $S = \pi \cdot (R+r) \cdot (R-r)$.

№14. Вычислить площадь боковой поверхности цилиндра по формуле $S = 2\pi \cdot R \cdot H$, где H-высота цилиндра, R - радиус основания.

№15. Вычислить объем прямоугольного параллелепипеда по формуле $V = a \bullet b \bullet c$, где a,b.c - стороны параллелепипеда.

№16. Вычислить объем тора, образованного вращением круга радиуса r вокруг оси, отстоящей на расстояние R от центра, по формуле $V = 2 \cdot \pi^2 \cdot R \cdot r^2$.

№17. Вычислить путь, пройденный телом при равноускоренном прямолинейном движении:

$S = v_0 \cdot t + \frac{at^2}{2}$, где v_0 - начальная скорость, a – ускорение, t- время.

№18. Вычислить скорость движения спутника по орбите высоты h по формуле

$V = r \cdot \sqrt{\frac{g}{k+h}}$, где $r = 6{,}37 \cdot 10^6$ м, $g = 9{,}81$ м/с^2.

№19. Вычислить вес тела P на поверхности земли по формуле

P= $\gamma \cdot \frac{m_2 \cdot m}{R_2}$, где m_2 - масса земли, *m*- масса тела, *R*-радиус земли.

№20. Вычислить работу A по формуле
A=F$\cdot S \cdot$cos(a), где F-сила, S- перемещение, a-угол между направлениями силы и перемещения.

№21. Вычислить силу тяготения по формуле
F=$\frac{Q \cdot m_1 \cdot m_2}{R^2}$, где *Q*=$6.67 \cdot 10^{-8}$ см^3/(ã c^2) -
гравитационная постоянная, m_1 m_2 - массы тел,
R-расстояние между телами.

№22. Вычислить потенциальную энергию тела E в однородном поле земного тяготения по формуле E= $m \cdot g \cdot h$

№23. Вычислить силу гравитационного притяжения двух тел массой m_1 m_2 по формуле
P= G$\cdot \frac{m_1 \cdot m_2}{R^2}$, G= $6.67 \cdot 10^{-8}$ñì ^{3}(г с^2)- гравитационная постоянная.

№24. Вычислить период математического маятника по формуле
T= 2$\cdot\pi \cdot \sqrt{\frac{l}{g}}$, где l-длина маятника, g=9/8 м/с^2 - ускорение силы тяжести.

№25. Вычислить по закону взаимодействия точечных зарядов (закон Кулона) силу
взаимодействия F:
A= $(q_1 \cdot q_2)/4\cdot\pi\cdot e\cdot r^2$, где q1,q2- величины зарядов, e-абсолютная диэлектрическая проницаемость среды, r- расстояние между точечными зарядами.

№26. Вычислить период собственных колебаний контура по формуле
T=$2\pi\sqrt{LC}$, где L-индуктивность контура, C-емкость контура.

№27. Вычислить сопротивление двух параллельно соединенных проводников по формуле
$R = \frac{R_1 \cdot R_2}{R_1 + R_2}$, где $R_1 R_2$ - сопротивление проводников.

№28. Вычислить количество теплоты, выделяемой за время t в проводнике сопротивлением R, через который проходит ток силой l, по формуле Q=$l^2 \cdot R \cdot t$

№29. Вычислить емкость плоского конденсатора по формуле
C=$(e \cdot S)/4\pi \cdot d$, где S- площадь одной пластины (меньшей, если они

равны), d-расстояние между пластинами, e- диэлектрическая проницаемость материала, находящегося между обкладками.

№30.Вычислить емкость сферического конденсатора по формуле

C=$\frac{e}{1/a-1/b}$, где a,b-радиусу внутренней и внешней сферы, соответственно,e – диэлектрическая проницаемость материала, находящегося между сферами.

2. Лабораторная работа на тему: Программирование разветвленной структуры

Цель работы: приобретение практических навыков записи условных выражений и использования в программе оператора условия и оператора выбора.

2.1 Методические указания к выполнению работы

В программе на Си в выражениях желательно использовать константы и переменные одного типа. Если происходит смешивание типов, то компилятор не считает программу неправильной, а использует набор правил для автоматического преобразования типов:

1. Если операция выполняется над данными двух разных типов, обе величины приводятся к "высшему" из двух типов. Этот процесс называется "повышением" типа.
2. Последовательность имен типов, упорядоченных от "высшего" к "низшему" выглядит так:

double float long int short char

Применение ключевого слова unsigned повышает ранг соответствующего типа данных со знаком.

3. В операторе присваивания конечный результат вычисления выражения в правой части приводится к типу переменной, которой должно быть присвоено это значение. Данный процесс может привести и к "повышению" ик "понижению" типа.

Для сохранения точности вычислений при арифметических операциях все величины типа float преобразуются в данные типа double, а типы char и short преобразуются к типу int. Это существенно уменьшает ошибку округления.

Для организации программ разветвленной структуры на языке Си используется условный оператор, условная операция и оператор переключатель.

Условный оператор

Условный оператор имеет две формы записи:

1.if (выражение) оператор1; не полный форма записи

2. if (выражение) оператор1; else оператор2; полный форма записи

Если выражение истинно, то выполняется оператор 1, если оно ложно, то при

использовании формы 1 управление передается следующему оператору, а при

применении формы 2 выполняется оператор2.

Если необходимо выполнить несколько действий, то используется составной оператор { }.

Пример. Рассмотрим фрагмент программы

```
 if (i<j) i++;
else { j=i-3;
i++;
}
```

Если значение i больше, чем j, то происходит увеличение его на 1.

Если значение j больше, чем i, то выполняется два действия: присвоение нового значения переменой j и увеличение i. В данном случае в ветви else используется составной оператор для объединения двух действий.

Допускается использование вложенных операторов if. Оператор может быть вложен в фазу if или else другого if. Если нет фигурных скобок, то ключевое слово else относится к ближайшему if, у которого нет else.

Пример. Рассмотрим два фрагмента программы

```
 if (a= =b) if (a= =b)
{ if ( a= =0) b=2; if ( a= =0) b=2;
}
else a=2; else a=2;
```

В первой программе else относится к первому if, а во втором -ко второму if.

Для записи условного оператора используются следующие операции сравнения и логические операции:

= = равно

!= не равно

<,<= меньше, меньше или равно

>,>= больше, больше или равно

! инверсия

&& логическое И

!! логическое ИЛИ.

Результатом сравнения является данное типа int, принимающее значение 0 при невыполнении сравнения (ложь) и значение 1 при выполнении условия сравнения (истина).

Условная операция

Это короткий способ записи оператора if. Форма записи оператора следующая:

выражение ? выражение1 : выражение2;

"Выражение" должно быть целого или плавающего типа или указатель.

Если "выражение" равно нулю (ложно), то вычисляется "выражение2", и его значение является результатом операции. Если значение "выражения" отлично от нуля (истинно), то результатом операции является значение "выражения1".

Пример. Нахождение максимального из двух значений и сохранение его в переменной max.

```
max=(a<b) ? b : a;
```

Условную операцию удобно использовать в тех случаях, когда переменной необходимо присвоить одно из двух возможных значений.

Оператор выбора

Оператор предназначен для организации выбора одного из множества вариантов.

Общий вид оператора переключателя:

```
Switch (выражение) {
case метка 1:операторы 1;
case метка 2:операторы 2;
case меткаN:операторыN;
default:операторы;
}
```

Значение "выражения" вычисляется и сравнивается с "метками" (обычно это целые или символьные константы). В случае совпадения выполняется группа операторов соответствующая метке. Оператор default выполняется, если ни один из предыдущих операторов не выполнился. Возможно, использование нескольких меток перед группой операторов. Наличие ветви default необязательно. Желательно в конце группы операторов, соответствующих каждой метке, использовать оператор break для завершения выполнения оператора переключателя. В случае отсутствия оператора разрыва сравнение по меткам будет продолжено.

Пример. Выполнение арифметической операции по заданному знаку в переменной sign.

Фрагмент программы будет следующим:

```
switch(sign) {
case '-':x=y-z; break;
case '+':x=y+z; break;
```

```
 case '*':x=y*z; break;
 case '/':x=y/z; break;
 default:printf("Неизвестная операция\n");
 }
```

Оператор переключатель может быть вложен один в другой, при этом их метки могут совпадать.

Оператор разрыва

Форма записи оператора разрыва: break;

Используется в операторах цикла и в операторе switch. Его выполнение приводит к выходу из указанных конструкций и переход к следующему оператору программы. Если оператор разрыва находится внутри некоторой совокупности вложенных структур, его действие распространяется только на самую внутреннюю

структуру, в которой он непосредственно содержится.

Оператор перехода

Вид оператора goto метка;

Меткой является ряд букв и цифр, начинающихся с буквы. Можно использовать также знак подчеркивания. Число знаков не ограничено, но значащими являются только первые 32 символа. Диапазоном действия оператора является функция, поэтому выполнение оператора не приведет к переходу вне границ функции.

Пример. Программа, которая по введенным значениям длин сторон выясняет, можно ли сложить из них треугольник и вычисляет его площадь. При вычислении значения переменной s используется операция приведения типа: (float). Она необходима для сохранения дробной части, так как тип исходных переменных int, а тип результата float.

```
#include<stdio.h>
#include<math.h>
void main()
{ int a=3,b=5,c=7;
float s,p;
if (a>=(b+c)||b>=(a+c)||c>=(a+b))
printf("треугольник не складывается\n");
else {
s=(float)1/2*(a+b+c);
p=sqrt(s*(s-a)*(s-b)*(s-c));
  printf("p=%f\n",p); }
}
```

Контрольные вопросы

1. Правила для автоматического преобразования типов в выражении.
2. Роль ключевого слова unsigned при задании типа.

3. Использование в программе условной операции.
4. Роль оператора разрыва в операторе переключателе.
5. Формы записи оператора условия.
6. Условная операция.
7. Построение условий в Си.
8.Оператор переключение
9.Операции отношение.
10. Логические операции.

2.2 Варианты заданий

№1.Даны положительные x. y, z. Выяснить, существует ли треугольник с длинами сторон x, y,z. Ответ получить в текстовой форме: "существует" или "не существует". Если треугольник существует, то ответить, является ли он остроугольным.

№2.Доказать, что любую целочисленную денежную сумму, большую 7 рублей можно выплатить без сдачи трёшками и пятерками, то есть для данного целого n>7 найти такие целые неотрицательные a и b, что 3a+5b=n.

№3.Дано натуральное четырехзначное число n (n≤9999). является ли это число полиндромом (перевертышем), как, например, числа 2222, 6116, 0440?

№4. Дано натуральное четырехзначное число n. Верно ли, что это число содержит ровно три одинаковых цифры, как, например, числа 6676, 4544, 0006?

№5. Дано натуральное четырехзначное число n. Верно ли, что все четыре цифры числа различны, как, например, различны все 4 цифры следующих чисел: 0123, 9760, 5432.

№6. Дано натуральное число n (n≤ 100). Определить первую и последнюю цифру числа n.

№7. Дано натуральное число n (n≤ 100). Определить:

1) сколько цифр в числе n;

2) чему равна сумма его цифр.

№8. Дано натуральное число $10 \le n \le 100$. Определить, сколько цифр в числе, и найти предпоследнюю цифру числа n.

№9. Дано натуральное число n (n≤ 99). Выяснить, верно ли, что n равно кубу сумму цифр числа n.

№10. Заданы вершины треугольника A(x1,y1), B(x2,y2), C(x3,y3).

Вычислить длину медианы, проведенной из A.

№11. Написать программу, которая по заданному значению A определяет количество корней уравнения

$x^2 = A + 1$ и выводит значения корней вместе с необходимыми сообщениями.

№12. Известно, что из четырех чисел x1,x2,x3,x4 три равны между собой, а одно отлично от них.

Присвоить переменной NF номер этого числа, а переменной F значение этого числа.

№13. Составить программу, которая бы с помощью оператора переключателя реализовала бы все возможные операции над двумя целыми числами.

№14. Составить программу, которая бы с помощью оператора переключателя реализовала бы все возможные операции над двумя вещественными числами.

№15. Для целого числа К от 1 до 9 вывести фразу "мне К лет", учитывая при этом, что при некоторых значениях К слово "лет" надо заменить на слово "год" или "года".

№16. Для натурального числа К напечатать фразу "мы нашли К грибов в лесу", согласовав окончание слова "гриб" с целым числом К.

№17. Составить программу, которая бы реализовала следующий алгоритм:

по введенным названиям двух нот (до, ре, ми, фа, соль, ля, си) определить интервал, образованный нотами. Секунда -это интервал из двух соседних нот (по кругу), терция -интервал через ноту и т.д. (кварта, квинта, секста, септима)

№18. Введенные значения переменных a, b, c поменять местами так, чтобы оказалось a>=b>=c.

№19. Дано число x. Напечатать в порядке возрастания числа cos(x), 1+|x|, (1+ xx) (1+ x x).

№20. Даны числа a, b, c, d, e, f. Найти координаты точки пересечения прямых, описываемых уравнениями

a x + b y=c и d x + e y=f, если она существует.

№21. Даны числа a, b, c. Если нельзя построить треугольник с такими длинами сторон, то вывести 0, если треугольник равносторонний -1, если равнобедренный - 2, если прямоугольный - 3.

№22. Составить программу согласно условию. Присвоить переменной F значение 1, если ни одно из чисел x, y, z не является положительным и целым, и 0 в противоположном случае.

№23. Составить программу согласно условию. Присвоить переменной F значение 1, если только два числа из трех чисел x, y, z являются положительными и целыми, и 0 в противоположном случае.

№24. Составить программу согласно следующему условию. Присвоить переменной f значение 1, если цифра 3 входит в запись заданного трехзначного числа x, и 0 в противоположном случае.

№25. Заданы координаты вершин треугольника. Выяснить является ли заданный треугольник тупоугольным или нет.

№26. Написать программу, которая по заданному значению A определяет количество корней уравнения $.x^2 = 10-A$ и выводит значения корней вместе с необходимыми сообщениями.

№27. По данным вещественным числам A и B написать программу решения линейного уравнения A*x=B. Программа должна выводить одно из сообщений: "Уравнение не имеет решений", "Уравнение имеет бесконечное множество решений" или "Уравнение имеет единственное решение: x=<значение>",

№28.Даны вещественные положительные числа a,b,c, d. Выяснить, можно ли прямоугольник со сторонами a,b уместить внутри прямоугольника со сторонами c,d так, чтобы каждая из сторон одного прямоугольника была параллельна или перпендикулярна каждой стороне второго прямоугольника.

№29.Даны положительные a,b,c,x. Выяснить, пройдет ли кирпич с ребрами a,b,c в квадратное отверстие со стороной x. Просовывать кирпич в отверстие разрешается только так, чтобы каждое из его ребер было параллельно или перпендикулярно каждой из сторон отверстия. Ответ получить в текстовой форме: "можно" или "нельзя".

№30. Даны вещественные числа a и R, представляющие собой длину стороны квадрата и радиус круга на плоскости. Написать программу, выводящую в зависимости от величин a и R одно из следующих сообщений:

"круг помещается в квадрат";
"квадрат помещается в крут";
"круг и квадрат не помещаются друг в друга".

3. Лабораторная работа на тему: Программирование циклических структур

Цель работы: изучение операторов цикла и особенностей их применения.

3.1 Методические указания к выполнению работы

Оператор цикла for

Данный оператор цикла имеет вид:

for(выражение 1; выражение 2; выражение 3) оператор;

"Выражение 1" задает начальные условия. Оно выполняется всего один раз при начале цикла. "Выражение 2" задает проверку условия. Проверка производится перед каждым возможным выполнением цикла. Когда "выражение 2" становится ложным, цикл заканчивается. "Выражение3" вычисляется в конце выполнения

каждого тела цикла, модифицирует проверяемое выражение.

"Оператор" может быть простым или составным.

Пример. Составить программу вычисления квадратов целых чисел от 0 до 9.

```
#include<stdio.h>
void main()
{ int i;
for(i=0;i<10;i++)
printf("квадрат числа %d = %d\n", i, i*i);
}
```

Возможности оператора цикла for:

1. Можно считать в порядке убывания и возрастания значений параметра цикла.

2. Шаг изменения параметра цикла может быть любым.

Например, for(n=2;n<60;n+=13) оператор;

3. Можно вести подсчет с помощью символов.

Например, for(char ch='a';ch<='z';ch++) оператор;

Этот оператор работает, поскольку символы в памяти машины размещаются в виде чисел.

4. В качестве "выражения3" можно использовать любое правильное выражение.

Его значение будет меняться при каждой итерации.

Пример. Фрагмент программы для табулирования функции y=3x+5 может выглядеть так

```
for(x=1;y<=75;y=3*x++ + 5) printf(" %d %d\n",x,y);
```

В качестве "выражения3" в данном примере используется формула вычисления заданной функции.

Изменение параметра x задается в самой формуле с помощью постфиксной операции инкремента.

5. Можно опустить одно или более выражений. Но при этом не опускаются символы; . Необходимо только включить в тело цикла несколько операторов, которые в конце концов приведут к завершению его работы.

Пример. Рассмотрим фрагмент программы

```
a=3;
for(n=5;a<=50;) a=a*n;
```

В данном случае пропущено "выражение3". Изменение параметра цикла происходит в теле цикла. Эти же

действия можно было бы записать, используя в качестве тела цикла пустой оператор.

```
a=3;
for(n=5;a<=50;a*=n) ;
```

Если опустить все три выражения в операторе цикла for, то можно задать бесконечный цикл, поскольку

пустое условие всегда считается истинным.

Например, for(;;) { операторы}

6. Первое выражение не обязательно должно инициировать переменную. Необходимо только помнить, что

"выражение 1" выполняется только один раз перед тем, как остальные части цикла начнут выполняться.

Например, следующий цикл будет работать, пока не будет введено число больше 5.

```
for(printf("введите числа\n");num<=5;scanf("%d",&num);
```

7. Параметры, входящие в выражения можно изменять в теле цикла.

8. Операция "запятая" позволяет включать в спецификацию цикла несколько инициализирующих и

корректирующих выражений. Операция "запятая" (,) связывает два выражения в одно, причем левое

выражение будет выполняться первым.

Например, for(i=0,j=10;i<j;i++,j--) оператор;

Оператор цикла while

Данный оператор цикла имеет вид:

while(выражение) оператор;

Сначала вычисляется значение выражения, если оно ложно, то управление передается на следующий за циклом оператор. Если условие истинно, то выполняется тело оператора. Оператор может быть пустым, простым и составным. Пустой оператор состоит только из ;. При выполнении оператора ничего не происходит. Используется, когда тела цикла не требуется, хотя по синтаксису нужен хотя бы один оператор.

Пример. Рассмотрим следующий фрагмент программы

```
index=1; while(index++<=5) printf("Привет!\n");
```

Переменная index отвечает за число повторений цикла. В результате работы приведенного фрагмента программы на экран пять раз будет выведено слово

«Привет!».

Оператор do

Оператор имеет следующую форму записи:

```
do
оператор;
while(выражение);
```

Оператор do используется в тех случаях, когда тело цикла должно выполниться хотя бы один раз. Вначале выполняется оператор, затем вычисляется выражение. Если выражение истинно, то цикл повторяется. Если выражение ложно, то управление передается следующему за циклом оператору.

Пример. Реализация фрагмента программы выдачи подсказки с использованием

оператора do.

```
do {
printf("введите Y или N\n");
scanf("%c",&c);
}
while(c!='Y' && c!='N')
```

До тех пор пока не будет введен один из правильных ответов, оператор будет выдавать подсказку и считывать введенный символ.

Оператор продолжения

Оператор имеет следующий вид: continue;

Его действие заключается в прерывании выполнения тела цикла. После этого вызывается следующая итерация (шаг) этого цикла.

Пример. Фрагмент программы вывода четных чисел до 100.

```
 for(i=0;i<100;i++)
 { if(i%2) continue;
 printf("%d\n",i);
 }
```

Если значение i -четно, то остаток от деления на 2 будет равен нулю и, следовательно, результат проверки условия в операторе if - ложь, и будет выполнен оператор печати. В противном случае, результат проверки условия -истина и, следовательно, будет выполнен оператор продолжения. В этом случае мы, минуя оператор печати, сразу переходим к следующей итерации (следующему значению i).

В операторах цикла очень удобно использовать оператор разрыва для перехода к следующему оператору программы.

Пример. Фрагмент программы выхода из бесконечного цикла при вводе числа, отличного от нуля.

```
 for(;;){
scanf("%d",&num);
if(!num) break:
 }
```

Контрольные вопросы

1. Что такое цикл?
2. Какие операторы цикла языка Си вам известны?
3. Оператор цикла for – общий вид.
4. Возможности оператора цикла for
5. Использования оператора-продолжения и оператора-разрыва в циклах.
6. Оператор цикла while – общий вид.
7. Особенности применения оператора while.
8. Оператор цикла do – общий вид.
9. Когда используется оператора цикла do?
10. Как задать бесконечный цикл?

3.2 Варианты заданий

№ 1. Возьмем натуральное число n найти факториал этого числа.

№ 2. Вычислить S = sqrt(3 + sqrt(6 + sqrt(9 + ... + sqrt(96 + sqrt(99)...).

№ 3. Написать программу, выводящую все делители заданного числа z

№ 4. Числа Фибоначчи определяются формулами F(0)=F(1)=1, F(i)=F(i-1)+F(i-2), i=2,3,...
Найти 35-е число Фибоначчи.

№ 5. Написать программу, выводящую все целочисленные решения уравнения
a*x+b*y=c

№ 6. Найти сумму чисел Фибоначчи, больших M и меньших N, где M и N -заданные натуральные числа,
1<M<N.

№ 7. Вычислить сумму всех чисел Фибоначчи, которые не превосходят 100.

№ 8. Написать программу вычисления числа ПИ

$$\pi = 4 \cdot \left(1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \cdots + (-1)^n \frac{1}{2 \cdot n - 1}\right)$$

для заданного n.

№ 9. Вычислить S = (1 + 1/3) * (1/5 + 1/7) * (1/9 +1/11 + 1/13) *...* (1/33+ 1/35 + 1/37 + 1/39 + 1/41 + 1/43).

№ 10. Вычислить S = 4/2 + (4 7)/(2 6) +(4 7 10)/(2 6 10) + ... +(4 7 10 301)/(2 6 10 ... 398).

№ 11. Вычислить S = cos(1 +cos(2 + ... + cos (39+ cos (40)...).

№ 12. Вычислить S = sh(x) = x + x^3/3! + x^5/5! + ...+ x^ (2 N+1) / (2 N+1)! для заданного N.

№ 13. Вычислить S = cos (x) = 1 - x^2/2! + x^4/4! +.+ (-1)^N x^ (2 N)/ (2 N)! для заданного N.

№ 14. Вычислить S = Ln (1+x) = x - x^2/2 + x^3/3 +..+ (-1)^(N-1) x^ N / N

для заданного N и |x|<1.

№ 15. Вычислить S = arctg (x) = x - x^3/3 + x^5/5 +...+ (-1)^ Nx ^(2 N+1)/(2 N+1) для заданного N и |x|<1.

№ 16. В процессе голодания вес пациента за 30 дней снизился с 96 до 70 кг. Было установлено, что

ежедневные потери веса пропорциональны весу тела. Вычислить, чему был равен вес пациента через k дней после начала голодания для k=1,2,3,…,29

№ 17. Дано натуральное число n получит все способы выплаты суммы n с помощью монет достоинством 1,3,5 копеек .

№ 18. Натуральное число из n цифр является числом Армстронга, например: $153 = 1^3 + 5^3 + 3^3$

№ 19. Дано натуральное число n указать все тройки x,y,z, таких натуральных чисел, что $n = x^2 + y^2 + z^2$

№ 20. Вычислить:

$$\cfrac{1}{1+\cfrac{1}{3+\cfrac{1}{5+\cfrac{1}{\cfrac{....}{101+\cfrac{1}{103}}}}}}$$

№ 21. Вычислить $\sqrt{3+\sqrt{6+\cdots\sqrt{3\cdot(n-1)+\sqrt{3\cdot n}}}}$

№ 22. Написать программу, проверяющую , является ли данное число Z совершенным и выводящую сообщение Да или Нет. Совершенным называется число, равное сумме всех своих делителей. 6=1+2+3

№ 23. Написать программу проверяющую, является ли число Z простым, и выводящую сообщение ДА или НЕТ

№ 24. Вычислить S = 1 + 2/2 + (2 4)/(2+4) + ... +(2 4 6 ...(2 N))/(2+4+6+..+(2 N)) для заданного N.

№ 25. Вычислить S = 1/sqrt(1 3) + 1/sqrt(3 5) + ... + 1/sqrt(199 201).

№ 26. Найти первое число Фибоначчи, большее N, где N -заданное натуральное число, большее 1.

№ 27. Подсчитать количество чисел Фибоначчи, которые не превосходят заданного целого числа. Напечатать их.

№ 28. Написать программу вычисления числа ПИ

$$\pi = 4 \cdot \left(\frac{8}{9} \cdot \frac{24}{25} \cdot \frac{48}{49} \cdot \ldots \cdot \frac{(2n+1)^2 - 1}{(2n+1)^2} \right)$$ для заданного n.

№ 29. Вкладчик взнес x рублей написать программу вычисляющую сколько денег будет у вкладчика через n лет. Кол-во денег каждый год увеличивается на Z % по отношению к предыдущему году.

№ 30. В начальный момент имеется S кг дрожжей. Через каждый час увеличивается на 15% на исходе часа M кг дрожжей удаляется написать программу, вычисляющую количество дрожжей через N часов.

4. Лабораторная работа на тему: Обработка массивов данных

Цель работы: ознакомиться с данными типа массив и основными приемами программирования задач обработки массивов.

4.1 Методические указания к выполнению работы

Массив -это группа элементов одинакового типа. Форма объявления массива на языке Си следующая:

Тип_данных имя_массива[размер]; «Тип_данных» задает тип элементов массива. «Размер» -количество элементов в нем. Элементы массива в Си нумеруются с нуля!

Пример. Описание одномерного массива из 10 целых чисел и двумерного массива вещественных чисел из 10 строк и 10 столбцов.

```
int a[9]; float mas[9][9];
```

Обращение к элементам этих массивов происходит следующим образом: **a[1]** -обращение ко второму элементу массива, **mas[0][0]**

При задании массива возможна и его инициализация. В этом случае присваиваемые значения указываются в квадратных скобках.

Пример инициализации одномерного массива целых чисел:
int b[2]={1,2,3};

Если размер массива не указан, то он определяется по числу начальных значений. Но рекомендуется всегда указывать размер объявляемого массива.

int d[]={31,28,31,30,31,30,31,31,30,31,30,31};

При инициализации многомерных массивов начальные значения для каждой новой строки заключаются в фигурные скобки. Если отдельных фигурных скобок нет, то инициализация производится по мере возрастания индексов.

Примеры инициализации двумерного массива:

int t[1][2]={{4,5,6},{7,8,9}}; int k[1][2]={10,11,12,13,14};

char ch[2][2]={{'n'},{'y'}};

Массив t инициализируется полностью заданными значениями. В массиве k из его шести значений (размер массива k - 2 строки и 3 столбца) инициализируется только первые 5 элементов (это элементы с индексами 0,0 0,1 0,2 1,0 1,1). В массиве ch инициализируются только 2 элемента: ch[0][0]='n' и p[1][0]='y'.

Если не проинициализировать элементы массива перед началом работы с ним, то внешние и статические массивы инициализируются нулем, а автоматические и регистровые будут содержать "мусор", оставшийся в этом участке памяти.

Если задан размер массива, то значения не заданные явно определяются в зависимости от класса памяти.

Пример.

```
/* Некоторые типовые задачи по  обработке одномерного массива*/
#include <stdio.h>
main ()
{
int  a[20],              /*Определение массива a*/
n,                       /*Фактическое число элементов массива*/
i,                       /*Индекс массива*/
maxa,                    /*Значение максимального элемента массива*/
k;                       /*Количество положительных элементов массива*/
/*Ввод одномерного целого массива*/
do {
printf(“Введите количество элементов массива (не более 20)”);
scanf(“%d”,&n);
while (n<1 ‖ n>20);
printf(“Введите количество элементов:\n”);
for (i=0;i<n;i++) scanf(“%d”, &a[i]);
/*Определение значения наибольшего элемента массива*/
```

```
maxa=a[0];
for (i=0;i<n;i++)
if (maxa<a[i])  maxa=a[i];
printf("Значения максимального элемента массива: %d\n ",maxa);
/*Определение количества положительных элементов массива*/
k=0;
for (i=0;i<n;i++)
if (a[i]>0) k++;
printf("Количество положительных элементов  массива: %d\n ",k);
/*Вывод одномерного целого массива*/
for (i=0;i<n;i++) printf ("%d",a[i]);
}
```

Ссылки и массивы

В языке Си существует сильная взаимосвязь между ссылками и массивами, настолько сильная, что ссылки и массивы фактически надо рассматривать одновременно. Любое действие, которое достигается индексированием массива, может быть выполнено и с помощью ссылок. Вариант со ссылками в общем-то будет быстрее, но он, по крайней мере для начинающих, несколько тяжеловат для понимания.

Описание int a[10] определяет массив a размером в 10 элементов, т.е. это блок из 10 последовательных объектов, именуемых a[0], a[1],..., a[9]. Запись a[i] обозначает элемент в i-й позиции от начала.

Если pa это ссылка на целое значение, описанная как int *pa; то присваивание: pa=&a[0] устанавливает в pa ссылку на нулевой элемент a, т.е. pa содержит адрес a[0].Теперь присваивание x= pa копирует содержимое a[0] в x.

Если содержимое pa указывает на отдельный элемент массива a, то по определению pa+1 указывает на следующий элемент, и, вообще, pa - i указывает на i-й элемент перед pa, а pa+i на i-й элемент после. Таким образом, если pa указывает на a[0], то *(pa+1) относится к содержимому a[1], pa+i есть адрес a[i], а *(pa+i) есть содержимое a[i].

Эти замечания справедливы вне зависимости от типа переменных в массиве a. Определение операции "добавление 1 к ссылке" и другой ссылочной арифметики подразумевает масштабирование, связанное с размером памяти для объекта, на который указывает ссылка. Таким образом, в pa+i значение i, прежде, чем будет добавлено к pa, будет умножено на размер объекта, на который указывает pa. Очевидно, что между индексированием и ссылочной арифметикой связь очень тесная.

Фактически любое упоминание массива приводится транслятором к ссылке на начало этого массива, т.е. имя массива есть ссылочное выражение. Это приводит к небольшому числу полезных следствий. Так как имя массива есть синоним для местоположения нулевого элемента, то присваивание pa=&a[0] можно записать и в таком виде: pa=a Не

удивительно теперь, по крайней мере на первый взгляд, что значение a[i] можно записать как *(a+i).

Вычисляя a[i], транслятор сразу же переводит его в *(a+i); эти две формы полностью эквивалентны. Применяя операцию & к обеим частям этого равенства, получаем, что &a[i] и a+i также идентичны: a+i -адрес iго элемента относительно a. С другой стороны, если pa -ссылка, то ее можно использовать с индексом: pa[i] идентично *(pa+i). Короче, любой массив и индексное выражение можно записать как ссылку и смещение и, наоборот, причем это можно делать даже в одном операторе. Однако между именем массива и ссылкой есть одно различие, о котором следует всегда помнить. Ссылка есть переменная, так что pa=a и pa++ суть осмысленные операции. Имя же массива -константа, а не переменная, поэтому конструкции вроде a=pa или a++, или p=&a недопустимы.

Пример. Программа распечатки содержимого одномерного массива с использованием ссылок. Массив предварительно проинициализирован.

```
#include<stdio.h>
void main()
{
int q[5]={1,2,3,4,5};
int *r;
r = q;
printf("\n");
for(int i=0;i<5;i++)
printf("%d\n",*(r+i));
}
```

Пример. Фрагмент программы, реализующий заполнение двумерного массива случайными элементами с использованием ссылок.

```
int b[5][6],*pb;
randomize();
pb=&b[0][0];
for(i=0;i<5*6;i++)
*(pb+i)=random(2);
```

Рассмотрим теперь, как получить доступ к элементу многомерного массива, используя ссылки. Допустим, в программе описан трехмерный массив и указатель на него:

```
int a1[L][M][K], *ptr;
ptr=&a1[0][0][0];
```

Массив a1 состоит из L элементов, каждый из которых -двумерный массив M на N. Каждый массив M на N в памяти располагается по строкам.

Необходимо получить доступ к элементу a1[i][j[k]. Последовательно это вычисляется так

ptr -адрес 0-го массива M на N

ptr+i*(M*N) - адрес i-го массива M на N
ptr+i*(M*N)+j*N -адрес j-й строки i-го массива M на N
ptr+i*(M*N)+i*N+k - адрес элемента a1[i][j][k]
(ptr+i(M*N)+i*N+k) - значение элемента a1[i][j][k]

Контрольные вопросы

1. Что понимается под массивом?
2. Что называется индексами элемента?
3. С чего начинается индекс?
4. Инициализации одномерного массива
5. Какие способы описания массивов применяются в Си?
6. Обращение к элементу массива с помощью ссылки.
7. Получение адреса и значения элемента многомерного массива
8. Ссылки и массивы
9. Использование ссылки
10. Как получить доступ к элементу многомерного массива?

4.2.Варианты заданий
Обработка одномерного массива

№1. Определить номер наименьшего по абсолютной величине элемента массива A.

№2. Определить наибольший элемент в массиве A и наименьший элемент в массиве B.

№3. Дан одномерный массив A, состоящий из N элементов. Переписать в одномерный массив B все элементы, заключенные между максимальным и минимальным значениями.

№4. Определить произведение наибольшего элемента в массиве A и наименьшего элемента в массиве B.

№5. Дан одномерный массив A, состоящий из N элементов. Подсчитать максимальное количество подряд идущих нулей.

№6. Определить разность наибольшего и наименьшего элементов в массиве A

№7. Дан одномерный массив A, состоящий из N элементов. Перенести в начало массива все четные элементы, а в конец массива -все нечетные.

№8. Определить среднее арифметическое наименьших элементов массивов A и B.

№9. Определить номер наименьшего из значений $A_i^2 - C_i^2$.

№10. Дан одномерный массив A, состоящий из N элементов. исключить из массива первый, предшествующий максимуму, положительный элемент.

№11. Дан одномерный массив A, состоящий из N элементов. Подсчитать максимальное количество подряд идущих отрицательных элементов.

№12. Дан одномерный массив A, состоящий из N элементов. Найти первый и последний положительные элементы массива и подчитать количество элементов, заключенных между ними.

№13. Дан одномерный массив A, состоящий из N элементов. Подсчитать максимальное количество положительных элементов, заключенных между нулями.

№14. Дан одномерный массив A, состоящий из N элементов. Считаем, что отрицательные элементы разбивают его на группы. Найти группу положительных элементов массива с максимальной суммой.

№15. Дан одномерный массив A, состоящий из N элементов. Считаем, что отрицательные элементы разбивают его на группы. Найти количество полученных групп, содержащих нули.

№16 Дан одномерный массив A, состоящий из N элементов. Сколько значений элементов в массиве A встречается более одного раза?

№17 Дан одномерный массив A, состоящий из N элементов. Сколько значений элементов встречается в массиве по 3 раза?

№18 Определить наибольшую по абсолютной величине разность между A_i и A_{i-1}.

№19 Определить номер наибольшего из отношений A_i / Q_i.

№20 Определить номер наименьшей среди сумм $\sum_{i=1}^{m} A_i$, где m = 1, 2, …, n.

№21 Дан одномерный массив A, состоящий из N элементов. Определить количество чисел, входящих в массив по одному разу.

№22 Дан одномерный массив A, состоящий из N элементов. Перенести в начало массива все положительные элементы, а в конец массива -все отрицательные.

№23 Определить номер наименьшей по абсолютной величине разности $A_i - C_i$.

№ 24 Определить наименьшую из разностей $|A_i| - |A_{n-i+1}|$, (n—четное).

№25 Определить наибольшую среди сумм $\sum_{i=1}^{m} (A_i \cdot C_i)$, (m = 1, 2,..., n).

№26 Определить наименьшее из значений $2/A_i + A_i^2$.

№27 Дан одномерный массив A, состоящий из N элементов. Исключить из массива все нулевые элементы, расположенные между максимальным и минимальным элементами.

№ 28 Определить номер m наибольшего среди произведений $\prod_{i=1}^{m} A_i^2$, (m = 1,2,..,n).

№ 29 Дан одномерный массив A, состоящий из N элементов. Исключить из массива первый положительный элемент, следующий за максимальным.

№30 Определить номер наибольшего элемента массива A и наибольшего значения среди модулей элементов массива A.

4.3 Варианты заданий
Обработка матрицы

№1. Для матрицы из 3 столбцов и 7 строк отпечатать номера тех строк, в которых третий элемент больше суммы двух других элементов строки, и число строк такого рода.

№2. Дана матрица символов. Подсчитать количество строк, в которых букв больше, чем цифр.

№3. Для матрицы из *3* строк и 6 столбцов отпечатать номера тех столбцов, в которых первый элемент меньше второго, а второй — меньше третьего, и число таких столбцов.

№4. Дана матрица целых чисел. Собрать все нулевые элементы выше главной диагонали (заполнение осуществлять параллельно главной диагонали).

№5. Для матрицы из 2 столбцов и 10 строк отпечатать номер каждой строки, оба элемента, которой имеют нулевые значения, и число таких строк.

№6. Дана матрица вещественных чисел. Найти максимальный элемент и ближайший к нему (по значению) элемент матрицы. Поиск осуществлять в квадратном контуре, центром которого является максимум, а длина стороны -пять элементов массива.

№7. Для матрицы из 2 столбцов и 10 строк отпечатать номер каждой строки, элементы, которой имеют совпадающие значения, и число таких строк.

№8. Для матрицы из 2 строк и 10 столбцов отпечатать номер каждого столбца, знаки элементов, которого не совпадают, и число таких столбцов.

№9. Дана матрица целых чисел. Подсчитать количество элементов, предшествующих максимуму и количество элементов, следующих за минимумом.

№10. Дана матрица символов. Определить строку, в которой максимальное количество букв.

№11. Дана матрица целых чисел. Собрать все отрицательные элементы выше побочной диагонали (заполнение осуществлять по строкам).

№12. Дана матрица вещественных чисел. Найти максимальный элемент и наиболее удаленный от него (по значению) элемент матрицы. Поиск осуществлять в квадратном контуре, центром которого является максимум, а длина стороны –три элемента массива.

№13. Дана матрица вещественных чисел. Найти минимальный элемент и наиболее удаленный от него (по значению) элемент матрицы. Поиск осуществлять в направлениях, параллельных главной и побочной диагоналям.

№14. Дана матрица целых чисел. Собрать все положительные элементы ниже побочной диагонали (заполнение осуществлять параллельно побочной диагонали).

№15 Дана матрица целых чисел. В строках, все элементы которых четные, расположить элементы в обратном порядке.

№16 Для матрицы из 2 строк и 8 столбцов отпечатать номер каждого столбца, сумма элементов которого меньше нуля, и число таких столбцов.

№17 Дана матрица целых чисел. Собрать все положительные элементы массива выше главной диагонали (заполнение осуществлять по строкам).

№18 Дана матрица символов. Написать программу обращения к каждому элементу этой матрицы, если считать, что имена строк -буквы алфавита (по возрастанию), а имена столбцов -целые числа (по возрастанию).

№19 Для матрицы из 3 столбцов и 6 строк отпечатать номер каждой строки, в которой второй элемент меньше среднего арифметического элементов этой строки, и число таких строк.

№20 Дана матрица из 2 строк и 10 столбцов. Первый элемент каждого столбца представляет абсциссу, а второй — ординату одной из 10 точек плоскости XOY.Отпечатать номера тех столбцов, которыми представлены точки первой четверти плоскости, а также общее количество таких столбцов.

№21 Для матрицы из 3 строк и 7 столбцов отпечатать номера тех столбцов, сумма элементов, которых превышает заданную величину, и число таких столбцов.

№22 Дана матрица вещественных чисел. Найти минимальный элемент и ближайший к нему (по значению) элемент матрицы. Поиск осуществлять в направлениях, параллельных главной и побочной диагоналям.

№23 Для матрицы из 3 столбцов и 6 строк отпечатать номер каждой строки, в которой не менее 2 элементов имеют нулевое значение, и число таких строк.

№24 Дана матрица вещественных чисел. Найти максимальный и минимальный элементы и сумму элементов, заключенных между ними.

№25 Для матрицы из 3 строк и 7 столбцов отпечатать номер каждого столбца, в котором значение его наибольшего элемента оказалось меньше заданной величины, и число таких элементов.

№26 Для матрицы из 2 столбцов и 9 строк отпечатать номер каждой строки, квадрат первого элемента, которой меньше абсолютного значения второго элемента строки, и число таких строк.

№27 Дана матрица с 4 столбцами 6 строками. В каждой строке содержатся оценки одного из 6 студентов по 4 экзаменам. Отпечатать номера тех строк, которым соответствует средний балл студента, больший заданной величины, и число таких строк.

№28 Дана матрица из 3 строк и 7 столбцов, соответствующая учетной ведомости материалов: первый элемент каждого столбца должен соответствовать количеству поступившего материала, второй элемент— израсходованную его часть, а третий — остаток. Отпечатать номера тех столбцов, в которых третий элемент не равен разности двух элементов, и число таких столбцов.

№29 Дана матрица из 2 столбцов и 10 строк. Первый элемент каждой строки представляет абсциссу, второй—ординату одной из 10 точек плоскости. Отпечатать расстояния от начала координат для тех точек, которые принадлежат кругу с заданным радиусом, и число таких точек.

№30 Для матрицы из 3 столбцов и 10 строк отпечатать сумму элементов каждой строки, для которой значение этой суммы оказалось больше заданной величины, но меньше другой заданной величины, и число таких строк.

5. Лабораторная работа на тему: Сортировка массивов

Цель работы: знакомство с методами сортировки данных, наиболее часто применяемых на практике.

5.1 Методические указания к выполнению работы

Сортировкой, или упорядочиванием называется процесс перестановки объектов данного множества в определенном порядке(возрастанию или убыванию).

Сортировка бывает внутренней и внешней. При внутренней сортировке все сортируемые данные помещаются в оперативную память компьютера, где можно получить доступ к данным в любом порядке (то есть используется модель памяти с произвольным доступом). Внешняя сортировка применяется тогда, когда объём упорядочиваемых данных слишком большой, чтобы все данные можно было поместить в оперативную память. Эта цель может быть достигнута с помощью различных алгоритмов, причем каждый из них имеет как свои преимущества, так и свои недостатки.

Метод простого выбора

В исходной последовательности отыскивается наименьший элемент. Этот элемент записывается в выходной массив. Найденное минимальное значение помечается признаком и не участвует в дальнейшей сортировке. Этот процесс необходимо повторить N раз (N -размерности исходного массива).

Модифицированный метод простого выбора

В последовательности чисел отыскивается наименьший элемент, который ставится на первое место. Для того, чтобы не потерять элемент, стоящий на первом месте, его устанавливают на место минимального, т.е. найденный минимальный элемент и первый меняются местами. Затем в усеченной последовательности

(из рассмотрения исключается первый элемент) отыскивается второй наименьший элемент и ставится на второе место. Этот процесс повторяется N-1 раз (N -размер сортируемого массива), пока не встанет на свое место предпоследний N-1 элемент, сдвинув максимальный элемент в конец последовательности.

Метод парных перестановок

Метод заключается в попарно сравнении элементов и их обмене , если они стоят не в порядке возрастания (убывания). Просмотр всего массива и сравнение пар повторяется до тех пор, пока не будут отсортированы все элементы, т.е. во время очередного просмотра не произойдет ни одной перестановки.

Метод "всплывающего пузырька"

Метод получил свое название в результате следующей аналогии. Если элементы в вертикально расположенном массиве представить себе пузырьками в резервуаре с водой, обладающими весом, равным значению элемента, то каждый просмотр массива и перестановки элементов будут рассматриваться как "всплывание" пузырька на соответствующий его весу

уровень. В последовательности сравниваются два соседних элемента, например, K и K+1, допустим, произошла перестановка этих элементов. В этом случае далее делается проверка для всех элементов от K-го до первого, т.е. движение "вверх" по последовательности с перестановкой соответствующих элементов. Чтобы не просматривать массив каждый раз с самого начала, можно запомнить место (индекс) последнего обмена. Все пары соседних элементов с индексами, меньшими запомненного, уже расположены в нужном порядке.

Метод шейкер-сортировки

В последовательности во время первого просмотра ищется минимальный элемент и ставится в начало массива. Далее, во время второго просмотра усеченного массива (исключен из рассмотрения первый элемент) находится максимальный элемент и ставится в конец массива. Затем следующим просмотром определяется минимальный элемент и ставится на свое место и так далее, постоянно чередуя поиск минимума и максимума и уменьшая зону просмотра с краев.

Сортировка методом Шелла

Сортировка методом Шелла подобна сортировке методом парных перестановок в том смысле, что идет сравнение и, в нужном случае, перестановка пар элементов. Но сравниваются не рядом стоящие элементы, а те, которые находятся друг от друга на расстоянии D.

Эта величина первоначально равна D=(N+1)/2 (N - количество элементов сортируемой последовательности). После каждого просмотра расстояние между элементами (D) уменьшается.

D(i+1)=floor((D(i)+1)/2)

Функция floor возвращает округленное в сторону уменьшения значение выражения.

Последний просмотр осуществляется при D=1.

Сортировка подсчетом

Суть метода заключается в определении места (индекса) нахождения каждого элемента исходного массива в отсортированном массиве. Для этого каждый элемент исходной последовательности сравнивается со всеми остальными элементами и подсчитывается, сколько элементов меньше сравниваемого. Это значение определяет число элементов, которые будут находиться перед рассматриваемым элементом в отсортированном массиве и определяет индекс элемента (индексы элементов массива в Си начинаются с 0). Индексы всех элементов хранятся во вспомогательном массиве. В конце работы необходимо расставить элементы на места в соответствии с полученными индексами.

Контрольные вопросы

1. Какие процессы понимаются под сортировкой в Си?

2. Метод простого выбора
3. Модифицированный метод простого выбора
4. Метод парных перестановок
5. Метод "всплывающего пузырька"
6. Метод шейкер-сортировки
7. Сортировка методом Шелла
8. Сортировка подсчетом
9. Сортировка возрастающей порядке.
10. Сортировка убывающей порядке.

5.2 Варианты заданий:

№1. Дана последовательность, расположить ее положительные элементы, стоящие на нечетных местах по возрастанию.

№2. Дана последовательность, расположить ее ненулевые элементы по убыванию.

№3. Переставить строки исходной матрицы так, чтобы убывало количество нулей в строках.

№4. Упорядочить все строки матрицы по числу элементов, кратных 3, т.е. на первое место поставить строку с наименьшим числом таких элементов и т.д., на последнее место -с наибольшим числом таких элементов.

№5. Расположить в порядке возрастания положительные элементы правого верхнего треугольника матрицы.

№6. Расположить в порядке убывания положительные элементы левого нижнего треугольника матрицы.

№7.Дана последовательность, элементы которой есть целые двузначные числа. Упорядочить последовательность по возрастанию сумм цифр соответствующих элементов.

№8. Дана последовательность, расположить ее отрицательные элементы по убыванию.

№9. Дана последовательность, элементы которой есть целые двузначные числа. Упорядочить последовательность по убыванию произведений цифр соответствующих элементов.

№10. Дана последовательность, расположить ее элементы, попадающие в интервал от A до B, в порядке возрастания.

№11. Дана последовательность, расположить ее четные (по значению) элементы по убыванию.

№12. Дана последовательность, расположить ее элементы, кратные 3, по убыванию.

№13. Дана матрица. Упорядочить ее строки, содержащие нули, в порядке возрастания их количества.

№14. Дана матрица. Упорядочить по убыванию положительные элементы ее правой половины.

№15. Дана матрица. Упорядочить по возрастанию ненулевые элементы ее нижней половины.

№16. Дано предложение составить программу, располагающую слова в порядке убывания длины слов.

№17. Дан одномерный массив размерностью N из положительных и отрицательных чисел. Упорядочить его так, чтобы в начале располагались все отрицательные, а затем все положительные элементы, сохранив порядок следования и не создавая новый массив.

№18. В неупорядоченном массиве K(m) есть совпадающие элементы. Из каждой группы одинаковых элементов оставить только один, удалив остальные и поджав массив к его началу.

№19. Составить программу, которая размещает элемент s неупорядоченного массива A на место, соответствующее ему в упорядоченном массиве.

№20. Упорядоченный по возрастанию массив B(n) преобразовать в упорядоченный по возрастанию, оставив по одному в каждой группе совпадающих элементов.

№21. Даны два упорядоченных по возрастанию массива A(m) и B(n). Получить из них путем слияния упорядоченный по возрастанию массив C; совпадающие элементы вставлять единожды. Подсчитать количество элементов в массиве C.

№22. Из двух упорядоченных по возрастанию массивов A(m) и B(n) получить путем слияния упорядоченный по убыванию массив C; удаляемые элементы собрать в массиве D. Подсчитать количество элементов в массивах C и D.

№23. Произвести слияние упорядоченного по возрастанию A(m) и неупорядоченного B(n) массивов (n<<m) в упорядоченный по неубыванию массив C.

№24. Путем слияние из возрастающего A(m) и невозрастающего B(n) массивов получить возрастающий массив C (с удалением совпадающих элементов). Подсчитать количество элементов в массиве C.

№25. Упорядочить строки матрицы A(m,n) лексикографически по убыванию.

№26. Упорядочить строки матрицы B(m,n) по неубыванию элементов ее k-го столбца.

№27. Произвести построчное слияние двух матриц A(m,n) и B(k,n), строки которых лексикографически упорядочены по неубыванию первых r элементов каждой строки.

№28. Упорядочить по неубыванию каждой строку матрицы A(m,n), а после этого перестановкой строк упорядочить всю матрицу по неубыванию элементов первого столбца.

№29. Элементы массива D(n) случайным образом перемешаны. Элементы k_i массива K(n) указывают номера позиций, которые занимали соответствующие элементы d_i до перемешивания. Восстановить исходное состояние массива D.

№30. Матрица L(m,n) состоит из нулей и единиц. Удалить из нее совпадающие строки, а оставшиеся упорядочить по возрастанию двоичных чисел, образуемых строками. Число n больше разрядности компьютера.

6. Лабораторная работа на тему: Обработка строк

Цель работы: знакомство с программными средствами описания и обработки строковых данных в языке Си. Умение использование строковых функций.

6.1 Методические указания к выполнению работы

Строки -это последовательность символов, заключенная в кавычки. В конце каждой строки компилятор добавляет нулевой символ, представляемый управляющей последовательностью \0.

Строка описывается как массив символов. Число элементов массива равно числу элементов в строке плюс символ конца строки (\0). Символьная строка в программе может располагаться на нескольких строках. Для переноса используется обратная дробная черта с последующим нажатием клавиши ввод. Обратная дробная черта игнорируется компилятором, и следующая строка считается продолжением предыдущей. Для работы со строками очень удобно использовать указатели.

Пример. Записать введенную строку символов в обратном порядке.

```
 #include<stdio.h>
void main()
{
 int t,b;
char st[10],tp; /*описание строки как массива символов*/
scanf("%s",st);
```

```
/* при вводе строк символ & не используется, так как имя массива является указателем на его начало */
for(t=0,b=10;t<b:t++,b--)
{
 tp=st[t];
st[t]=st[b];
st[b]=tp;
}
printf("%s\n",st);
}
```

Для ввода одиночного символа из входного потока используется функция getchar(). Для вывода одиночного символа используется функция putchar(ch), где ch -выводимый символ. Аргументом функции вывода может быть одиночный символ (включая знаки, представляемые управляющими последовательностями), переменная или функция, значением которой является одиночный символ.

Пример. Программа вводит из входного потока один символ, а затем выводит

его на экран.

```
 #include<stdio.h>
void main()
{
 char ch;
ch=getchar();
putchar(ch);
}
```

Пример. Программа, реализующая чтение и печать символов до ввода знака .

```
 #include<stdio.h>
#define STOP
void main()
{
 char ch;
while((ch=getchar())!=STOP)
putchar(ch);
}
```

В условии, стоящем после ключевого слова while, реализовано сразу три действия:

ввод символа с помощью функции getchar(); занесение символа в переменную ch; проверка на признак конца ввода. Для реализации проверки на конец ввода последовательности символов используется идентификатор STOP, определенный директивой препроцессора.

Операции со строками

Описание функции работы со строками содержится в файле <string.h> библиотеки стандартных функций.

1. Соединение последовательностей символов.

char *strcat(char *s1,char *s2)

Функция возвращает указатель на полученную строку.

2. Поиск первого вхождения символа в строку.

char *strchr(char *s,int c)

Функция просматривает строку (обращение к строке с помощью указателя s) и ищет символ с кодом с. Возвращает указатель на найденный символ или пустое значение.

3. Сравнение строк.

int strcmp(char *s1,char *s2)

Сравниваются две указанные строки. Результат -переменная типа int:

s1<s2 отрицательное значение

s1==s2 значение 0

s1>s2 положительное значение

4.Копирование символов.

char *strcpy(char *s1,char *s2)

Копируется последовательность символов, указанная параметром s1 по адресу s2.

5. Определение длины строки (без завершающего нуля).

int strlen(char *s)

Проверка символов

Для проверки символов используются функции, возвращающие значения "истина" или "ложь". Прототипы этих функций описаны в файле <ctype.h> библиотеки стандартных функций.

Функция "Истина", если

isalpha(c) с -символ алфавита

isupper(c) с -символ верхнего регистра

islower(c) с -символ нижнего регистра

isdigit(c) с -цифра от 0 до 9

isxdigit(c) с -шестнадцатеричная цифра

isalnum(c) с -буква или цифра

isspace(c) с -пробел, табуляция, перевод строки, перевод формата

Для ввода и вывода строк в программе удобно использовать функции gets(char *string) и puts(char * string).

Пример. Программа проверяет введенную строку и, затем, выводит на экран у ее часть, которая начинается с символа 'y'.

```
#include<stdio.h>
#include<string.h>
void main()
{
char string[10];
char *ptr;
printf("Введите строку\n");
gets(string);
ptr=strchr(string,'y');
printf("это строка %s\n",ptr);
}
```

Пример.

```
/*Ввод двух строк с помощью функции scanf*/
#include <stdio.h>
main ()
{
char name[20], fam[20];
printf(“\n Как Вас зовут (имя и фамилия)?”);
scanf(“%s%s”, name, fam);
printf(“Здравствуйте,%s %s\n”,name,fam);
}
```

Замечание. Для данного случая в функции scanf спецификации %s можно записывать подряд друг за другом: “%s%s”, а можно разделять пробелом: “%s%s” ,- это не имеет никакого значения. При вводе с клавиатуры имя и фамилию можно разделить или пробелом, или клавишей «Ввод »- это также не имеет значения.

Пример.

```
/* Ввод строки с помощью функции gets*/
#include <stdio.h>
main ()
{
char name[40];
printf(“\n Как Вас зовут (имя и фамилия через пробел)?”);
gets(name);
printf(“Здравствуйте,%s \n”,name);
}
```

Замечание 1. Функция gets читает все, что набирает пользователь, до тех пор, пока он не нажмет клавишу «Ввод» . Код клавиши «Ввод» не помещается в строку, однако в конец строки добавляется нулевой символ ‘\0’*/

Замечание 2. Если при вводе строки количество вводимых знаков превысит размер строки, установленный при ее описании, то это скорее

всего приведет к сбою работы программы, так как лишние знаки запишутся в память в “чужую” области, не резервированную под строку.

Пример использования строковых стандартных функций

```
/*strlen – определяет длину строки;
strcpy – копирует вторую строку в первую;
strcat – добавляет копию второй строки в конец первой;
puts – выводит строку на экран и завершает вывод символом ‘\n’,*/
#include <stdio.h>
#include <string.h>
#define s 19
main ()
{
char *msg ;
msg=”Вы изучили язык Си хорошо”; /*Можно было:
strcpy (msg, “Вы изучили язык Си хорошо”); */
puts(msg);
if (strlen(msg)>s)  *(msg+s)=’\0’; puts(msg);
strcat(msg, “плохо”);
puts(msg);
}
/*Программа выдает:
Вы изучили язык Си хорошо
Вы изучили язык Си
Вы изучили язык Си плохо
```

При выполнении оператора if, если длина строки msg больше 19, то в 20-й символ (вместо буквы ‘x’) помещается символ ‘\0’. Остаток массива остается на прежнем месте, но последующая функция puts(msg) прекращает вывод при встрече первого нулевого символа и игнорирует остаток массива*/

Контрольные вопросы:

1. Как описываются строки на языке Си?
2. Функции для ввода и вывода символов.
3. Функции для ввода и вывода строк.
4. Функции проверки символов.
5. Функции, реализующие операции со строками.
6. Соединение последовательностей символов.
7. Поиск первого вхождения символа в строку.
8. Сравнение строк.
9. Копирование символов.
10. Определение длины строки.

6.2 Варианты заданий

№1. Дано предложение, слова в нем разделены пробелом, подсчитать сколько букв "а" в каждом слове.

№2. Дан текст. Определить в нем все слова, в которых доля заданного символа максимальна.

Пример. Текст: "Veni, vidi, vici" ("Пришел, увидел, победил"). Символ: "i".

Результат: слова "vidi", "vici", доля символа {равна 0.5.

№3. Дано предложение, слова в нем разделены пробелом, поменять местами первое и последнее слова.

№4. Дан текст. Выделить из него все слова, которые не содержат одинаковых символов. Различие строчных и прописных букв во внимание не принимать.

Пример. Текст: "Няня Нину мылом мыла". Слова: "мыла

№5. Даны N предложений. Найти в каждом первое слово и напечатать их в строку через пробел.

№6. Дан текст. Найти все слова, в.которые заданный символ входит не менее 2 раз.

№7. Даны N предложений. Подсчитать количество слов в каждом предложении и вывести на печать.

№8. Дано предложение, слова в нем разделены пробелом. Подсчитать количество слов, которые начинаются с той буквы, которой заканчивается предыдущее слово.

№9. Дан текст. Найти такие слова, которые начинаются и кончаются одним символом.

№10. Дано предложение, слова в нем разделены пробелом. Упорядочить слова в порядке возрастания их длины.

№11. Дан текст. Найти все слова, которые оканчиваются тем же символом, что и первое слово.

№12. Дано N предложений, слова в которых разделены пробелами. Вывести их на печать в порядке возрастания количества слов в предложении.

№13. Дано N предложений, слова в которых разделены пробелами. Вывести их на печать в порядке возрастания общей длины слов в предложении (без учета количества разделяющих пробелов).

№14. Дано предложение, слова в нем разделены пробелом. Составить из него два предложения по правилу: в одно переписать все четные по порядку следования слова, а в другое - нечетные.

№15. Дано N предложений, слова в которых разделены пробелом. Составить новый текст по следующему правилу: исключить из текста все слова на букву 'а'.

№16.Дан текст. Определить в нем наиболее часто .встречающийся символ; затем определить все слова, в которых доля этого символа максимальна.

Пример. Текст: "Veni, scrip si, vixi" ("Пришел, написал, прожил"). Наиболее часто встречающийся символ - "i". Слова, в которых доля этого символа максимальна: "vixi" (0.5).

№17. Дано предложение, слова в нем разделены пробелом, подсчитать сколько букв и цифр в последнем слове.

№18. Дан текст. Определить, является ли он "перевертышем". Пример текста-перевертыша: "А роза упала на лапу Азора". Различие строчных и прописных букв во внимание не принимать.

№19.Дан текст. Выделить из текста все слова-"перевертыши". Различие строчных и прописных букв во внимание не принимать.

Например: "top apple pot". Top —pot - слова "перевертыши".

№20. " Дано предложение, слова в нем разделены пробелом, поменять местами четные и нечетные по порядку следования слова.

№21. Дан текст. Найти в нем все симметричные слова. Различие строчных и прописных букв во внимание не принимать.

Примеры 'симметричных слов: Anna, апрпа. ,

№22. Дан текст. Найти в нем слова, jb которых некоторый заданный символ встречается наибольшее число раз.

Пример. Текст: "Алиа and Dasha wain to take this apple". Символ: "a". Результат: слова-"Anna", "Dasha" содержат символ "a" 2 раза.

№ 23. Даны N предложений. Найти в каждом последнее слово и напечатать их в строку через пробел.

№24.Дан текст (английский). Найти все слова, содержащие наибольшее количество гласных латинских букв. (a,e,i,,o,u).

Пример. Anna , Tabic .

№25.Дан текст. Найти все слова, которые содержат символ «t» и встречаются в тексте не менее 2 раз.

Пример. ' Текст: "to be of not to be". Слова: "to".

№26. Дано предложение, слова в нем разделены пробелом. Подсчитать количество слов, которые начинаются с той же буквы, что и последующее слово.

№27.Дан текст. Найти номер первого по порядку слова, которое начинается с заданного символа.

№28. Дано предложение, слова в нем разделены пробелом. Упорядочить слова по алфавиту (только по первой букве).

№29.Дан текст. Выделить из него все слова, не содержащие букв из последнего слова.

№30.Дан текст. Определить в нем все слова, содержащие сочетание "ab", и заменить его на "ba".

7. Лабораторная работа на тему: Обработка структур данных

Цель работы: знакомство с описанием структур данных на языке Си, получение

практических навыков обработки структур с использованием указателей.

7.1 Методические указания к выполнению работы

Записи (структуры) являются одними из основных структур данных в языках программирования высокого уровня. Понятие записи используется при машинной обработке различных документов, таблиц, баз данных.

Запись -это структура, состоящая из фиксированного числа компонент, называемых полями. В одном поле данные имеют один и тот же тип, а в разных полях могут иметь разные типы, за исключением функций.

Объявление структуры осуществляется с помощью ключевого слова struct, за которым идет ее тип и далее список элементов, заключенных в фигурные скобки:

```
struct тип { тип элемента_1 имя элемента_1;
тип элемента_2 имя элемента_2;
тип элемента_n имя элемента_n;
};
```

Именем элемента может быть любой идентификатор. Как и выше, в одной строке можно записывать через запятую несколько идентификаторов одного типа.

Рассмотрим пример:

```
sruct date { int day;
int month;
int year;
};
```

Следом за фигурной скобкой, заканчивающей список элементов, могут записываться переменные данного типа, например:

struct date {...} a, b, c;

(при этом выделяется соответствующая память). Описание без последующего списка не выделяет никакой памяти; оно просто задает форму структуры. Введенное имя типа позже можно использовать для объявления структуры, например:

struct date days;

Теперь переменная days имеет тип date.

Пример. Описание переменных date1и date2. Каждая переменная содержит два поля.

```
struct {
int year;
short day;
} date1, date2;
```

Для описания структуры удобно использовать шаблоны.

Описание шаблона идет без последующего списка переменных.

Формат шаблона следующий

```
struct имя_типа_структуры
{
список описаний;
};
```

Описание шаблона является описанием нового типа данных. Далее можно описывать переменные, используя имя шаблона.

Пример. Описание шаблона для даты (день, месяц, год).

```
struct data{
int day;
char month[10];
int year;
};
struct data d1,d2,d3; /*описание переменных*/
```

При описании возможна инициализация переменной.

```
struct data d1={27,"Январь",2009};
```

Можно задавать имя типа структуры с помощью описателя типа typedef. Описатель типа

Этот описатель позволяет создать свое собственное имя типа.

Формат описателя типа:

```
typedef спецификатор_типа описатели;
```

Спецификатор типа -это основной или производный тип данных или

тип, который ранее определен программистом. Описатель -это новое имя созданного нами типа.

Пример задания нового имени для типа float:

```
typedef float real;
```

После такого описания в программе вместо типа float можно писать тип real, привычный по Паскалю.

Пример. Описание нового типа данных "дата", состоящего их трех полей.

```
typedef struct{
int day;
char month[10];
int year;
}data;
data d1,d2,d3;
```

Структура не может содержать в качестве элемента структуру такого же типа, но может включать указатель на структуру этого типа, при условии, что в объявлении структуры указано имя типа. Это позволяет создавать связанные списки структур.

Пример. Описание бинарного дерева.

```
struct tree{
int number;
struct tree *left;
struct tree *right;
};
```

Доступ к элементу структуры осуществляется с помощью символа ".", обозначающего операцию получения элемента структуры.

Примеры обращения к элементам структур, описанных выше:

```
d1.day
d2.year
```

Доступ к элементам структур может осуществляться и с помощью указателей.

Пример. Описание указателя на структуру и обращение к ее элементам.

```
data * ptr;
ptr=&d2;
ptr->day=5;
ptr->month="Декабрь";
```

ptr->year=2008;

Обращение к элементам структуры можно записать еще следующим образом:

(*ptr).day=5;

(*ptr).month="Декабрь";

(*ptr).year=2008;

Пример. Описание массива, элементами которого являются структуры типа

data.

data d[5];

Пример

/*Дан список автомобилей, каждая строка которого содержит: марку автомобиля, год выпуска, цену. Распечатать список тех автомобилей, год выпуска которых не ранее некоторого заданного года, а цена не превышает некоторой заданной цены*/.

```
#include <stdio.h>
main ()
{
struct {
char marka[10];   /*Марка авто*/
int year;         /*Год выпуска*/
float money;      /*Цены*/
} avto[20]        /* Массив структур-список данных по автомобилям*/
int     n,        /*Количество элементов в массиве avto*/
i,                /*Индекс массива avto*/
y,                /*Контрольный год*/
float  m;         /*Контрольная цена */
printf("Введите количество автомобилей n(n<20");
scanf("%d",&n);
printf("Введите список из %d  автомобилей: \n", n);
for (i=0;i<n;i++)
{ffush(stdin); gets(avto[i].marka);
ffush(stdin);scanf(%d.&avto[i].year);
ffush(stdin); scanf(%d.&avto[i].money);}
printf("Введите контрольный год и цену:");
scanf("%d%f",&y.&m);
for (i=0;i<n;i++)
if (avto[i].year>=y&&avto[i].money<=m)
printf("\n%12s %4d %7.3f", avto[i].marka, avto[i].year, avto[i].money);
}
```

Пример. Программа, реализующая ввод списка студентов и вывод информации о студенте по его номеру в списке. Ввод информации о студенте реализован в виде отдельной функции.

```
#include<stdio.h>
const int n=3;
typedef struct{
 int num;
char fam[20];
char name[15];
}student;
student mas[n];
student *ptr=&mas[0];
void vvod(student *p)
{
scanf("%d",&p->num);
scanf("%s",p->fam);
scanf("%s",p->name);
}
void main()
{
int i,d;
for(i=0;i<n;i++)
vvod(ptr+i);
printf("Введите номер студента в списке\n");
scanf("%d",&d);
for(i=0;i<n;i++)
 if (mas[i].num==d)
printf(" Студент %s %s\n",mas[i].fam,mas[i].name);
}
```

Процедуре vvod передается значение указателя на элемент массива mas, который является структурой. В приведенной программе использованы оба способа обращения к элементу структуры (с использованием указателя и без него).

Контрольные вопросы

1. Что такое запись?
2. Где используется понятие записи?
3. С чего начинается объявление структуры?
4. Способы описания структуры.
5. Способы обращения к элементу структуры.
6. Описатель типа typedef.
7. Как осуществляться доступ к элементам структур с помощью указателей?
8. Описание указателя на структуру.
9. Описание массива, элементами которого являются структуры.
10. Две способы обращения к элементу структуры.

7.2 Варианты заданий

№1. Состав студентов факультета с разбивкой на группы. Количество специальностей на факультете, групп каждой специальности и студентов в каждой группе задать самостоятельно. Составить модуль поиска адреса (ссылки) элемента списка по его информационным полям.

№2. Состав специальностей вуза с разбивкой на факультеты. Количества факультетов и специальностей каждого факультета задать самостоятельно. Составить модуль поиска элементов с диапазоном шифров специальностей от ШИФР1 до ШИФР2.

№3. Вывести фамилии, имена и отчества всех студентов мужского пола, старших 18 лет.

№4. Вывести фамилии, имена и отчества студентов мужского пола, не сдавших более двух экзаменов.

№5. Вывести фамилии студентов женского пола, имеющих "5" по информатике.

№6. Вывести фамилии и средние баллы студентов, имеющих "5" по информатике.

№7. Подсчитать число студентов, которые моложе 16 лет, и вывести все данные о них.

№8. Список ПО на лазерных дисках в вычислительном центре с разбивкой по операционным системам. Количество ОС и дисков для каждой системы задать самостоятельно. Составить модуль поиска элементов списка по его информационным полям.

№9. Состав спортивного соревнования с разбивкой по видам спорта. Количества видов спорта и участников каждого вида спорта задать самостоятельно. Составить модуль формирования нового списка спортсменов, содержащего фамилии каждого второго спортсмена.

№10. Программа работы конференции с разбивкой докладов по секциям. Количество секций и докладов в каждой секции задать самостоятельно. Составить модуль формирования нового списка, содержащего доклады с несколькими авторами.

№11. Конструкция технического устройства, состоящего из нескольких блоков. Причем каждый блок содержит определенное количество модулей. Составить модуль формирования нового списка, из которого удаляется модуль с заданным наименованием.

№12. Конструкция модуля, состоящая из микросхем. Количества микросхем и вводов-выводов каждой микросхемы задать самостоятельно.

Составить модуль формирования нового списка вводов-выводов, из которого удаляются элементы с номерами 7 и14.

№13. Конструкция вычислительной сети, состоящей из узлов разных типов. Количества узлов и типов узлов задать самостоятельно. Составить модуль формирования нового списка узлов, в котором добавлен новый заданный узел.

№14. Книга состоит из глав и параграфов. Количества глав и параграфов каждой главы задать самостоятельно. Составить модуль формирования нового списка параграфов, в котором первый и последний элементы поменялись местами.

№15. Задачники состоят из списка задач, разбитых на темы. Количества тем и задач задать самостоятельно. Составить модуль формирования нового списка задач, в котором после задачи с заданным номером записан элемент с новой задачей.

№16. Набор материалов, имеющихся на складе с разбивкой по виду товара.

Количества видов товаров и штук каждого товара задать самостоятельно. Составить модуль формирования двух новых списков из исходного списка товаров. В первый список попадают номера товаров меньше некоторого заданного, во второй больше.

№17. Список школьников, занимающихся в кружках, с разбивкой по кружкам. Количества кружков и школьников в каждом кружке задать самостоятельно. Составить модуль формирования двух новых списков из списка школьников по следующему принципу: в первый список попадают первые десять школьников, во второй -остальные.

№18. Список покупок, сделанных кем-то в течении месяца, с разбивкой по статьям расходов. Количества статей расхода и покупок по каждой статье задать самостоятельно. Составить модуль формирования нового списка покупок с ценой больше заданной ЦЕНЫ.

№19. Список кварталов города с разбивкой по районам. Количества районов и кварталов в каждом районе задать самостоятельно. Составить модуль формирования нового списка кварталов, сводного для двух указанных районов.

№20. Список учебных дисциплин, которые должен изучить студент за время обучения в вузе с разбивкой на циклы. Количества циклов и дисциплин в циклах для разных специальностей задать самостоятельно. Составить модуль формирования нового списка, полученного слиянием трех списков дисциплин для заданных циклов.

№21. Список деталей механического устройства с разбивкой деталей по агрегатам. Количества агрегатов и деталей в каждом агрегате задать самостоятельно. Составить модуль формирования нового списка деталей, общих для двух заданных агрегатов.

№22. Список номеров автомобилей, паркующихся на платной стоянке с разбивкой по маркам. Количества марок и автомобилей каждой марки задать самостоятельно. Составить модуль формирования нового списка, содержащего номера с заданной частью номера.

№23. Список предприятий, расположенных в городе, с разбивкой по министерствам. Количества министерств и предприятий по каждому министерству задать самостоятельно. Составить модуль формирования нового списка предприятий, являющихся заводами.

№24. Список вузов страны с разбивкой по профилям. Количества профилей и вузов каждого профиля задать самостоятельно. Составить модуль формирования нового списка вузов заданного профиля, являющихся университетами.

№25. Список сотрудников предприятия с разбивкой по должностям. Количество должностей и количество сотрудников, имеющих одинаковую должность задать самостоятельно. Составить модуль формирования списка однофамильцев, работающих на одной должности.

№ 26 Список внутренних телефонов организации с разбивкой по отделам. Количества отделов и телефонов внутри отдела задать самостоятельно. Составить модуль поиска всех телефонов с заданными двумя первыми цифрами.

№ 27 Список участков предприятия с разбивкой по цехам. Количества цехов и участков каждого цеха задать самостоятельно. Составить модуль поиска участка с максимальным номером.

№ 28 Список работников цеха с разбивкой по профессиям. Количества профессий и работников каждой профессии цеха задать самостоятельно. Составить модуль сортировки фамилий по алфавиту.

№ 29 Состав парка ЭВМ вычислительного центра с разбивкой ЭВМ по сериям. Количество серий и ЭВМ разных серий задать самостоятельно. Составить модуль сортировки ЭВМ по возрастанию объема оперативной памяти.

№ 30 Список номеров и маршрутов рейсов автобусов с разбивкой рейсов по районам следования. Количество районов и рейсов в каждый район задать самостоятельно. Составить модуль сортировки рейсов по убыванию номеров.

8. Лабораторная работа на тему:
Обработка списков

Цель работы: приобрести практические навыки работы с динамическими структурами данных на языке Си.

8.1 Методические указания к выполнению работы

Список -это множество элементов, каждый из которых состоит, как минимум, из двух полей. Одно поле содержит либо саму информацию, либо ссылку на нее. Другое поле содержит ссылку на следующий элемент списка. Элемент списка называют "звено" списка. Таким образом, список - это цепочка связанных звеньев от

первого до последнего. Последнее звено не ссылается на следующий элемент, поэтому поле ссылки имеет значение "пустой указатель". По однонаправленному списку можно двигаться только в одном направлении -от заглавного (первого) звена к последнему.

Двунаправленный (двусвязный) список -это множество элементов, каждый из которых имеет два поля с указателями; одно поле содержит ссылку на следующий элемент, другое поле -ссылку на предыдущий элемент, и информационное поле.

Наличие ссылок как на следующее звено, так и на предыдущее позволяет от каждого звена двигаться по списку в любом направлении.

Список, первое звено которого имеет ссылку на последнее, а последнее на первое называется кольцевым. Кольцевой список имеет заглавное звено. Заглавное звено, как и в случае однонаправленного списка, позволяет обрабатывать первое и последнее звенья в общем цикле. Однако в таком кольцевом списке надо каждый раз проверять, не является ли очередное звено заглавным. Характерной особенностью динамических данных является невозможность установить заранее фиксированный объем памяти. Выделение памяти под отдельное звено списка происходит в тот момент, когда она появляется во время выполнения программы, а не во время трансляции.

Управление оперативной памятью реализуется посредством библиотечных функций, содержащихся в файле <alloc.h>: malloc, calloc, free. Функции malloc и calloc осуществляют выделение памяти, а функция free -ее освобождение. Функция malloc(unsigned size) возвращает в качестве своего значения указатель на область памяти. Размер этой области в байтах определяется с помощью аргумента size. Если выделение объема памяти невозможно, то результатом функции будет пустой указатель.

Функция calloc(unsigned count, size) выделяет обнуленную область памяти, в которой можно разместить count объектов размером size каждый. Результат функции -указатель на выделенную область памяти.

Функция free(ptr) освобождает область памяти, выделенную ранее, определяемую с помощью аргумента ptr.

Пример. Программа, создающая линейный однонаправленный список из фамилий студентов.

```
#include<alloc.h>
#include<stdio.h>
void main()
{ typedef struct man { char name[20];
 man *next; } man;
man *first, *cur;
int n;
printf("введите кол-во имен ");
scanf("%d",&n);
first=(man *)malloc(sizeof(man));
cur=first;
for(int i=0;i<n;i++)
{ if (i) { (*cur).next=(man *)malloc(sizeof(man));
 cur=(*cur).next; }
printf("введите имя ");
scanf("%s",(*cur).name);
(*cur).next=NULL;
 }
/* просмотр и вывод */
cur=first;
while (cur!=NULL)
 {printf("Это %s\n",(*cur).name);
cur=(*cur).next; }
}
```

Данная программа вначале запрашивает количество имен в списке, а затем формирует его, заполняя информационное поле. Наиболее рациональной является программа, где формирование списка происходит в зависимости от ответа на запрос о прекращении ввода информационных полей. Такая программа приведена ниже.

```
#include<alloc.h>
#include<stdio.h>
void main()
{ typedef struct man { char name[20];
 man *next;
 } man;
man *first, *cur;
int c;
```

```
first=(man *)malloc(sizeof(man));
cur=first;
 printf("введите имя ");
scanf("%s",(*cur).name);
(*cur).next=NULL;
 printf("введете еще 1/0\n");
scanf("%d",&c);
while(c==1)
{
(*cur).next=(man *)malloc(sizeof(man));
cur=(*cur).next;
printf("введите имя ");
scanf("%ls",(*cur).name);
(*cur).next=NULL;
printf("введете еще 1/0\n");
scanf("%d",&c);
}
/* просмотр и вывод */
cur=first;
while (cur!=NULL)
{printf("Это %s\n",(*cur).name);
cur=(*cur).next; }
}
```

Контрольные вопросы

1. Что такое динамические данные?
2. Что такое список?
3. Сколько имеется виды списков?
4. Функции управления оперативной памятью.
5. Минимум сколько поля имеет список?
6. Что такое информационное поле?
7. Понятие однонаправленной список.
8. Понятие двунаправленный список.
9. Функция malloc(unsigned size).
10. Функция calloc(unsigned count, size).

8.2 Варианты заданий

№1. Состав студентов факультета с разбивкой на группы. Количество специальностей на факультете, групп каждой специальности и студентов в каждой группе переменно. Для фрагмента модели, являющегося одномерным списком, составить модуль поиска адреса (ссылки) элемента списка по его информационным полям.

№2. Состав специальностей вуза с разбивкой на факультеты. Количества факультетов и специальностей каждого факультета переменны. Для фрагмента модели, являющегося одномерным списком, составить модуль поиска элементов с диапазоном шифров специальностей от ШИФР1 до ШИФР2.

№3. Список внутренних телефонов организации с разбивкой по отделам. Количества отделов и телефонов внутри отдела переменны. Для фрагмента модели, являющегося одномерным списком, составить модуль поиска всех телефонов с заданными двумя первыми цифрами.

№4. Список участков предприятия с разбивкой по цехам. Количества цехов и участков каждого цеха переменны. Для фрагмента модели, являющегося одномерным списком, составить модуль поиска участка с максимальным номером.

№5. Список работников цеха с разбивкой по профессиям. Количества профессий и работников каждой профессии цеха переменны. Для фрагмента модели, являющегося одномерным списком, составить модуль сортировки фамилий по алфавиту.

№6. Состав парка ЭВМ вычислительного центра с разбивкой ЭВМ по сериям. Количества серий и ЭВМ разных серий переменны. Для фрагмента модели, являющегося одномерным списком, составить модуль сортировки ЭВМ по возрастанию объема оперативной памяти.

№7. Список номеров и маршрутов рейсов автобусов с разбивкой рейсов по районам следования. Количества районов и рейсов в каждый район переменны. Для фрагмента модели, являющегося одномерным списком, составить модуль сортировки рейсов по убыванию номеров.

№8. Список ПО на лазерных дисках в вычислительном центре с разбивкой по операционным системам. Количества ОС и дисков для каждой системы переменны. Для фрагмента модели, являющегося одномерным списком, составить модуль поиска элементов списка по его информационным полям.

№9. Состав спортивного соревнования с разбивкой по видам спорта. Количества видов спорта и участников каждого вида спорта переменны. Для фрагмента модели, являющегося одномерным списком, составить модуль формирования нового списка спортсменов, содержащего фамилии каждого второго спортсмена.

№10. Программа работы конференции с разбивкой докладов по секциям. Количества секций и докладов в каждой секции переменны. Для фрагмента модели, являющегося одномерным списком, составить модуль формирования нового списка, содержащего доклады с несколькими авторами.

№11. Конструкция технического устройства, состоящего из нескольких блоков. Причем каждый блок содержит произвольное количество модулей. Для фрагмента модели, являющегося одномерным списком, составить модуль формирования нового списка, из которого удаляется модуль с заданным наименованием.

№12. Конструкция модуля, состоящая из микросхем. Количества микросхем и вводов-выводов каждой микросхемы переменны. Для фрагмента модели, являющегося одномерным списком, составить модуль формирования нового списка вводов-выводов, из которого удаляются элементы с номерами 7 и14.

№13. Конструкция вычислительной сети, состоящей из узлов разных типов. Количества узлов и типов узлов переменны. Для фрагмента модели, являющегося одномерным списком, составить модуль формирования нового списка узлов, в котором добавлен новый заданный узел.

№14. Книга состоит из глав и параграфов. Количества глав и параграфов каждой главы переменны. Для фрагмента модели, являющегося одномерным списком, составить модуль формирования нового списка параграфов, в котором первый и последний элементы поменялись местами.

№15. Задачники состоят из списка задач, разбитых на темы. Количества тем и задач переменны. Для фрагмента модели, являющегося одномерным списком, составить модуль формирования нового списка задач, в котором после задачи с заданным номером записан элемент с новой задачей.

№16. Набор материалов, имеющихся на складе, с разбивкой по виду товара. Количества видов товаров и штук каждого товара переменны. Для фрагмента модели, являющегося одномерным списком, составить модуль формирования двух новых списков из исходного списка товаров. В первый список попадают номера товаров меньше некоторого заданного, во второй -больше.

№17. Список школьников, занимающихся в кружках, с разбивкой по кружкам. Количества кружков и школьников в каждом кружке переменны. Для фрагмента модели, являющегося одномерным списком, составить модуль формирования двух новых списков из списка школьников по следующему принципу: в первый список попадают первые десять школьников, во второй -остальные.

№18. Список покупок, сделанных кем-то в течении месяца, с разбивкой по статьям расходов. Количества статей расхода и покупок по каждой статье переменны. Для фрагмента модели, являющегося

одномерным списком, составить модуль формирования нового списка покупок с ценой больше заданной ЦЕНЫ.

№19. Список кварталов города с разбивкой по районам. Количества районов и кварталов в каждом районе переменны. Для фрагмента модели, являющегося одномерным списком, составить модуль формирования нового списка кварталов, сводного для двух указанных районов.

№20. Список учебных дисциплин, которые должен изучить студент за время обучения в вузе, с разбивкой на циклы (аппаратный, общенаучный, гуманитарный и т.д.). Количества циклов и дисциплин в циклах для разных специальностей переменны. Для фрагмента модели, являющегося одномерным списком, составить модуль формирования нового списка, полученного слиянием трех списков дисциплин для заданных циклов.

№21. Список деталей механического устройства с разбивкой деталей по агрегатам. Количества агрегатов и деталей в каждом агрегате переменны. Для фрагмента модели, являющегося одномерным списком, составить модуль формирования нового списка деталей, общих для двух заданных агрегатов.

№22. Список номеров автомобилей, паркующихся на платной стоянке, с разбивкой по маркам. Количества марок и автомобилей каждой марки переменны. Для фрагмента модели, являющегося одномерным списком, составить модуль формирования нового списка, содержащего номера с заданной частью номера.

№23. Список предприятий, расположенных в городе, с разбивкой по министерствам. Количества министерств и предприятий по каждому министерству переменны. Для фрагмента модели, являющегося одномерным списком, составить модуль формирования нового списка предприятий, являющихся заводами.

№24. Список вузов страны с разбивкой по профилям. Количества профилей и вузов каждого профиля переменны. Для фрагмента модели, являющегося одномерным списком, составить модуль формирования нового списка вузов заданного профиля являющихся университетами.

№25. Список сотрудников предприятия с разбивкой по должностям. Количества должностей и количество сотрудников, имеющих одинаковую должность, переменны. Для фрагмента модели, являющегося одномерным списком, составить модуль формирования списка однофамильцев, работающих на одной должности.

№26. Составить программу, которая содержит информацию о наличии автобусов в автобусном парке.

Сведения о каждом автобусе содержат: номер автобуса, фамилию и инициалы водителя, номер маршрута.

№27. Составить программу, которая содержит текущую информацию о книгах в библиотеке. Сведения о книгах содержат: номер УДК, фамилию и инициалы автора, название, год издания, количество экземпляров данной книги в библиотеке.

№28. Составить программу, которая содержит текущую информацию о заявках на авиабилеты. Каждая заявка содержит : пункт назначения, номер рейса, фамилию и инициалы пассажира, желаемую дату вылета.

№29. Картотека в бюро обмена квартир организации как линейный список. Сведения о каждой квартире содержат: количество комнат, этаж, площадь, адрес.

№30. Анкета для опроса населения содержит две группы вопросов. Первая группа содержат сведения о респонденте: возраст, пол, образование (начальное. среднее, высшее).

Вторая группа содержит собственно вопрос анкеты, ответом на который может быть либо «Да», либо «Нет»

Составить программу, которая:

- обеспечивает начальный ввод анкет и формирует из них линейный список;
- на основе анализа анкет выдает ответы на следующие вопросы:

1) сколько мужчин старше 40 лет, имеющих высшее образование, ответили «Да» на вопрос анкеты;

2) сколько женщин моложе 30 лет, имеющих среднее образование, ответили «Нет» на вопрос анкеты;

3) сколько мужчин моложе 25 лет, имеющих начальное образование, ответили «Да» на вопрос анкеты;

- производит вывод всех анкет и ответов на вопросы.

9. Лабораторная работа на тему: Обработка файлы данных

Цель работы: познакомиться с организацией работы с файлами в языке Си а также приобрести практический опыт создание программ, использующих файлы, в частности таких программ, для которых исходные данные поступают из файла, а результаты работы записываются в файл.

9.1 Методические указания к выполнению работы

Файлом называют способ хранения информации на физическом устройстве. Файл - это понятие, которое применимо ко всему - от файла на диске до терминала.

В языке Си отсутствуют операторы для работы с файлами. Все

необходимые действия выполняются с помощью функций, включенных в стандартную библиотеку. Они позволяют работать с различными устройствами, такими, как диски, принтер, коммуникационные каналы и т.д. Эти устройства сильно отличаются друг от друга. Однако файловая система преобразует их в единое *абстрактное логическое устройство*, называемое *потоком*. В Си существует два типа потоков: текстовые (text) и двоичные (binary).

Текстовый поток - это последовательность символов. При передаче символов из потока на экран, часть из них не выводится (например, символ возврата каретки, перевода строки).

Двоичный поток - это последовательность байтов, которые однозначно соответствуют тому, что находится на внешнем устройстве.

Прежде чем читать или записывать информацию в файл, он должен быть открыт и тем самым связан с потоком. Это можно сделать с помощью библиотечной функции fopen( ). Она берет внешнее представление файла (например, c:\my_prog.txt) и связывает его с внутренним логическим именем, которое используется далее в программе. Логическое имя - это указатель на требуемый файл. Его необходимо определить; делается это, например, так:

```
FILE *fp;
```

Здесь FILE - имя типа, описанное в стандартном заголовочном файле stdio.h, fp - указатель на файл. Обращение к функции fopen( ) в программе осуществляется выражением:

```
fp = fopen(спецификация файла, "способ использования файла");
```

Спецификация файла (т.е. имя файла и путь к нему) может, например, иметь вид: "c:\\my_prog.txt" - для файла my_prog.txt на диске c:.

Способ использования файла задается следующими символами:

r - открыть существующий файл для чтения;
w - создать новый файл для записи (если файл с указанным именем существует, то он будет переписан);
a - дополнить файл (открыть существующий файл для записи информации, начиная с конца файла, или создать файл, если он не существует);
r+ - открыть существующий файл для чтения и записи;
w+ - создать новый файл для чтения и записи;
a+ - дополнить или создать файл с возможностью чтения и записи;
rb - открыть двоичный файл для чтения;
wb - создать двоичный файл для записи;
ab - дополнить двоичный файл;
r+b - открыть двоичный файл для чтения и записи;
w+b - создать двоичный файл для чтения и записи;
a+b - дополнить двоичный файл с предоставлением возможности чтения и записи;
rt - открыть текстовой файл для чтения;

wt - создать текстовый файл для записи;
at - дополнить текстовый файл;
r+t - открыть текстовой файл для чтения и записи;
w+t - создать текстовый файл для чтения и записи;
a+t - дополнить текстовый файл с предоставлением возможности записи и чтения.

Если режим t или b не задан (например, r, w или a), то он определяется значением глобальной переменной _fmode. Если fmode=0_BINARY, то файлы открываются в двоичном режиме, а если _fmode=0_TEXT - в текстовом режиме. Константы 0_BINARY и 0_TEXT определены в файле fcntl.h.

Строки вида r+b можно записывать и в другой форме: rb+.

Если в результате обращения к функции fopen( ) возникает ошибка, то она возвращает указатель на константу NULL.

Рекомендуется использовать следующий способ открытия файла:

```
 if ((fp = fopen("c:\\my_prog.txt", "rt")) == NULL)
{
puts("Открыть файл не удалось\n");
exit(1);
}
```

После окончания работы с файлом он должен быть закрыт. Это делается с помощью библиотечной функции fclose( ). Она имеет следующий прототип:

```
int fclose(FILE *fp);
```

При успешном завершении операции функция fclose( ) возвращает значение нуль. Любое другое значение свидетельствует об ошибке.

Рассмотрим другие библиотечные функции, используемые для работы с файлами (все они описаны в файле stdio.h):

1. Функция putc( ) записывает символ в файл и имеет следующий прототип:

```
int putc(int c, FILE *fp);
```

Здесь fp - указатель на файл, возвращенный функцией fopen( ), c - символ для записи (переменная c имеет тип int, но используется только младший байт). При успешном завершении putc( ) возвращает записанный символ, в противном случае возвращается константа EOF. Она определена в файлestdio.h и имеет значение -1.

2. Функция getc( ) читает символ из файла и имеет следующий прототип:

```
int getc(FILE *fp);
```

Здесь fp - указатель на файл, возвращенный функцией fopen( ). Эта функция возвращает прочитанный символ. Соответствующее значение

имеет тип int, но старший байт равен нулю. Если достигнут конец файла, то getc( ) возвращает значение EOF.

3. Функция feof( ) определяет конец файла при чтении двоичных данных и имеет следующий прототип:

int feof(FILE *fp);

Здесь fp - указатель на файл, возвращенный функцией fopen( ). При достижении конца файла возвращается ненулевое значение, в противном случае возвращается 0.

4. Функция fputs( ) записывает строку символов в файл. Она отличается от функции puts( ) только тем, что в качестве второго параметра должен быть записан указатель на переменную файлового типа.

Например:

fputs("Example", fp);

При возникновении ошибки возвращается значение EOF.

5. Функция fgets( ) читает строку символов из файла. Она отличается от функции gets( ) тем, что в качестве второго параметра должно быть указано максимальное число вводимых символов плюс единица, а в качестве третьего - указатель на переменную файлового типа. Строка считывается целиком, если ее длина не превышает указанного числа символов, в противном случае функция возвращает только заданное число символов.

Рассмотрим пример:

fgets(string, n, fp);

Функция возвращает указатель на строку string при успешном завершении и константу NULL в случае ошибки либо достижения конца файла.

6. Функция fprintf( ) выполняет те же действия, что и функция printf( ), но работает с файлом. Ее отличием является то, что в качестве первого параметра задается указатель на переменную файлового типа.

Например:

fprintf(fp, "%x",a);

7. Функция fscanf( ) выполняет те же действия, что и функция scanf(), но работает с файлом. Ее отличием является то, что в качестве первого параметра задается указатель на переменную файлового типа.

Например:

fscanf(fp, "%x", &a);

При достижении конца файла возвращается значение EOF.

8. Функция fseek( ) позволяет выполнять чтение и запись с произвольным доступом и имеет следующий прототип:

int fseek(FILE *fp, long count, int access);

Здесь fp - указатель на файл, возвращенный функцией fopen( ), count -

номер байта относительно заданной начальной позиции, начиная с которого будет выполняться операция, access - способ задания начальной позиции.

Переменная access может принимать следующие значения:

0 - начальная позиция задана в начале файла;

1 - начальная позиция считается текущей;

2 - начальная позиция задана в конце файла.

При успешном завершении возвращается нуль, при ошибке - ненулевое значение.

9. Функция ferror( ) позволяет проверить правильность выполнения последней операции при работе с файлами. Имеет следующий прототип:

int ferror(FILE *fp);

В случае ошибки возвращается ненулевое значение, в противном случае возвращается нуль.

10. Функция remove( ) удаляет файл и имеет следующий прототип:

int remove(char *file_name);

Здесь file_name - указатель на строку со спецификацией файла. При успешном завершении возвращается нуль, в противном случае возвращается ненулевое значение.

11. Функция rewind( ) устанавливает указатель текущей позиции в начало файла и имеет следующий прототип:

void rewind(FILE *fp);

12. Функция fread( ) предназначена для чтения блоков данных из потока. Имеет прототип:

unsigned fread(void *ptr, unsigned size, unsigned n, FILE *fp);

Она читает n элементов данных, длиной size байт каждый, из заданного входного потока fp в блок, на который указывает указатель ptr. Общее число прочитанных байтов равно произведению n*size. При успешном завершении функция fread( ) возвращает число прочитанных элементов данных, при ошибке - 0.

13. Функция fwrite( ) предназначена для записи в файл блоков данных. Имеет прототип:

unsigned fwrite(void *ptr, unsigned size, unsigned n, FILE *fp);

Она добавляет n элементов данных, длиной size байт каждый, в заданный выходной файл fp. Данные записываются с позиции, на которую указывает указатель ptr. При успешном завершении операции функция fwrite( ) возвращает число записанных элементов данных, при ошибке - неверное число элементов данных.

В языке Си имеются пять стандартных файлов со следующими логическими именами:

stdin - для ввода данных из стандартного входного потока (по умолчанию - с клавиатуры);

stdout - для вывода данных в стандартный выходной поток (по умолчанию - на экран дисплея);
stderr - файл для вывода сообщений об ошибках (всегда связан с экраном дисплея);
stdprn - для вывода данных на принтер;
stdaus - для ввода и вывода данных в коммуникационный канал.

В языке Си имеется также система низкоуровневого ввода/вывода (без буферизации и форматирования данных), соответствующая стандарту системы UNIX. Прототипы составляющих ее функций находятся в файле io.h. К этим функциям относятся:

open( ) - открыть файл;
close( ) - закрыть файл;
read( ) - читать данные;
write( ) - записать данные;
lseek( ) - поиск определенного байта в файле;
unlink( ) - уничтожить файл.

Контрольные вопросы

1.Что такое файлы?
2. Что называется потоком?
3. Типы потоков.
4. Библиотечной функции fopen( ).
5. Что такое логическое имя?
6. Стандартные файлы.
7. Режимы файлы.
8. Функции putc( ) и getc( ).
9. Функции fprintf( ) и fscanf( ).
10. Функции fseek( ) и feof( ).

9.2 Варианты заданий

№1. Дан файл f, компоненты которого являются целыми числами. Найти сумму компонент файла f.

№2. Дан файл g, компоненты которого являются целыми числами. Найти количество четных чисел среди компонент.

№3. Дан файл q, компоненты которого являются действительными числами. Найти наибольшее из значений компонент.

№4. Дан файл f (создать его), компоненты которого являются целыми числами. Никакая из компонент файла не равна 0. Файл f содержит равное число отрицательных и положительных чисел. Используя вспомогательный файл, переписать компоненты файла f в другой файл g так, чтобы в файле g числа шли сначала все положительные числа, а потом все отрицательные числа.

№5. Создать файл f, компоненты которого являются целыми числами. Никакая из компонент не равна 0. Числа в файле записаны в следующем порядке: 5 положительных, 5 отрицательных и т.д.Число компонентов файла f делится на 20. Переписать компоненты файла f в файл g так, чтобы в файле g числа шли в следующем порядке: ++ -- ++ -- …, т.е. два положительных числа, два отрицательных числа и т.д. Допустимо использование вспомогательного файла.

№6. Дан файл f (создать его), компоненты которого являются целые числа. Получить новый файл из данного исключением повторных вхождений одного и того же числа.

№7. Дан файл f (создать его), компоненты которого являются целые числа, не равные 0.Числа в файле записаны в следующем порядке: ++ -- ++ -- …, т.е. два положительных числа, два отрицательных числа и т.д. Числа компонентов файла f делится на 12. переписать компоненты файла f в файл h так, чтобы в файле h числа шли в следующем порядке: ++ -- ++ -- …, т.е. два положительных числа, два отрицательных числа и т.д. Допустимо использование вспомогательного файла.

№8. Дан файл f (создать его), компоненты которого являются целые числа. Число их кратно 5. записать в новый файл g наибольшее из первых 5 компонент файла f, затем – наибольшее из следующих 5 компонент и.т.д.

№9. Дан файл f (создать его), компоненты которого являются целыми числами. Записать в файл g все четные числа файла f, а в файл h – все нечетные числа файла f. Порядок следования чисел сохраняется.

№10. Дан символьный файл (создать его), содержащий слова. Слова в тексте разделяются пробелами и знаками препинания. Получить 2 наиболее часто встречающихся слова и число их повторений. Если таких слов более 2, выдать соответствующую информацию для каждого из них.

№11. Создать символьный файл s, содержащий слова разделенный пробелами. Удалить из файла все однобуквенные слова и лишние пробелы. Полученный файл записать в новый файл f.

Пример: Файл s: «стол и стул». Файл f: «стол стул».

№12. Дан символьный файл (создать его). Найти самое длинное слово среди слов, вторая буква которых есть ‘a’. Если таких слов с наибольшей длиной несколько, то найти все. Если таких слов нет, то сообщить об этом.

№13. Создать символьный файл. Подсчитать число вхождений в файл сочетаний ‘ab’ и ‘cd’. Если таких сочетаний нет, то сообщить об этом.

№14. Дан текстовый файл, содержащий программу на языке Паскаль (создать его). Проверить эту программу на несоответствие числа открывающихся и закрывающихся круглых и операторных скобок.

№15. Дан символьный файл (создать его). Файл состоит из слов, разделенных пробелами или знаками препинания. Определить количество слов в файле и распечатать первое и последнее слово.

№16. Даны (создать) два файла – упорядоченные по номерам телефонные справочники. Номер представляет собой 5 – значное число, начинающееся с цифры 1 или с цифры 2. Формат записи справочника: Телефон ФИО Адрес. Написать программу, строящую третий файл – общий упорядоченный справочник.

№17. Дан файл «Товар» (создать его), содержащий сведения об экспортируемых товарах: указывается наименование товара; страна, экспортирующая товар; объем поставляемой партии товара в штуках. Составить и записать в новый файл список стран, в которые экспортируется определенный товар, и найти общий объем экспорта этого товара. Исходный файл и полученный список распечатать.

№18. Создать два файла f1 и f2. Файл f1- это инвентарный файл, содержащий сведения о том, сколько изделий каких видов продукции хранится на складе. Файл f2 – это вспомогательный файл, содержащий сведения о том, на сколько уменьшилось или увеличилось количество изделий по некоторым видам продукции. Файл f2 может содержать несколько сообщений по продукции одного вида или не содержать ни одного сообщения по продукции какого-то вида. Обновить инвентарный файл f1 на основе вспомогательного файла f2, образовав новый файл f3. Все три файла распечатать.

№19. Багаж каждого пассажира характеризуется количеством его вещей и общим весом этих вещей. Создать файл «Багаж» содержащий сведения о багаже нескольких пассажиров. Упорядочить сведения записанные в файле «Багаж», по убыванию общего веса багажа одного пассажира. Предполагается, что число пассажиров равно N и не слишком велико, то есть упорядочивание можно сделать в оперативной памяти.

№20. Багаж каждого пассажира характеризуется количеством его вещей и общим весом этих вещей. Создать файл «Багаж», содержащий сведения о багаже нескольких пассажиров. Найти число пассажиров, имеющих более 2-х вещей, и число пассажиров, количество вещей у которых превосходит среднее число вещей.

№21. Создать файл «Кубики», содержащий сведения о кубиках: размер (длина ребра), цвет (допустимы 7 цветов радуги) и материал (дерево, металл, картон). Найти количество кубиков каждого из возможных цветов и их суммарный объем.

№22. Для каждого авиарейса в файле записано: номер рейса, название порта назначения, время отправления и прибытия, номера дней вылета.

Указан порт назначения. Напечатать и записать в другой файл номера рейсов и дни отправления рейсов до указанного порта.

№23. Сведения о студентах (ФИО, группа, оценки за контрольные работы по пятибалльной системе) содержатся в файле (создать его). Собрать в новый файл сведения о лучших студентах, не имеющих за все контрольные работы оценок ниже четырёх баллов.

№ 24. Дан файл f, компоненты которого являются действительными числами. Найти произведение компонент файла f.

№25 Создать файл f, компоненты которого являются целыми числами. Никакая из компонент файла не равна 0. Файл f содержит равное число отрицательных и положительных чисел. Число компонентов файла f делится на 4. Используя вспомогательный файл, переписать компоненты файла f в файл g так, чтобы файле g числа шли в следующем порядке: ++ -- ++ -- … , то есть два положительных числа, два отрицательных числа и т.д.

№26. Дан файл f (создать его), в котором количество положительных чисел равно количеству отрицательных чисел и нет чисел, равных 0. Создать новый файл, в котором знаки чисел чередуются: положительное число, отрицательное число и т.д. Допустимо использовать вспомогательный файл.

№27. Дан файл g, компоненты которого являются целыми числами. Получить в файле t все компоненты файла g делящиеся на 3 и не делящиеся на 7.

№28. Дан символьный файл z. Получить файл s, образованный из файла z заменой всех его прописных (больших) букв одноименными строчными (малыми).

№29. Дан файл f, компоненты которого являются целыми числами. Записать в файл g все четные числа файла f, а в файл h – все нечетные. Порядок следования чисел сохраняется.

№30. Дан символьный файл h. Записать в файл d компоненты файла h в обратном порядке.

10. Лабораторная работа на тему: Подпрограммы в Си

Цель работы: познакомиться с основными правилами оформления подпрограмм(функций) в языке Си а также приобрести практический опыт работы с подпрограммами.

10.1 Методические указания к выполнению работы

Программы на языке Си обычно состоят из большого числа

отдельных функций (подпрограмм). Подпрограммы, или функции, представляют собой важный инструмент языка Си. Они позволяют писать хорошо структурированные модульные программы. Как правило, эти функции имеют небольшие размеры и могут находиться как в одном, так и в нескольких файлах. Все функции являются глобальными. В языке запрещено определять одну функцию внутри другой. Связь между функциями осуществляется через аргументы, возвращаемые значения и внешние переменные.

В общем случае функции в языке Си необходимо объявлять. Объявление функции (т.е. описание заголовка) должно предшествовать ее использованию, а определение функции (т.е. полное описание) может быть помещено как после тела программы (т.е. функции main ( ), так и до него. Если функция определена до тела программы, а также до ее вызовов из определений других функций, то объявление может отсутствовать. Как уже отмечалось, описание заголовка функции обычно называют прототипом функции.

Функция объявляется следующим образом:

тип имя_функции(тип имя_параметра_1, тип имя_параметра_2, ...);

Тип функции определяет тип значения, которое возвращает функция. Если тип не указан, то предполагается, что функция возвращает целое значение (int).

При объявлении функции для каждого ее параметра можно указать только его тип (например: тип функция (int, float, ...), а можно дать и его имя (например: тип функция (int a, float b, ...)).

В языке Си разрешается создавать функции с переменным числом параметров. Тогда при задании прототипа вместо последнего из них указывается многоточие.

Определение функции имеет следующий вид:

```
тип имя_функции(тип имя_параметра_1, тип имя_параметра_2,...)
{
тело функции
}
```

Передача значения из вызванной функции в вызвавшую происходит с помощью оператора возврата return, который записывается следующим образом:

return выражение;

Таких операторов в подпрограмме может быть несколько, и тогда они фиксируют соответствующие точки выхода. Например:

```
int f(int a, int b)
{
```

```
if (a > b) { printf("max = %d\n", a); return a; }
printf("max = %d\n", b); return b;
}
```

Вызвать эту функцию можно следующим образом:

```
c = f(15, 5);
c = f(d, g);
f(d, g);
```

Вызвавшая функция может, при необходимости, игнорировать возвращаемое значение. После слова return можно ничего не записывать; в этом случае вызвавшей функции никакого значения не передается. Управление передается вызвавшей функции и в случае выхода "по концу" (последняя закрывающая фигурная скобка).

В языке Си аргументы функции передаются по значению, т.е. вызванная функция получает свою временную копию каждого аргумента, а не его адрес. Это означает, что вызванная функция не может изменить значение переменной вызвавшей ее программы. Однако это легко сделать, если передавать в функцию не переменные, а их адреса.

Например:

```
void swap(int *a, int *b)
{
int *tmp = *a;
*a = *b;
*b = *tmp;
}
```

Вызов swap(&b, &c) (здесь подпрограмме передаются адреса переменных b и c) приведет к тому, что значения переменных b и c поменяются местами.

Если же в качестве аргумента функции используется имя массива, то передается только адрес начала массива, а сами элементы не копируются. Функция может изменять элементы массива, сдвигаясь (индексированием) от его начала.

Рассмотрим, как функции можно передать массив в виде параметра. Здесь возможны три варианта:

Параметр задается как массив (например: int m[100];).

Параметр задается как массив без указания его размерности (например: int m[];).

Параметр задается как указатель (например: int *m;). Этот вариант используется наиболее часто.

Независимо от выбранного варианта вызванной функции передается указатель на начало массива. Сами же элементы массива не копируются.

Если некоторые переменные, константы, массивы, структуры объявлены как глобальные, то их не надо включать в список параметров вызванной функции.

Контрольные вопросы

1.Что такое подпрограммы?
2. В каком виде используется подпрограмма в Си?
3. Что такое модульное программирование?
4. Можно определять одну функцию внутри другой?
5. Как объявляется функция?
6. Какие вид имеет определение функции?
7. Значение оператора возврата.
8. Как вызывается функция?
9. Передача входных и выходных параметров.
10. Передача массивов в качестве параметров.

10.2 Варианты заданий

1)Если обрабатываются массивы чисел, то ввод и вывод массивов организовать в подпрограммах.

2)При организации программы все исходные данные задавать самостоятельно.

№1. Для матрицы A (4 строки, 5 столбцов) вычислить выражение $Y=M_1M_5 +M_2M_4+\ldots+M_5M_1)$, где M_k — значение наибольшего элемента k - м столбце этой матрицы. Для поиска наибольшего элемента в произвольном столбце матрицы использовать подпрограммы(функций).

№2. Для матрицы B (5 строк, 4 столбца) вычислить выражение $Y=5M_1+ 4M_2 +...+ 1M_5$, где M_k— значение наименьшего элемента в k-й строке этой матрицы. Для поиска наименьшего элемента в произвольной строке матрицы использовать подпрограммы(функций).

№3. Для матрицы B (5 строк, 4 столбца) вычислить выражение $y=(C_1-1)^5+(C_2-1)^4+\ldots+(C_5-1)$, где C_k — среднее арифметическое значение элементов в k-ой строке матрицы B, значения которых превышают значение первого элемента в данной строке. Для вычисления среднего арифметического значения указанных элементов в произвольной строке матрицы использовать подпрограммы (функций).

№4. Для матрицы A (4 строки, 5 столбцов) вычислить выражение $Y=(1-P_5)^2+(2-P_4)^2+\ldots+(5-P_1)^2$, где P_k— произведение отрицательных элементов в k-м столбце матрицы A. Для вычисления произведения отрицательных элементов в произвольном столбце матрицы использовать подпрограммы(функций).

№5. Для матрицы A (4 строки, 4 столбца) вывести те столбцы, в которых есть нулевые элементы, или вывести сообщение "нет столбцов с нулями". Для проверки наличия нулевых элементов в произвольном столбце матрицы использовать подпрограммы(функций).

№6. Для матрицы A (4 строки, 5 столбцов) вывести те строки, в которых есть нулевые элементы, или вывести сообщение "нет строк с нулями". Для проверки наличия нулевых элементов в произвольной строке матрицы использовать подпрограммы(функций).

№7. Для матрицы A (5 строк, 4 столбца) определить, в какой из ее половин (левой или правой) больше нулевых элементов (вывести одно из сообщений: "больше в левой", "больше в правой" или "одинаково"). Для подсчета количества нулевых элементов в произвольной части матрицы использовать подпрограммы(функций).

№8. Для матрицы A (4 строки, 5 столбцов) определить, в какой из ее половин (верхней или нижней) больше нулевых элементов (вывести одно из сообщений: "больше в верхней", "больше в нижней" или "одинаково"). Для подсчета количества нулевых' элементов в произвольной части матрицы использовать подпрограммы(функций).

№9. Для двух матриц: A (3 строки, 6 столбцов) и B (3 строки, 3 столбца) определить, в какой из них больше нулевых элементов. Вывести одно из сообщений: "больше в A", "больше в B" или "одинаково". Для подсчета количества нулевых элементов в матрице использовать подпрограммы(функций).

№10. В матрице A (4 строки, 3 столбца) поменять местами наибольшие элементы в первом и третьем столбцах. Для поиска номера наибольшего элемента в произвольном столбце матрицы использовать подпрограммы(функций).

№11. В матрице A (3 строки, 5 столбцов) поменять местами наименьшие элементы во второй и третьей строке. Для поиска номера наименьшего элемента в произвольной строке матрицы использовать подпрограммы(функций).

№12. В матрице A (4 строки, 5 столбцов) поменять местами наибольшие элементы в ее верхней и нижней половинах. Для поиска индексов наибольшего элемента в произвольной части матрицы использовать подпрограммы(функций).

№13. В матрице A (3 строки, 4 столбца) поменять местами наименьшие элементы в ее левой я правой половинах. Для поиска индексов наименьшего элемента в произвольной части матрицы использовать подпрограммы(функций).

№14 В двух матрицах: A (3 строки, 3 столбца) и B (3 строки, 4 столбца) поменять местами наибольшие элементы. Для поиска индексов наибольшего элемента в матрице использовать подпрограммы(функций).

№15. В двух матрицах: A (3 строки, 3 столбца) и B (4 строки, 4 столбца) поменять местами наименьшие элементы в их главных диагоналях. Для поиска индексов наименьшего элемента в главной диагонали матрицы использовать подпрограммы(функций).

№16. В каждой из матриц: A (3 строки, 4 столбца) и B (5 строк, 5 столбцов) поменять местами две строки: в матрице A—строки 2 и 3, в матрице B — строки 2 и 4. Для обмена в матрице строк с произвольными номерами использовать подпрограммы(функций).

№17. В каждой из матриц: A (5 строк, 4 столбца) и B (4 строки, 3 столбца) поменять местами два столбца: в матрице A — столбцы 2 и 4, в матрице B — столбцы 2 и 3. Для обмена в матрице столбцов с произвольными номерами использовать подпрограммы(функций).

№18. На основе матрицы A (4 строки, 5 столбцов) сформировать одномерный массив B из 5 элементов. Каждый элемент B_k этого массива получает значение, равное полу сумме значений наибольшего и наименьшего элементов в k-м столбце матрицы A.Для вычисления полу суммы значений наибольшего и наименьшего элементов в произвольном столбце матрицы использовать подпрограммы(функций).

№19. На основе матрицы A (4 строки, 4 столбца) сформировать одномерный массив B из 4 элементов, Каждый элемент B_k этого массива получает значение, равное абсолютной величине разности значений элемента главной диагонали и наименьшего элемента в k-ой строке матрицы A. Для вычисления значения указанной разности в произвольной строке матрицы использовать подпрограммы(функций).

№20 Для каждой из двух матриц: A (2 строки, 4 столбца) и B (3 строки, 3 столбца) определить номер того столбца, в котором среднее арифметическое значений его элементов минимально. Для поиска номера указанного столбца в произвольной матрице использовать подпрограммы(функций).

№21 На основе матрицы A (4 строки, 5 столбцов) сформировать одномерный массив B из 5 элементов, каждый элемент которого принимает значение 1,если в соответствующем по номеру столбце матрицы A элементы расположены в порядке возрастания их значений, и значение 0 в противном случае. Для проверки упорядоченности элементов по возрастанию в произвольном столбце матрицы использовать подпрограммы(функций).

№22. Для матрицы A (4 строки, 4 столбца) сформировать одномерный массив B из 4 элементов. Каждый элемент B_k получает значение 0, если в k-м столбце матрицы A есть хотя бы один нулевой элемент, и значение 1 в противном случае. Для проверки наличия (отсутствия) нулевых элементов в произвольном столбце матрицы использовать подпрограммы(функций).

№23. Для матрицы A (4 строки, 4 столбца) сформировать одномерный массив B из 4 элементов. Каждый элемент B_k получает значение, равное номеру максимального элемента в k-м столбце матрицы A. Для поиска номера максимального элемента в произвольном столбце матрицы использовать подпрограммы(функций).

№24. В матрице A (5 строк, 4 столбца) определить номер столбца с минимальным значением среднего арифметического значения отрицательных элементов столбца. Для вычисления среднего арифметического значения отрицательных элементов в произвольном столбце матрицы использовать подпрограммы(функций).

№25. Для каждой из двух матриц: A (3 строки, 4 столбца) и B (5 строк, 3 столбца) определить номер строки с максимальной суммой положительных элементов. Для поиска номера такой строки в произвольной матрице использовать подпрограммы(функций).

№26. Для матрицы A (5 строк и 5 столбцов) сформировать одномерный массив B из 5 элементов. Каждый элемент B_k принимает значение 1, если в k-ой строке матрицы A есть элементы, превышающие значение соответствующего элемента главной, диагонали, и значение 0 в противном случае. Для проверки наличия элементов, превышающих значение элемента главной диагонали, в произвольной строке матрицы использовать подпрограммы(функций).

№27. Для матрицы A (5 строк, 5 столбцов) сформировать одномерный массив B из 5 элементов. Каждый элемент B_k принимает значение 1, если все элементы k-го столбца матрицы A не превышают значения соответствующего элемента главной диагонали, и значение 0 в противном случае. Для проверки отсутствия элементов, превышающих значение элемента главной диагонали, в произвольном столбце матрицы использовать подпрограммы(функций).

№28. В матрице A (4 строки, 5 столбцов) для каждой ее половины (верхней и нижней) вычислить и вывести количество положительных (больших 0) элементов. Также определить, в какой из этих половин среднее арифметическое значение прочих элементов имеет большую величину (вывести одно из сообщений: "больше в верхней", "больше в нижней" или "одинаково"). Для вычисления количества положительных элементов и среднего арифметического значения прочих элементов в произвольной части матрицы использовать подпрограммы(функций).

№29 В матрице А (3 строки, 4 столбца) для каждой ее половины (левой и правой) вычислить и вывести количество нулевых элементов. Также определить, в какой из этих половин среднее арифметическое значение прочих элементов имеет меньшую величину (вывести одно из сообщений: "меньше в левой", "меньше в правой" или "одинаково"). Для вычисления количества нулевых элементов и среднего арифметического значения прочих элементов в произвольной части матрицы использовать подпрограммы(функций).

№30 Для каждой из двух матриц: А (2 строки, 4 столбца) и В (3 строки, 3 столбца) вычислить и вывести количество отрицательных элементов. Также определить, в какой из них среднее - арифметическое значение прочих элементов имеет большую величину (вывести одно из сообщений: "больше в А", "больше в В" или "одинаково"). Для вычисления количества отрицательных элементов и среднего арифметического значения прочих элементов матрицы использовать подпрограммы(функций).

ПРИЛОЖЕНИЕ 1
Некоторые ошибки времени компиляции

(expected - (Ожидается
) expected -) Ожидается
, expected - , Ожидается
: expected - : Ожидается
< expected - < Ожидается
{ expected - { Ожидается
} expected - } Ожидается
Array bounds missing -] Отсутствие границ Массива
Array must have at least one element - Массив должен иметь по крайней мере один элемент
Array size too large - Размер Массива слишком большой
Bit field cannot be static - Битовое поле не может быть статическим
Bit field too large - Битовое поле слишком большое
Bit fields must be signed or unsigned int - Битовое поле должно быть знаковым или беззнаковым
Bit fields must contain at least one bit - Битовое поле должно содержать по крайней мере один бит
Call of nonfunction - Вызов не функции
Cannot allocate a reference - Не может разместить ссылку
Cannot call 'main' from within the program - Не может вызывать " main " изнутри программы
Cannot convert 'type1' to 'type2' - Не может конвертировать " type1 " к " type2 "
Case outside of switch - Выбор вне оператора-переключателя
Constant expression required - Требуется константное выражение
Could not find file 'filename' - Не может найти файл " filename "
Declaration syntax error - Ошибка описания
Declaration was expected - Ожидается описание
Division by zero - Деление на ноль
do statement must have while do - должен иметь while
do-while statement missing (- в do-while операторе отсутствует (
do-while statement missing) - в do-while операторе отсутствует)
do-while statement missing ; - в do-while операторе отсутствует ;
Duplicate case - Двойной случай
Expression expected - Ожидается выражение
Expression of scalar type expected - Ожидается выражение скалярного типа
extern variable cannot be initialized - Extern переменная не может быть инициализирована
File name too long - Имя файла слишком длинное

For statement missing (- В for операторе отсутствует (

For statement missing) - В for операторе отсутствует)

For statement missing ; - В for операторе отсутствует ;

Function 'function' should have a prototype - Функция " функция " должна иметь прототип

Function call missing) - Отсутствие) в вызове функции

Function should return a value - Функция должна возвращать значение

Functions may not be part of a struct or union - Функции не могут быть частью структуры или объединения

Identifier expected - Ожидается идентификатор

If statement missing (- В операторе if отсутствует (

If statement missing) - В операторе if отсутствует)

Illegal initialization - Неправильная инициализация

Illegal octal digit - Неправильная восьмеричная цифра

Illegal pointer subtraction - Неправильное вычитание указателя

Illegal structure operation - Неправильное действие над структурой

Illegal use of floating point - Неправильное использование плавающей точки

Illegal use of member pointer - Неправильное использование указателя

Illegal use of pointer - Неправильное использование указателя

Implicit conversion of 'type1' to 'type2' not allowed - Неявное преобразование " type1 " к " type2 " не позволено

Incompatible type conversion - Несовместимое преобразование типа

Incorrect number format - Неправильный числовой формат

ncorrect use of default - Неправильное использование default

Member pointer required on right side of .* or ->* - Указатель требует с правой стороны * или - > *

Memory reference expected - Ожидается ссылка

Misplaced break - break стоит не на месте

Misplaced continue - continue стоит не на месте

Misplaced elif directive - elif директива не на месте

Misplaced else - else стоит не на месте

Misplaced endif directive - endif директива не на месте

Multiple declaration for 'identifier' - Многократная декларация для " идентификатор "

Numeric constant too large - Числовая константа слишком большая

operator [] missing] - В операторе [] пропущена]

sizeof may not be applied to a bit field - Sizeof не имеет права обращаться к битовому полю

sizeof may not be applied to a function - Sizeof не имеет права обращаться к функции

Size of 'identifier' is unknown or zero- Размер " идентификатора " неизвестен или ноль

Size of the type is unknown or zero - Размер типа неизвестен или ноль

Statement missing ; - Отсутствие ;

Structure required on left side of . or .* - Структура требует на левой стороне . или . *

Structure size too large - Размер Структуры слишком большой

Switch statement missing (- Отсутствие в операторе switch (

Switch statement missing) - Отсутствие в операторе switch)

Too few parameters in call - Слишком мало параметров в вызове

Too few parameters in call to function - Слишком мало параметров в вызове функции

Too many default cases - Слишком много default

Too many error or warning messages - Слишком много ошибок или предупреждений

Type name expected - Ожидается имя типа

Unable to open include file 'filename' - Невозможно открыть файл " filename "

Unexpected } - Неожиданная }

Use . or -> to call function - Использование . или - > для вызова функции

Use . or -> to call 'member', or & to take its address - Использование . или - > для обращения к памяти и получения адреса

Value of type void is not allowed - Значение типа не дозволено

Variable 'identifier' is initialized more than once - Переменная " идентификатор " -инициализирована больше чем однажды

While statement missing (- Отсутствие (в операторе while

While statement missing) - Отсутствие) в операторе while

СПИСОК ЛИТЕРАТУРЫ

1. Дейтел Х. Как программировать на Си- М:ООО « Бином – Пресс», 2006-912с.
2. Джехани Н. Программирование на языке Си- М.: Радио и связь 1988-270с.
3. Белецкий Я. Энциклопедия языка Си. -М.:Мир, 1992. – 687 с.
4. Белецкий Я. Турбо Си: Новая разработка: Учеб. Пособие для студентов высших учебных заведений. -М.: Машиностроение, -1994. – 400 с.
5. Глушаков, С. В. Язык программирования С++: учебное пособие / С. В. Глушаков, А. В. Коваль, С. В. Смирнов. - Харьков: Фолио, 2002. - 500 с.
6. Керниган Б, Ритчи Д. Язык программирования Си. Задачи по языку Си. М.:ФиС, 1985. – 280 с.
7. Павловская Т. А. С/С++ Структурное программирование: практикум / Т. А. Павловская. - СПб.: Питер , 2007. - 239 с.
8. Павловская Т.А. С/С++. Программирование на языке высокого уровня. : учебник для вузов / Павловская Т.А. - СПб.: Питер, 2006. - 461 с. Похомов Б.И. С/С++ и Borland C++ C++ Builder для студента БХВ – Петербург, 2006-448с.
9. Родионова Т.Е. Элементы программирования на Си: Методические указания для студентов специальности 5102 -Ульяновск , УлГТУ, 1998. -52 с.
10. Романовская Л.М. и др. Программирования в среде Си для ПЭВМ ЕС 1991-352с.
11. Страуструп, Б. Языки программирования С++, специальное издание / Б. Страуструп; пер. с англ С. Анисимова. - М. : Издательство БИНОМ- Невский диалект., 2001. - 1099 с. : илл. - (Специальное издание).
12. Тимофеев В.В. Язык С и С++: Программирование в среде С++ Builder 5 – М.: ЗАО «БИНОМ», 2000-368с

Содержание

Эркинбаев М., Мукамбетова С.,
Иманканова К.

Методическое указание
к выполнению лабораторных работ по языку программирования

Технический редактор: Ч.А.Жакыпова
Компьютерная верстка: З.А. Чоколокова

Отпечатано в полиграфическом комплексе
ЫГУ им. К.Тыныстанова.
Заказ 333. Тираж 25.
Тел.: (03922) 52696.