



Белим С.Ю.

**Задания по программированию
на языке высокого уровня
(Учебно-методическое пособие)**

Подписано в печать 10.03.2009
Формат 60 x 84 1/16
Уч.-изд. л. 3,25
Заказ № 506
Тираж 100 экз.

Полиграфический центр КАН
644050, г. Омск, пр. Мира 11 А
Тел.: (381-2) 65-23-73
Лицензия ПЛД № 58-47 от 21.04.97 г

**Кафедра
вычислительных
систем**

Федеральное агентство по образованию
Омский государственный университет им. Ф.М. Достоевского
Факультет компьютерных наук
Кафедра вычислительных систем

Белим С.Ю.

**Задания по программированию на языке высокого уровня
(Учебно-методическое пособие)**

Омск - 2009

УДК 681.3.06

ББК 32.973.26

Б432

Белим С.Ю. Задания по программированию на языке высокого уровня: учебно-методическое пособие. - Омск: Издательство Наследие. Диалог-Сибирь, 2009. - 52 с.

Учебно-методическое пособие составлено для проведения лабораторных работ по курсу «Программирование на языке высокого уровня» для студентов первого курса специальности «Вычислительные машины, комплексы, системы и сети». Задания разбиты на семь тем. По каждой теме приведены примеры выполнения заданий и указаны требования к выполнению заданий.

Введение

Задания, представленные в данном пособии, выполняются на языке программирования C. Материал пособия охватывает семь тем. Первая тема разбита на четыре раздела. Вторая тема разбита на два раздела. Остальные темы не содержат разделов. Тематика заданий соответствует первому семестру лекционного курса:

Тема 1. Базовые конструкции

Раздел 1. Условный оператор и оператор выбора

Раздел 2. Определенный цикл. Циклы с предусловием и постусловием.

Раздел 3. Функции ввода-вывода.

Раздел 4. Функции.

Тема 2. Массивы

Раздел 1. Статические массивы.

Раздел 2. Динамические массивы

Тема 3. Строки

Тема 4. Структуры

Тема 5. Односвязный список

Тема 6. Двусвязный список

Тема 7. Абстрактные типы данных

В каждом разделе (или теме, кроме темы 7) задания разбиты на три уровня сложности: А, В и С. Задания уровня А выполняются студентами во время занятий, после ознакомления с решенными примерами, приведенными в начале раздела. Задания уровня В предназначены для домашней работы, и выполняются студентами самостоятельно в течение недели. Уровень С – это задания повышенной сложности и выполняются студентами в течение более длительного срока, чем уровень В.

Рекомендовано к изданию ученым советом ФКН ОмГУ им. Ф.М.Достоевского.

© Омский госуниверситет, 2009.

3

Тема 1. Базовые конструкции

Раздел 1. Условный оператор и оператор выбора

Пример 1. Написать программу, запрашивающую целое число с клавиатуры и выводящую строку "нечётное число ", если число нечетное и строку "чётное", в противном случае.

Решение:

```
// Подключаем заголовочный файл для работы функций ввода-  
// вывода информации  
#include <stdio.h>  
// С функцией main() начинается выполнение программы  
int main(void){  
    // Определяем переменную, в которую будет записываться число  
    int a;  
    // Приглашаем ввести число с клавиатуры  
    printf("Введите a: ");  
    scanf("%d",&a);  
    // Проверяем число на четность и выводим соответствующие сообщения  
    if (a%2==1){ // можно написать просто if(a%2)- результат не изменится  
        printf("\nнечётное число ");  
    }  
    else{  
        printf("\n чётное число ");  
    }  
    // Завершаем программу  
    return 0;  
}
```

Пример 2. Написать программу, решающую квадратное уравнение. Коэффициенты должны вводиться с клавиатуры.

Решение:

```
// Подключаем заголовочные файлы для работы функций ввода-  
// вывода информации и функции извлечения квадратного корня  
#include <stdio.h>  
#include <math.h>  
// С функцией main() начинается выполнение программы  
int main(void){  
    printf("Найдём корни уравнения ax^2+bx+c=0\n");  
    // Определяем переменные, в которые будут заноситься значения  
    // коэффициентов уравнения  
    float a;  
    float b;  
    float c;  
    // Организуем ввод коэффициентов уравнения  
    printf("Input a: \n");  
    scanf("%f",&a);  
    printf("Input b: \n");  
    scanf("%f",&b);  
    printf("Input c: \n");  
    scanf("%f",&c);  
    // Определяем переменную для хранения значения дискриминанта  
    float D;  
    D=b*b-4*a*c;
```

/Проверяем знак дискриминанта и вычисляем значения корней, если

```
// они действительные  
float x1,x2;  
if (D<0){  
    printf("корни-комплексные");  
}  
else{  
    if (D==0){  
        x1=-b/(2*a);  
        printf("корень кратности 2 x=%3.4f",x1);  
    }  
    else{  
        x1=(-b+sqrt(D))/(2*a);  
        x2=(-b-sqrt(D))/(2*a);  
        printf("x1=%3.4f x2=%3.4f",x1,x2);  
    }  
}
```

/Завершаем программу
return 0;

Пример 3. Написать программу, возвращающую название дня недели по его номеру.

Решение:

```
// Подключаем заголовочные файлы для работы функций ввода-  
// вывода информации
```

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

```
// С функцией main() начинается выполнение программы
```

```
int main(void){
```

```
    // Определяем переменную, в которую будет заноситься значение
```

```
    int n;
```

```
    // Организуем ввод номера дня недели
```

```
    printf("Введите номер дня недели (1-7): \n");
```

```
    scanf("%d",&n);
```

```
    // Определяем день недели
```

```
    switch(n){
```

```
        case 1:{ printf("Понедельник\n"); break; }
```

```
        case 2:{ printf("Вторник\n"); break; }
```

```
        case 3:{ printf("Среда\n"); break; }
```

```
        case 4:{ printf("Четверг\n"); break; }
```

```
        case 5:{ printf("Пятница\n"); break; }
```

```
        case 6:{ printf("Суббота\n"); break; }
```

```
        case 7:{ printf("Воскресение\n"); break; }
```

```
    default:{ printf("Неточний ввод номера\n"); break; }
```

```
}
```

```
/Завершаем программу  
return 0;
```

```
}
```

Уровень А:

1.1.1A. Написать программу, выводящую знак зодиака по введенной дате.

1.1.2A. Написать программу, выводящую количество дней в году по его номеру.

4

5

Примечание: года бывают високосными и невисокосными. В невисокосном году 365 дней, в високосном - 366. Високосными считаются годы, номер которых нацело делится на 4, кроме тех, у которых в номере последние два нуля, а первые две цифры образуют число не делящееся на 4. Остальные годы не високосные.

1.1.3A. Написать программу, выводящую количество дней в месяце невисокосного года по его номеру.

1.1.4A. Написать программу, выводящую время года по введенному номеру месяца.

Уровень B:

Написать программу, вычисляющую значение выражения f , не используя стандартную функцию $abs()$.

$$\begin{array}{ll} 1.1.1B \quad f = |x - |x + 3||, & 1.1.2B \quad f = |x - x - 3| \\ 1.1.5B \quad f = |2x - |x + 3||, & 1.1.6B \quad f = |5x - |x + 3|| \\ 1.1.7B \quad f = |4 - |x - 3||, & 1.1.8B \quad f = |2x + |x - 3|| \end{array}$$

Написать программу, вычисляющую значение выражения

1.1.9B	1.1.10B	1.1.11B
$\begin{cases} 2x^2 + 1, & x < 0 \\ x, & x \geq 10 \\ x - 3, & \text{иначе} \end{cases}$	$\begin{cases} x^2 - 1, & x < 0 \\ x, & x \geq 4 \\ x - 3, & \text{иначе} \end{cases}$	$\begin{cases} 2x^2 - 1, & x < 0 \\ x, & x \geq 7 \\ x - 3, & \text{иначе} \end{cases}$
1.1.12B	1.1.13B	1.1.14B
$\begin{cases} 2x^2, & x < 0 \\ x + 2, & x \geq 5 \\ x - 3, & \text{иначе} \end{cases}$	$\begin{cases} 2 - x^2, & x < 0 \\ x, & x \geq 4 \\ x - 3, & \text{иначе} \end{cases}$	$\begin{cases} 2x^2 + 1, & x < 0 \\ x + 1, & x \geq 1 \\ x, & \text{иначе} \end{cases}$
1.1.15B	$\begin{cases} x^2 + 1, & x < 0 \\ 2x, & x \geq 3 \\ x - 3, & \text{иначе} \end{cases}$	

Раздел 2. Определенный цикл. Циклы с предусловием и постусловием

Пример 1. Написать программу, выводящую на печать квадраты чисел от 0 до 25.

Решение:

```
// Подключаем заголовочный файл для работы функций ввода-вывода информации
#include <stdio.h>
// С функции main() начинается выполнение программы
int main(void){
    int a; // очередной член последовательности
    // Организуем цикл for, определяя параметр цикла (переменная int i)
    // внутри самого оператора for;
    for (int i=0;i<25;i+=1){
        a=i*i;
        printf ("%d\n",a);
    }
    // Завершаем программу
    return 0;
}
```

Пример 2. Написать программу, выводящую на печать первые 6 членов последовательности, определяемую соотношениями: $a_1=2$, $a_{n+1}=a_n^2$.

6

Решение:

```
// Подключаем заголовочный файл для работы функций ввода-вывода информации
#include <stdio.h>
// С функции main() начинается выполнение программы
int a=2; // переменная инициализирована значением первого члена
// внутри самого оператора for;
for (int i=1;i<6;i++){
    // выводим текущий член последовательности
    printf("%d-й член последовательности - равен %d\n",i,a);
    a=a*a; // вычисляем следующий член последовательности
}
```

// Завершаем программу

return 0;

Пример 3. Написать программу, вычисляющую сумму первых n членов последовательности, определяемую соотношениями: $a_1=2$, $a_{n+1}=a_n^2$.

Решение:

```
// Подключаем заголовочный файл для работы функций ввода-вывода информации
#include <stdio.h>
// С функции main() начинается выполнение программы
int main(void){
    int a=2; // первый член последовательности
    int n=0; // переменная для хранения значения суммы
    int sum; // переменная, в которую будет заноситься количество слагаемых
    // Организуем ввод п
    printf("Введите количество членов последовательности \n");
    scanf("%d",&n);
    // Организуем цикл for, определяя параметр цикла (переменная int i)
    // внутри самого оператора for;
    for (int i=1;i<=n;i++){
        a=a*a; // вычисляем следующий член последовательности
        sum+=a; // добавляем новый член к сумме
    }
}
```

// Выводим значение суммы

printf ("Сумма равна %d\n",sum);
// Завершаем программу

return 0;
}

Пример 4. Написать программу, выводящую на печать положительные целые числа, квадраты которых не превышают некоторого натурального N .

Решение:

```
// Подключаем заголовочный файл для работы функций ввода-вывода информации
#include <stdio.h>
// С функции main() начинается выполнение программы
int main(void){
    int a=1; // переменная для перебора текущих чисел
    // Завершаем программу
```

7

```
int n; // переменная, в которую будет заноситься значение границы N
// Организуем ввод п
printf("Введите натуральное число N: \n");
scanf("%d",&n);
// Организуем цикл while
while(a<=n){
    printf("Следующее число, чей квадрат меньше числа N=%d - %d\n", n , a);
}
// Завершаем программу
return 0;
}

Пример 5. Написать программу, выводящую на печать сумму членов убывающей геометрической прогрессии, превышающих некоторое число  $a$ .
```

Решение:

// Подключаем заголовочный файл для работы функций ввода-вывода информации
#include <stdio.h>

```
int main(void){
    float b; // текущий член геометрической прогрессии
    float q; // множитель
    float s; // текущее значение суммы
    //-----
```

printf("Введите первый член прогрессии: \n");
scanf("%f",&b);
printf("Введите знаменатель прогрессии: \n");
scanf("%f",&q);
//-----

int a;
printf("Введите число a: \n");
scanf("%f",&a);
//-----

// Организуем цикл do-while

do{
 b*=q;
 s+=b;
} while(b>a);
printf("Сумма заданных членов равна %f\n");
return 0;
}

Пример 6. Написать программу, запрашивающую целое число до тех пор, пока не будет введено 0 или 1.

Решение:

```
# include <stdio.h>
int main(void){
    int a;
    do{
        printf("Введите целое число: \n");
        scanf("%d",&a);
    } while((a!=0)||(a!=1));
    return 0;
}
```

Уровень А:

1.2.1A. Написать программу, выводящую на экран 40 первых членов арифметической прогрессии с первым членом равным 1 и модулем 5.

1.2.2A. Написать программу, выводящую на экран 20 первых членов последовательности Фибоначчи. Числа Фибоначчи образуются по правилу: $a_1=1$, $a_2=1$, $a_{n+1}=a_1+a_n$.

1.2.3A. Написать программу, вычисляющую сумму первых n членов последовательности, члены которой могут быть получены из формулы $a_i=(i-2)^2/(i+1)$.

1.2.4A. Написать программу, вычисляющую сумму первых n членов последовательности, члены которой могут быть получены из формулы $a_i=(a_{i-1}+1)^2/(i+1)$.

1.2.5A. Написать программу, выводящую на печать члены убывающей геометрической прогрессии, превышающие некоторое число a .

1.2.6A. Написать программу, запрашивающую целое число в интервале от 0 до 10 и выводящую на экран куб этого числа. При вводе целого числа, не попадающего в данный интервал или дробного числа, программа должна просить повторить ввод.

1.2.7A. Написать программу, вычисляющую сумму первых n членов последовательности, члены которой могут быть получены из формулы $a_i=(a_{i-1}+1)^2/(i+1)$.

1.2.8B. Составить программу, печатающую квадраты всех натуральных чисел от 0 до заданного натурального n , но разрешается использовать из арифметических операций лишь сложение и вычитание. /*Можно использовать $(a-1)^2=2a+1$ */

1.2.7B. Составить программу, печатающую разложение на простые множители заданного натурального числа $n > 0$ (другими словами, требуется печатать только простые числа и произведение напечатанных чисел должно быть равно n ; если $n = 1$, печатать ничего не надо).

1.2.8B. Разрешено использовать функцию printf() для вывода на консоль лишь одной из цифр - 0, 1, 2, ..., 9. Составить программу, печатающую десятичную запись заданного натурального числа n , но надо напечатать десятичную запись в обратном порядке. (Для $n = 173$ надо напечатать 371.) (Случай $n = 0$ явился бы некоторым исключением, так как обычно нули в начале числа не печатаются, а для $n = 0$ - печатаются.) /* Сначала надо найти степень 10, не превосходящую данное число*/

1.2.9B. Разрешено использовать функцию printf() для вывода на консоль лишь одной из цифр - 0, 1, 2, ..., 9. Составить программу, печатающую десятичную запись заданного натурального числа n , но надо напечатать десятичную запись обратным порядком. (Для $n = 173$ надо напечатать 371.) (Случай $n = 0$ явился бы некоторым исключением, так как обычно нули в начале числа не печатаются, а для $n = 0$ - печатаются.) /* Сначала надо найти степень 10, не превосходящую данное число*/

1.2.10B. Написать программу, вычисляющую сумму первых n членов последовательности, члены которой могут быть получены из формулы $a_i=(a_{i-1}+a_{i-2})^2$.

1.2.11B. Написать программу, вычисляющую сумму первых n членов последовательности, члены которой могут быть получены из формулы $a_i=(i+a_1+i^2a_2)^2$.

1.2.12B. Написать программу, вычисляющую сумму первых n членов последовательности, члены которой могут быть получены из формулы $a_i=1/(i+a_{i-1}+i)$.

9

1.2.13В. Написать программу, вычисляющую сумму первых n членов последовательности, члены которой могут быть получены из формулы $a_i = 1/(i+a_{i-1}+a_{i-2})$.

1.2.14В. Написать программу, вычисляющую сумму первых n членов последовательности, члены которой могут быть получены из формулы $a_i = (i+a_{i-1}+a_{i-2})^2$.

1.2.15В. Написать программу, вычисляющую сумму первых n членов последовательности, члены которой могут быть получены из формулы $a_i = i/a_{i-1} + a_{i-2}$.

1.2.16В. Написать программу, вычисляющую сумму первых n членов последовательности, члены которой могут быть получены из формулы $a_i = i^2/a_{i-1} + a_{i-2}$.

1.2.17В. Написать программу, вычисляющую сумму первых n членов последовательности, члены которой могут быть получены из формулы $a_i = (i/a_{i-1} + a_{i-2})^2$.

1.2.18В. Написать программу, вычисляющую сумму первых n членов последовательности, члены которой могут быть получены из формулы $a_i = (1+i^2 a_{i-1})^2$.

1.2.19В. Написать программу, вычисляющую сумму первых n членов последовательности, члены которой могут быть получены из формулы $a_i = i/(i+a_{i-1})$.

1.2.20В. Написать программу, вычисляющую сумму первых n членов последовательности, члены которой могут быть получены из формулы $a_i = (2+a_{i-1}+i^2 a_{i-2})^2$.

1.2.21В. Написать программу, вычисляющую сумму первых n членов последовательности, члены которой могут быть получены из формулы $a_i = (2^i + i a_{i-1})^2$.

1.2.22В. Написать программу, вычисляющую сумму первых n членов последовательности, члены которой могут быть получены из формулы $a_i = (i+a_{i-1}+i^2 a_{i-2})^2$.

1.2.23В. Написать программу, вычисляющую сумму первых n членов последовательности, члены которой могут быть получены из формулы $a_i = i^2 + a_{i-1} + a_{i-2}$.

Раздел 3. Функции ввода-вывода

Пример 1. Написать программу, которая из файла c.txt прочитает хранящиеся там значения и выведет их на консоль.

/* в файле c.txt хранится следующее:

i1=0xFa i2=-22 i3=074 f= 1.57 l=-125874 ch=k

str=a8 получится

*/

Решение:

```
#include <stdio.h>
int main()
{
    int i1, i2, i3;
    int otv; // переменная для хранения возвращаемого значения функции fscanf()
    float f;
    long l;
    char ch, str[20];
    FILE *in; // указатель на структуру, хранящую сведения о файле, для ввода
    // открываем файл для чтения c.txt
    in=fopen("/home/student/c.txt","r");
    // обрабатываем возможную ошибку открытия файла
    if (in==NULL)
        {printf("не открылся для чтения\n");return 1;}
    // читаем данные из файла c.txt
    otv=fscanf(in, "i1=%x i2=%d i3=%o f=%f l=%ld ch=%c str=%s", &i1, &i2, &i3, &f,
    &l, &ch, str);
}
```

10

```
// обрабатываем возможную ошибку чтения
if (otv!=7)
    {printf("данные прочитаны с ошибками. \n"); return 2;}
// закрываем файл
otv=fopen("c.txt","w");
// обрабатываем возможную ошибку закрытия файла
if (otv==EOF)
    {printf("не закрыли файл \n");return 3;}
// выполним на конsole текущие значения переменных
printf("i1=%d\n",i1); // попробуйте вариант printf("i1=%x\n",i1);
printf("i2=%d\n",i2);
printf("i3=%o\n",i3); // попробуйте вариант printf("i3=%d\n",i3);
printf("f=%f\n",f); // попробуйте вариант printf("f=%f\n",f);
printf("l=%ld\n",l);
printf("ch=%c\n",ch);
printf("str=%s\n",str);
return 0;
}
```

Пример 2. Написать программу, которая выведет на консоль такую картинку:

**
*

Решение:

```
#include <stdio.h>
int main()
{
    int i,j; // переменная i – для перебора строк, переменная j – для пробега внутри i-ой //строки
    printf("Треугольник из звёздочек.\n");
    for(i=0;i<12;i++)
    {
        for(j=0;j<=i;j++)
        {
            printf("%*s");
        }
        printf("\n");
    }
    printf("\n");
    return 0;
}
```

Пример 3. Написать программу, выводящую на печать количество разрядов в натуральном числе а.

Решение:

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

11

```
int main()
{
    int i,a; // переменная i – количество разрядов в числе а
    printf("Введите а: ");
    scanf("%d",&a);
    b=a;
    for(i=0;i>0;i++)
        b/=10;
    printf("Количество разрядов в числе а - %d", i);
    return 0;
}
```

Уровень А:

1.3.1A. Написать программу, которая выведет на консоль такую картинку

1 2 3 4 5
3 123 4 5
32 11 0 10
56 8 1 19

1.3.2B. Написать программу, которая выведет на консоль в виде прямогольной таблицы размером 4×4 числа, хранящиеся в файле a.txt. Числа должны быть выровнены по первой цифре числа.

Содержимое файла a.txt:

1 2 3 4 5 3 123 4 5 32 11 0 10 56 8 1 19
Ответ должен выглядеть следующим образом:

1 2 3 4 5
3 123 4 5
32 11 0 10
56 8 1 19

1.3.3B. В файле a.txt задано 10 целых чисел в десятичной системе счисления. Выведите числа, стоящие на четных местах, в восемеричной системе счисления, а числа, стоящие на нечетных местах, в шестнадцатеричной системе счисления.

1.3.4B. В файле a.txt задано 5 вещественных чисел. Выведите на экран первое число из файла с одним знаком после запятой, второе число с двумя знаками после запятой, третье – с тремя и т.д.

1.3.5B. В файле a.txt задано 5 вещественных чисел. Выведите на экран числа из файла, стоящие на нечетных местах с одним знаком после запятой, а числа, стоящие на четных местах с двумя знаками после запятой.

1.3.6B. В файле a.txt задано 10 целых чисел в восемеричной системе счисления. Выведите числа, стоящие на четных местах в десятичной системе счисления, а числа, стоящие на нечетных местах как вещественные с одним знаком после запятой.

1.3.7B. В файле a.txt задано 10 символов через пробелы. Выведите на печать ASCII-коды этих символов. (Примечание: чтобы получить ASCII-код символа, нужно символьную переменную выводить с шаблоном целого числа.)

1.3.8B. В файле a.txt задано 10 целых чисел. Выведите их на печать в четыре строки так, чтобы в первой строке было 4 числа, во второй – 3, в третьей – 2, в четвертой – 1.

1.3.9B. В файле a.txt задано 10 целых чисел. Выведите их на печать в четыре строки так, чтобы в первой строке было 1 число, во второй – 2, в третьей – 3, в четвертой – 4.

1.3.10B. В файле a.txt задано 5 целых чисел. Выведите их на печать по одному числу в строке, так чтобы у первого числа не было отступа от левого края, у второго числа отступ один символ, у третьего – 2 символа, у четвертого – 3 символа, у пятого – 4 символа.

1.3.11B. В файле a.txt задано 5 целых чисел. Выведите их на печать по одному числу в строке, так чтобы у первого числа отступ от левого края был 4 символа, у четвертого – 1 символом, у пятого – 0 символов.

1.3.12B. В файле a.txt задано 10 целых чисел. Выведите их на печать по одному числу в строке, так чтобы у четных чисел отступ от левого края был 4 символа, а у нечетных – 1 символ.

1.3.13B. В файле a.txt задано 5 целых чисел. Выведите их на печать в одну строку так, чтобы между первым и вторым символом был один пробел, между вторым и третьим – два пробела, между третьим и четвертым – 3 пробела и т.д.

1.3.14B. В файле a.txt задано некоторое количество целых чисел. Напишите программу, которая запрашивает у пользователя целое число m , и выводит числа строками по m элементов. (Последняя строка может содержать меньше чем m элементов).

1.3.15B. В файле a.txt задано некоторое количество целых чисел. Напишите программу, которая запрашивает у пользователя целое число m , и выводит числа в m строк. Последняя

1.3.3A. Написать программу, которая выведет на консоль такую картинку

*

**

строка может иметь меньше элементов чем предыдущие.

Раздел 4. Функции.

Пример 1. Написать программу, которая из файла `c.txt` прочитает хранящиеся там значения, и выведет их на консоль. Всё оформить в виде нескольких функций.

/* в файле `c.txt` хранится следующее:

`i1=0xFA i2=-22 i3=074 f= 1.57 l=125874 ch=k`

страница получится

*/

Решение:

```
#include <stdio.h>
int i1, i2, i3;
int otv; // переменная для хранения возвращаемого значения функции fscanf()
float f;
long l;
char ch, str[20];
FILE *in; // указатель на структуру, хранящую сведения о файле, для ввода
int obrabotka_fails(); // прототип функции, обрабатывающей файл
void input(); // прототип функции вывода значений на консоль
//----- определение функции работы с файлом
int main()
{
    int a=obrabotka_fails(); // вызов функции
    switch(a)
    {
        case 1: printf("не открылся для чтения \n"); break;
        case 2: printf("данные прочитаны с ошибками.\n"); break;
        case 3: printf("не закрыты файл \n"); break;
        case 0: { input(); break; }
    }
    return 0;
}
//----- определение функции работы с файлом
int obrabotka_fails()
{
    // открываем файл для чтения c.txt
    in=fopen("/home/student/c.txt","r");
    // обрабатываем возможную ошибку открытия файла
    if (in==NULL) return 1;
    // читаем данные из файла c.txt
    otv=fscanf(in, "i1=%d i2=%d i3=%d f=%f l=%ld ch=%c str=%s", &i1, &i2, &i3, &f,
    &l, &ch, str);
    // обрабатываем возможную ошибку чтения
    if (otv!=7) return 2;
    // закрываем файл
    otv	fclose(in);
    // обрабатываем возможную ошибку закрытия файла
    if (otv==EOF) return 3;
    return 0;
}
```

14

----- определение функции вывода значений на консоль

```
void input()
{
    // выводим на консоль текущие значения переменных
    printf("i1=%d\n",i1); // попробуйте вариант printf("i1=%ex\n",i1);
    printf("i2=%d\n",i2); // попробуйте вариант printf("i2=%d\n",i2);
    printf("i3=%d\n",i3); // попробуйте вариант printf("i3=%d\n",i3);
    printf("f=%f\n",f); // попробуйте вариант printf("f=%f\n",f);
    printf("ch=%c\n",ch);
    printf("str=%s\n",str);
    return;
}
```

Пример 2. Написать программу, которая выводит на печать значения функции $F(x)=3x^2+1$ в точках интервала от 0 до 20 с шагом 1. Вычисление $F(x)$ должно быть реализовано в виде отдельной функции.

Решение:

```
#include <stdio.h>
int F(int x); // прототип функции, вычисляющей значения F(x)=3x^2+1
int main()
{
    int i;
    for(i=0; i<=20; i++) printf("%d ", F(i)); // передача параметра по значению
    return 0;
}
int F(int x)
{
    return 3*x*x+1;
}
```

Пример 3. Написать программу, в которой функция `main()` вызывает функцию, передавая ей три числа. Функция должна первое число увеличить на 1, второе – на 2, третье – на 3. Функция `main()` полученные числа выводит на печать.

Решение:

```
#include <stdio.h>
void F(int x, int *y, int *z); // прототип функции, изменяющей числа
int main()
{
    int a,b,c;
    printf("Введите a: ");
    scanf("%d",&a);
    printf("Введите b: ");
    scanf("%d",&b);
    printf("Введите c: ");
    scanf("%d",&c);
    F(&a,&b,&c); // передача параметров по величине
    printf("%d %d %d", a,b,c);
    return 0;
}
void F(int *x, int *y, int *z)
{
    *x++; *y+=2; *z+=3;
}

```

Уровень А:

1.4.1A. Написать программу, которая выводит на печать значения функции $F(x)=3x^3+x^2$ на

15

интервале от 0 до 20 с шагом 2. Вычисление $F(x)$ должно быть реализовано в виде отдельной функции.

1.4.2A. Написать программу, проверяющую делимость целого числа на числа от 2 до 10. Проверка делимости на конкретное число должна быть реализована в виде отдельной функции.

1.4.3A. Написать программу, которая выводит на печать квадрат и куб числа, они должны вычисляться в отдельной функции.

1.4.4A. Написать программу, которая выводит на печать значения $1^1, 2^2, 3^3, 4^4, 5^5$. Степени чисел должны вычисляться в отдельной функции.

Уровень В:

Условия задач 1.4.1B - 1.4.15B совпадают с 1.3.1B - 1.3.15B. Переписать текст программы, реализуя отдельные блоки в виде функций.

Уровень С:

Вычислить и вывести на экран в виде таблицы значения функции F в интервале от $X1$ до $X2$ с шагом $4X$. Значения параметров $a, b, c, X1, X2, dX$ должны запрашиваться с клавиатуры и проверяться на корректность ($X2>=X1, dX<=|X2-X1|$). В случае невозможности вычисления значения F при некотором x , должно выводиться сообщение *нетого!*

1.1C $F = \frac{x^2 + b}{x - c}, x < 0, b \neq 0$	1.7C $F = \begin{cases} -ax^2 - b, x < 5, c = 0 \\ \frac{x-a}{x-c}, x > 0, b = 0 \\ \frac{x}{x-c}, \text{иначе} \end{cases}$
1.2C $F = \frac{\frac{1}{ax} - b, x + 5 < 0, c = 0}{\frac{x-a}{x}}, x > 0, c \neq 0$	1.8C $F = \begin{cases} -ax^2, c < 0, a \neq 0 \\ \frac{x-a}{xc}, x > 0, a = 0 \\ \frac{x}{c}, \text{иначе} \end{cases}$
1.3C $F = \frac{ax^2 + bx + c, a < 0, c \neq 0}{\frac{-a}{x-c}, a > 0, c = 0}$	1.9C $F = \begin{cases} ax^2 + b^2 x, a < 0, x \neq 0 \\ \frac{x-a}{x-c}, a > 0, x = 0 \\ 1 + \frac{x}{c}, \text{иначе} \end{cases}$
1.4C $F = \frac{-ax + b, c < 0, x \neq 0}{\frac{x-a}{c}, c > 0, x = 0}$	1.10C $F = \begin{cases} ax^2 - bx + c, x < 3, b \neq 0 \\ \frac{x-a}{x-c}, x > 3, b = 0 \\ \frac{x}{c}, \text{иначе} \end{cases}$
1.5C $F = \frac{a - \frac{x}{10+b}, x < 0, b \neq 0}{\frac{x-a}{x-c}, x > 0, b = 0}$	1.11C $F = \begin{cases} ax^2 + bc, x < 1, b \neq 0 \\ \frac{x-a}{x-c}, x > 1, b = 0 \\ \frac{x}{c}, \text{иначе} \end{cases}$

1.6C $F = \begin{cases} ax^3 + b^2 x, c < 0, b \neq 0 \\ \frac{x-a}{x-c}, c > 0, b = 0 \\ \frac{x}{c}, \text{иначе} \end{cases}$	1.12C $F = \begin{cases} -ax^2 + b, x < 0, b \neq 0 \\ \frac{x-a}{x-c} + 5, x > 0, b = 0 \\ \frac{-x}{c}, \text{иначе} \end{cases}$
1.13C $F = \begin{cases} ax^4 + b, x < 1 < 0, b - x \neq 0 \\ \frac{x-a}{x}, x - 2 > 0, b + x = 0 \\ \frac{x}{c}, \text{иначе} \end{cases}$	1.18C $F = \begin{cases} ax^3 + bx^2, x < 0, b \neq 0 \\ \frac{x-a}{x-c}, x > 0, b = 0 \\ \frac{-x+5}{c(x-10)}, \text{иначе} \end{cases}$
1.14C $F = \begin{cases} -ax^4 - b, x < c < 0, a \neq 0 \\ \frac{x-a}{x-c}, x > c > 0, a = 0 \\ \frac{x}{c}, \text{иначе} \end{cases}$	1.19C $F = \begin{cases} (ax + c)^2 - b, x < 5, b \neq 0 \\ \frac{x-ac}{ax}, x > b, b = 0 \\ \frac{x}{c}, \text{иначе} \end{cases}$
1.15C $F = \begin{cases} ax^5 + b, x < 0, b + c \neq 0 \\ \frac{x-a}{x-c}, x > 0, b + c = 0 \\ \frac{x}{c}, \text{иначе} \end{cases}$	1.20C $F = \begin{cases} \frac{2x-c}{cx-a}, x < 0, b \neq 0 \\ \frac{x-a}{x-c}, x > 0, b = 0 \\ \frac{-x-c}{c}, \text{иначе} \end{cases}$
1.16C $F = \begin{cases} a(x+c)^2 - b, x = 0, b \neq 0 \\ \frac{x-a}{x-c}, x > 0, b = 0 \\ a + \frac{x}{c}, \text{иначе} \end{cases}$	1.21C $F = \begin{cases} ax^2 + b(x-3), x < 0, bc \neq 0 \\ \frac{x-b}{x-c}, x > 0, bc = 0 \\ \frac{x-a}{c}, \text{иначе} \end{cases}$
1.17C $F = \begin{cases} ax^5 - cx + b, x + 10 < 0, b \neq 0 \\ \frac{x-a}{x-c}, x + 10 > 0, b = 0 \\ \frac{-x}{a-c}, \text{иначе} \end{cases}$	1.22C $F = \begin{cases} (ax - c)x^2 + b, xc < 0, b + a \neq 0 \\ \frac{-a}{x-c}, xc > 1, b + a = 0 \\ \frac{x+2}{c}, \text{иначе} \end{cases}$
1.23C $F = \begin{cases} a(x-4)^2 + b, ac < 0, b \neq 0 \\ \frac{x-a}{x-c}, ac > 0, b = 0 \\ 3 + \frac{x}{c}, \text{иначе} \end{cases}$	1.25C $F = \begin{cases} ax^2 + b \frac{c}{x}, x < 0, b + a \neq 0 \\ \frac{x-a}{x-c}, \frac{2}{x} > 0, b + a = 0 \\ \frac{a-x}{x}, \text{иначе} \end{cases}$

17

$$1.24CF = \begin{cases} ax^2 + bx(x-5), & x+c < 0, b \neq 0 \\ \frac{x-3a}{x-c}, & x+c > 0, b=0 \\ a + \frac{x}{c}, & \text{иначе} \end{cases}$$

Тема 2. Массивы

Раздел 1. Статические массивы.

Пример 1. Написать функцию, которая заполняет массив arr[n] нулями (то есть, все элементы этого массива arr[0]..arr[n-1] равнялись бы нулю, независимо от начального значения переменных arr[i]).

Решение:

```
#include <stdio.h>
void nulo(int mas[], int n); //прототип функции
const int n=10;
int main()
{
    int arr[n]={1,2,3,4,5}; //объявление массива и явная инициализация первых пяти //элементов
    for(int i=0; i<n; i++) printf("%i- элемент=%i\n",i,arr[i]);
    nulo(arr, n); // вызов функции, которая зануляет элементы и выводит их на печать
    return 0;
}
void nulo(int mas[],int n) // определение функции
{
    for(int i=0; i<n; i++)
    {
        mas[i]=0;
        printf("%i- элемент=%i\n",i,mas[i]);
    }
    return;
}
```

Пример 2. Написать функцию, которая определяет количество элементов массива, превышающих число а. Количество элементов массива n=10. Значения элементов массива и число а запрашиваются в отдельной функции с консоли.

Решение:

```
#include <stdio.h>
void input(void); //прототип функции ввода
const int n=10;
int mas[n];
```

```
int main(void)
{
    int k=0;
    input();
    for(int i=0; i<n; i++){
        if(mas[i]>a) k++;
    }
    printf("количество элементов, больших %i, в массиве=%i\n",a,k);
    return 0;
}
```

void input(void) { // определение функции ввода элементов массива

```
int i;
for(i=0;i<n;i++){
    printf("\nВведите целое число, которое будет элементом массива m[%i]: ",i);
    scanf("%i",&m[i]);
}
```

printf("\nВведите число а: ");

scanf("%i",&a);

return; }

Уровень А:

2.1.1A. Подсчитать количество нулей в массиве. Количество элементов массива n=10. Значения элементов массива запрашиваются в отдельной функции с консоли.

2.1.2A. Не используя оператора присваивания для массивов, написать программу, позволяющую копировать один массив в другой, попутно подсчитывая сколько раз повторяется в качестве элемента одинаковое число а. Количество элементов массива n=10. Значения элементов массива запрашиваются в отдельной функции с консоли.

2.1.3A. Найти максимальный и минимальный элементы массива. Количество элементов массива n=10. Значения элементов массива запрашиваются в отдельной функции с консоли.

2.1.4A. Найти количество различных чисел среди элементов массива. Количество элементов массива n=10. Значения элементов массива запрашиваются в отдельной функции с консоли.

Уровень В:

2.1.1B. Прямоугольное поле n на n разбито на n² квадратных клеток. Некоторые клетки покрашены в черный цвет. Известно, что все черные клетки могут быть разбиты на несколько непересекающихся и не имеющих общих вершин черных прямоугольников. Считая, что цвета клеток даны в виде массива типа arr [m] [n] , где каждый элемент равен единице только тогда когда эта клетка черная, если она белая, то элемент массива равен 0; подсчитать число черных прямоугольников, о которых шла речь.

Указание: число прямоугольников равно числу их левых верхних углов. Является ли клетка верхним углом, можно узнать, посмотрев на ее цвет, а также цвет верхнего и левого соседей. (Не забудьте, что их может не быть, если клетка с краю.)

2.1.2B. Дан массив arr [m] целых чисел. Не используя других массивов, переставить

19

элементы массива в обратном порядке.

2.1.3B. Для массива целых чисел втагу [m+n], рассматриваемый как соединение двух его отрезков, начало длины m (элементы от втагу [0] до втагу [m]) и конец длины n (элементы от втагу [m+1] до втагу [m+n-1]). Не используя дополнительных массивов, переставить начало и конец. Указание: Перевернем (расположим в обратном порядке) отдельно начало и конец массива, а затем перевернем весь массив как единое целое.

2.1.4B. Коэффициенты многочлена хранятся в массиве a[n+1] целых чисел (n - натуральное число, степень многочлена). Вычислить значение этого многочлена в точке x (т. е. a[n]*x^n+...+a[1]*x+a[0]). (по схеме Горнера).

2.1.5B. В массивах a [k] и b [l] хранятся коэффициенты двух многочленов степеней k и l. Поместить в массив c[m] коэффициенты их произведения. (Числа k, l, m - натуральные, m = k + l; элемент массива i содержит коэффициент при x в степени i.)

2.1.6B. Данные два вектора массива x[k] и y[l]. Найти количество общих элементов в этих массивах (т. е. количество тех целых i, для которых x[i] = y[i] для некоторых i и j).

2.1.7B. Решить предыдущую задачу, если известно лишь, что x[1] <= ... <= x[k] и y[1] <= ... <= y[l] (возрастание заменено неубыванием).

2.1.8B. Даны два массива x[0] <= ... <= x[k] и y[0] <= ... <= y[l-1]. "Соединить" их в массив z[0] <= ... <= z[m-1] (m = k+l; каждый элемент должен входить в массив z столько раз, сколько раз онходит в общей сложности в массивах x и y).

Этот процесс можно пояснить так. Пусть у нас есть две стопки карточек, отсортированных по алфавиту. Мы соединяем их в одну стопку, выбирая каждый раз ту из верхних карточек обеих стопок, которая идет раньше в алфавитном порядке.

2.1.9B. Даны два массива x[0] <= ... <= x[k] и y[0] <= ... <= y[l]. Найти их "пересечение", т. е. массив z[0] <= ... <= z[m], содержащий их общие элементы, причем кратность каждого элемента в массиве z равнается минимуму из его кратностей в массивах x и y.

2.1.10B. Даны два массива x[0]<=...<=x[k] и y[0]<=...<=y[l] и число q. Найти сумму вида x[i]*y[j], наиболее близкую к числу q. (Число действий порядка k+l, дополнительная память - фиксированное число целых переменных, сами массивы менять не разрешается.)

Указание: Надо найти минимальное расстояние между элементами x[1]<=...<=x[k] и q-y[l]<=...<=q-y[1], что непросто сделать в ходе их слияния в один (воображаемый) массив.

Раздел 2. Динамические массивы

Пример 1. Напишите программу, которая запрашивает число n с консоли, создает одномерный динамический массив размера n и заполняет его единицами. Далее необходимо вывести на печать все элементы массива.

Решение:

```
#include <stdio.h>
int main() {
int n;/определляем переменную для сохранения количества элементов массива
printf("Введите количество элементов массива: \n");//запрашиваем n
scanf("%i",&n); //читываем n с консоли
int* m=new int[n];//заполняем динамический массив
int i;
for(i=0;i<n;i++) m[i]=1;//присваиваем всем элементам массива значение 1
```

```
for(i=0;i<n;i++) printf("%i",m[i]);//выводим элементы массива на печать
return 0;
```

}

Пример 2. Написать программу, которая, узнав от пользователя количество строк и столбцов двумерного динамического массива, выделяет место в памяти и заполняет его идущими подряд натуральными числами, начиная с 1. Далее необходимо, по номеру строки, заданному пользователем, вывести на печать элементы, делящиеся без остатка на 3, расположенные в этой строке. Проход по элементам массива должен осуществляться с помощью указателя.

```
Primer:
#include <stdio.h>
int main()
{
int n,m;/определляем переменные для сохранения количества элементов массива
printf("Введите количество строк массива: "); //запрашиваем n
scanf("%i",&n); //читываем n с консоли
printf("Введите количество столбцов массива: "); //запрашиваем m
scanf("%i",&m); //читываем m с консоли
int i,j,k;l;
//заполняем динамический массив
int** mass=new int*[n];
for(i=0;i<n;i++)
{
    *(mass+i)=new int[m];
    // присваиваем значения элементам массива, *(*(mass+i)+j) то же, что и mass[i][j]
    for(j=0;j<m;j++)
    {
        *(*(mass+i)+j)=k;
        k++;
    }
}
int a;
printf("Введите номер строки массива: "); //запрашиваем a
scanf("%i",&a); //читываем a с консоли
for(i=0;i<n;i++)
{
    if(*(*(mass+(a-1))+i)%3==0) printf("%i ",*(*(mass+(a-1))+i));
}
return 0;
}
```

Пример 3. Сформулировать результат выполнения функции, определить назначение ("смысловую нагрузку") использованных переменных, написать вызов функции.

20

21

```

int F (int n, int A[])
{
    int i, k, m;
    for (i=0, k=-1, m=0; i<n; i++)
    {
        if (A[i]<0) continue;
        if (k!=i && A[k]<A[i]) m++;
        k=i;
    }
    return m;
}

Решение: Функция осуществляет просмотр массива размерности n, подсчитывается количество пар элементов массива, в которых текущий неотрицательный элемент A[i] больше предыдущего неотрицательного элемента A[k] данного массива. Другими словами, подсчитывается количество упорядоченных по возрастанию неотрицательных чисел в данном массиве (переменная m). Переменная k- номер последнего неотрицательного элемента, i - параметр цикла.

```

Уровень А:

2.2.1А. Напишите программу, которая запрашивает число с консоли, создает одномерный динамический массив размера p и заполняет его натуральными числами от 0 до 1 по убыванию. Далее необходимо вывести на печать элементы массива, для которых модуль разность между номером элемента и его значением является четным числом.

2.2.2А. Напишите программу, которая запрашивает число с консоли, создает одномерный динамический массив размера p и заполняет его последовательными четными числами, начиная с 2 в возрастании. Далее необходимо вывести на печать элементы массива, кратные 4.

2.2.3А. Напишите программу, которая, узнав от пользователя количество строк и столбцов двумерного динамического массива, выделяет место в памяти и заполняет его идущими подряд натуральными числами, начиная с 0. Далее необходимо, по номеру столбца, заданному пользователем, вывести на печать элементы этого столбца, делающиеся без остатка на 3. Проход по элементам массива должен осуществляться с помощью указателя.

2.2.4А. Напишите программу, которая, узнав от пользователя количество строк двумерного динамического массива, выделяет место в памяти для квадратного массива и заполняет его идущими подряд натуральными числами, начиная с 7. Далее необходимо вывести на печать четные элементы массива, расположенные на главной диагонали. Проход по элементам массива должен осуществляться с помощью указателя.

Уровень В:

2.2.1В. Сформулировать результат выполнения функции, определить назначение ("смысловую нагрузку") использованных переменных, написать вызов функции.

```

void F (int a, int A[10])
{
    int i,n;
    for (i=0, n=0; n!=0; i++, n=n/10);
    for (A[i]=1,n=0; n!=0; i--, n=n/10)
        A[i]=n%10;
}

```

2.2.2В. Сформулировать результат выполнения функции, определить назначение ("смысловую нагрузку") использованных переменных, написать вызов функции.

```

void F (int v, int A[], int m)
{
    int i,a;
    for (i=0, a=2; a<v && i<m-1; a++)
        for (n=2; n<a; n++)
            if (a%n == 0) break;
}

```

22

```

for (j=i+1, m=0; j<n; j++) if (c[i]==c[j]) m++;
if (m>s) s=m;
}
return s;
}

2.2.9В. Сформулировать результат выполнения функции, определить назначение ("смысловую нагрузку") использованных переменных, написать вызов функции.

int F (int n, int c[])
{
    int i,j, k, m;
    for (i=k=m=0; i<n-1; i++)
    {
        if (c[i]<c[i+1]) k++;
        else {
            if (k>m) m=k;
            k=0;
        }
    }
    if (k>m) m=k;
    return m;
}

```

Уровень С:
Программа должна запрашивать размеры матрицы и самостоятельно заполнять ее с помощью генератора случайных чисел. Верхняя граница для значения элементов матрицы также вводится с клавиатуры.

2.1С Даны целочисленная прямоугольная матрица. Определить:
1) количество строк, не содержащих ни одного нулевого элемента;
2) максимальное из чисел, встречающихся в заданной матрице более одного раза.

2.2С Даны целочисленная прямоугольная матрица. Определить:
1) количество столбцов, не содержащих ни одного нулевого элемента;
2) минимальное из чисел, встречающихся в заданной матрице более одного раза.

2.3С Даны целочисленная прямоугольная матрица. Определить:
1) количество строк, содержащих хотя бы один нулевой элемент;
2) количество чисел, встречающихся в заданной матрице более одного раза.

2.4С Даны целочисленная прямоугольная матрица. Определить:
1) номер строки, содержащей самую длинную серию одинаковых элементов;
2) максимальное из чисел, встречающихся в заданной матрице ровно один раз.

2.5С Даны целочисленная прямоугольная матрица. Определить:
1) произведение элементов тех строк, которые не содержат ни одного нулевого элемента;
2) максимум среди сумм элементов диагоналей, параллельных главной диагонали.

2.6С Даны целочисленная прямоугольная матрица. Определить:
1) сумму элементов тех строк, которые содержат хотя бы один нулевой элемент;
2) количество чисел, совпадающих с номером столбца, в которых они находятся.

2.7С Даны целочисленная квадратная матрица. Определить:
1) скалярное произведение строки, в которой находится наибольший элемент матрицы, на столбец с наименшим элементом;
2) количество чисел, совпадающих с номером строки, в которых они находятся.

2.8С Даны целочисленная квадратная матрица.
1) Упорядочить ее строки по их убыванию суммы их элементов;
2) Сумму элементов, оба индекса которых нечетные.

2.9С Даны целочисленная квадратная матрица.
1) Упорядочить ее строки по их убыванию их наибольших элементов;

2) Сумму элементов, оба индекса которых четные.

if (n== a) A[i]=a; }
A[i]=0;

2.2.3В. Сформулировать результат выполнения функции, определить назначение ("смысловую нагрузку") использованных переменных, написать вызов функции.

```

void F (int v, int A[], int m)
{
    int i,a;
    for (i=0, a=2; i<v && i<m-1; a++)
    {
        if (a%A[i]== 0) break;
        if (j== i) A[i]=a;
    }
    A[i]=0;
}

```

2.2.4В. Сформулировать результат выполнения функции, определить назначение ("смысловую нагрузку") использованных переменных, написать вызов функции.

```

void F (int val, int A[], int n)
{
    int i, m;
    for (i=0; val==1 && i<n-1; i++)
    {
        for (m=2; val % m !=0; m++);
        val /= m;
        A[i]=m;
    }
    A[i]=0;
}

```

2.2.5В. Сформулировать результат выполнения функции, определить назначение ("смысловую нагрузку") использованных переменных, написать вызов функции.

```

int F (int n, int c[])
{
    int i, j, k;
    for (i=0; i<n-1; i++)
    {
        for (j=i+1; j<n; j++)
        {
            if (c[i]== c[j]) return i;
        }
        return -1;
    }
}

```

2.2.6В. Сформулировать результат выполнения функции, определить назначение ("смысловую нагрузку") использованных переменных, написать вызов функции.

```

void F (int n, int c[])
{
    int i, j, k;
    for (i=0, j=n-1; i<j; i++)
    {
        if (c[i]== c[j]) k++;
    }
    return k;
}

```

2.2.7В. Сформулировать результат выполнения функции, определить назначение ("смысловую нагрузку") использованных переменных, написать вызов функции.

```

int F (int n, int c[])
{
    int i, j, k1, k2;
    for (i=0; i<n; i++)
    {
        for (j=i+1; k2=0; j<n; j++)
        {
            if (c[i]== c[j]) { if (c[i]<c[j]) k1++; else k2++; }
        }
        if (k1==k2) return i;
    }
    return -1;
}

```

2.2.8В. Сформулировать результат выполнения функции, определить назначение ("смысловую нагрузку") использованных переменных, написать вызов функции.

```

int F (int n, int c[])
{
    int i, j, m, s;
    for (s=0, i=0; i<n-1; i++)
    {
        for (j=i+1; j<n; j++)
        {
            if (c[i]== c[j]) m++;
        }
        if (m>s) s=m;
    }
    return s;
}

```

23

2.10С Даны целочисленная квадратная матрица.

1) Заменить нечетные строки матрицы на заданный вектор.

2) Сумму элементов, один из индексов которых нечетный, а второй четный.

2.11С Даны целочисленная квадратная матрица.

1) Заменить нечетные столбцы матрицы на заданный вектор.

2) Сумму элементов, один из индексов которых нечетный.

2.12С Даны целочисленная квадратная матрица.

1) Поменять в ней 1-ю строку со 2-ой, 3-ю с 4-ой и т.д.

2) Сумму элементов, один из индексов которых четный.

2.13С Даны целочисленная квадратная матрица.

1) Поменять в ней 1-ю строку с n-ой, 2-ю с (n-1)-ой и т.д.

2) Сумму элементов, которые при сложении с обоями своими индексами дают четное число.

2.14С Даны целочисленная квадратная матрица.

1) Поменять в ней 1-й столбец со 2-ым, 2-ой с (n-1)-ым и т.д.

2) Сумму элементов, которые при сложении с обоями своими индексами дают нечетное число.

2.15С Даны целочисленная квадратная матрица.

1) Поменять в ней 1-й столбец с n-ым, 2-ой с (n-1)-ым и т.д.

2) Сумму элементов, которые при сложении с одним из своих индексов дают четное число.

2.16С Даны целочисленная квадратная матрица.

1) Найти количество строк, среднее арифметическое элементов которых меньше заданной величины.

2) Сумму элементов, которые при сложении с одним из своих индексов дают четное число.

2.17С Даны целочисленная квадратная матрица.

1) Найти количество строк, среднее арифметическое элементов которых меньше заданной величины.

2) Удалить из данной матрицы нулевые элементы, заменив их средним арифметическим элементом данной строки.

2.18С Даны целочисленная квадратная матрица.

1) Найти сумму модулей элементов, расположенных выше главной диагонали.

2) Удалить из данной матрицы нулевые элементы, заменив их средним арифметическим соседних четырех элементов.

2.19С Даны целочисленная квадратная матрица.

1) Найти сумму элементов, расположенных выше главной диагонали.

2) Определить номер первого столбца, содержащего нулевой элемент.

2.21С Даны целочисленная квадратная матрица.

1) Найти сумму элементов, расположенных ниже главной диагонали.

2) Определить номер последнего столбца, содержащего нулевой элемент.

2.22С Даны целочисленная квадратная матрица.

1) Найти сумму модулей элементов в строках, содержащих хотя бы один отрицательный элемент.

2) Определить номер первой строки, содержащей нулевой элемент.

2.23С Даны целочисленная квадратная матрица.

1) Найти сумму элементов в строках, содержащих хотя бы один отрицательный элемент.

2) Определить номер последней строки, содержащей нулевой элемент.

24

25

- 2.24С** Дано целочисленная квадратная матрица.
 1) Найти сумму модулей элементов в строках, содержащих хотя бы один неотрицательный элемент.
 2) Определить номер последней строки, не содержащей ни одного нулевого элемента.
- 2.25С** Дано целочисленная квадратная матрица.
 1) Найти сумму элементов в строках, содержащих хотя бы один неотрицательный элемент.
 2) Определить номер последнего столбца, не содержащего ни одного нулевого элемента.

Тема 3. Строки

Пример 1. Сформулировать результат выполнения функции, написать вызов функции.

```
void F(char c[])
{
int i=0,comm=0; // ненулевое значение переменной comm –
//признак нахождения внутри закомментированного блока
for( ; c[i]!='\0'; i++)
{
    if (c[i]=='*' && c[i+1]== '/') {comm--; continue;}
    if (c[i]== '/' && c[i+1]=='*') {comm++; continue;}
    if (comm == 0) c[i]=c[i];
}
c[i]='\0';
}

Решение: Функция осуществляет изменение обрабатываемой строки, удаляя из неё все символы, находящиеся между ограничителями /* и */.
```

```
#include <stdio.h>
void F(char c[])
{
int i=0,comm=0; // ненулевое значение переменной comm –
//признак нахождения внутри закомментированного блока
for( ; c[i]!='\0'; i++)
{
    if (c[i]=='*' && c[i+1]== '/') {comm--; continue;}
    if (c[i]== '/' && c[i+1]=='*') {comm++; continue;}
    if (comm == 0) c[i]=c[i];
}
c[i]='\0';
}

int main()
{
char c[10]={"djkjkj"};
char c[10]={"/*jkjk*/"};
char c[10]={"djk/*jk*"};
printf("Первая строка после обработки символов \n");
F(c);
for(int i=0; c[i]!='\0'; i++)
{
printf("%c",c[i]);
}
printf("\n Вторая строка после обработки символов \n");
F(c);
for(int i=0; c[i]!='\0'; i++)
{
printf("%c",c[i]);
}
printf("\n Третья строка после обработки символов \n");
F(c);
for(int i=0; c[i]!='\0'; i++)
{
printf("%c",c[i]);
}
return 0;
}
```

26

Пример 2. Даны строки, в которых между словами может быть более одного пробела. Написать функцию, копирующую в другую строку все символы первой строки, кроме лишних пробелов.

Решение: Первую строку будем просматривать посимвольно, равномерно двигаясь по ней с помощью индекса i. Когда текущий символ первой строки – пробел, во вторую строку ничего не вносится. Во вторую строку копируется символ, когда он не пробел, при этом надо убедиться, что это не самое первое слово в строке и не первая буква слова (т.е. перед текущим символом был пробел – тогда во вторую строку сначала добавляется пробел, потом текущий символ). По завершению цикла вторую строку оканчиваем символом конца строки. Заметим, что индекс j, с помощью которого движемся по второй строке, меняется не равномерно от итерации к итерации, а только в моменты добавления очередного символа, поэтому длина второй строки будет меньше длины первой.

```
void F(char c1[],char c2[])
{
int i=0,j=0;
for( ; c1[i]!='\0'; i++)
{
    if (c1[i]!=' '){c2[j]=c1[i];
    if (i!=0 && c1[i-1]==' ') {c2[j++]=' ';}
    c2[j++]=c1[i];
    }
    c2[j]='\0';
}

}


```

Пример 3. Даны строки. Написать функцию, которая ищет первое вхождение заданного фрагмента строки.

Решение: Функция должна получать указатель на начало строки, продвигать его к началу обнаруженного фрагмента и возвращать его значение в качестве результата. Для проверки на наличие фрагмента нужно при совпадении элемента строки с первым символом фрагмента далее попарно сравнивать (до обнаружения первого расхождения) все элементы строки с элементами фрагмента (информация о котором в функцию передается с помощью указателя, настроенного на начало). Если фрагмент кончается, то это означает, что все его символы стоят в нашей строке и нужно вернуть указатель в точку вызова функции.

```
char* find (char *p, char *q)
{
for ( ; *p=='\0'; p++)
    if (*q=='\0') return p;
    if (*q!=*p) return p;
}


```

Уровень А:

- 3.1.A. Напишите программу, которая осуществляет изменение обрабатываемой строки, удаляя из неё все цифры.
- 3.2.A. Напишите программу, которая подсчитывает количество символов между ограничителями /* и */.
- 3.3.A. Напишите программу, которая, просматривая строку посимвольно, подсчитывает количество слов в строке.

3.4.A. Перепишите пример 1 так, чтобы заголовок функции void F(char c[]) имел бы вид void F(char *c), а в функции main() строка создавалась бы динамически и заполнялась бы с консоли.

3.5.A. Напишите программу, которая, просматривая строку посимвольно, находит слово максимальной длины (в обрабатывающей функции перемещается указатель по строке, в точку вызова возвращается указатель, настроенный на начало слова).

Уровень В:

- 3.1.B. Сформулировать результат выполнения функции, определить назначение

```
if (*p1==' ') { n=0; *q+=*p; }
else { n++, if (n==1)*q+=*p; }
}
*q=0;
3.10B. Сформулировать результат выполнения функции, определить назначение ("смысловую нагрузку") использованных переменных, написать вызов функции.
void F(char *p)
{
char *q;
int n;
for (n=0, q=p; *p1=='0'; p1++)
{
if (p1[0] == '*' && p1[1] == '/') { n++; p1++; continue; }
if (p1[0] == '/' && p1[1] == '*') { n++; p1++; continue; }
if (n == 0) *q+=*p;
}
*q=0;
}


```

Уровень С:
Во всех заданиях текст находится в файле, имя которого вводится с клавиатуры. Вывод результата также осуществляется одновременно в файл, имя которого вводится с клавиатуры, и на экран монитора.

- 3.1.C. Дан текст из строчных русских букв, за которым следует точка. Напечатать этот текст заглавными буквами.

3.2.C. Дан неупорядоченный текст из заглавных русских букв, за которым следует точка. Определить, упорядочены ли эти буквы по алфавиту. В случае неупорядоченности указать позицию первого символа, нарушающего алфавитный порядок.

3.3.C. Напечатать в алфавитном порядке все различные строчные русские буквы, входящие в заданный текст.

3.4.C. Данна строка латинских символов. Напечатать эту строку, предварительно заменив все вхождения "abc" на "def".

3.5.C. Данна строка латинских символов. Напечатать эту строку, предварительно удалив первое вхождение "w", если такое есть (образовавшуюся «дыру» заполнить последующими буквами, а в конце добавить пробел).

3.6.C. Данна строка латинских символов. Напечатать эту строку, предварительно заменив на "k" первое вхождение "x", если оно есть.

3.7.C. Данна последовательность, содержащая от 1 до 30 слов, в каждом из которых от 1 до 5 строчных латинских букв; между соседними словами – запятая, за последним словом точка. Напечатать эту же последовательность слов, удалив из нее повторно вхождение слова.

3.8.C. Данна последовательность, содержащая от 1 до 30 слов, в каждом из которых от 1 до 5 строчных латинских букв; между соседними словами – запятая, за последним словом точка. Напечатать эту же последовательность слов, удалив из нее повторно вхождение слова.

3.9.C. Данна последовательность, содержащая от 1 до 30 слов, в каждом из которых от 1 до 5 строчных латинских букв; между соседними словами – запятая, за последним словом точка. Напечатать все слова, которые встречаются в последовательности по одному разу.

3.10.C. Данна последовательность, содержащая от 1 до 30 слов, в каждом из которых от 1 до 5 строчных латинских букв; между соседними словами – запятая, за последним словом точка. Напечатать все различные слова, указав для каждого из них число его вхождений в последовательность.

3.11.C. Данна последовательность, содержащая от 1 до 30 слов, в каждом из которых от 1 до 5 строчных латинских букв; между соседними словами – запятая, за последним словом точка. Напечатать все слова в алфавитном порядке разделенные пробелами без запятых.

3.12.C. Данна последовательность, содержащая от 1 до 30 слов, в каждом из которых от 1 до

"смысловую нагрузку") использованных переменных, написать вызов функции.

```
void F (char *p) {
char *q;
for (q=p; *q!= '\0'; q++);
for (q=q-1; p1=q; p1++)
{
    if (p1[0] == '*' && p1[1] == '/') { n++; p1++; continue; }
    if (p1[0] == '/' && p1[1] == '*') { n++; p1++; continue; }
    if (n == 0) *q+=*p;
}
*q=0;
}

3.6.B. Сформулировать результат выполнения функции, определить назначение ("смысловую нагрузку") использованных переменных, написать вызов функции.
int F (char *p) {
int n;
for (n=0; *p!= '\0'; p++, n++);
return n;
}

3.6.B. Сформулировать результат выполнения функции, определить назначение ("смысловую нагрузку") использованных переменных, написать вызов функции.
int M (char *p) {
int n;
if (*p== '\0') return 0;
if (*p!= '\0') n=1;
else n=0;
for (p++; *p!= '\0'; p++)
if (p[0]== '\0' && p[1]== '\0') n++;
return n;
}

3.7.B. Сформулировать результат выполнения функции, определить назначение ("смысловую нагрузку") использованных переменных, написать вызов функции.
char* F (char *p, char *q) {
for (int i=0; p[i]!='\0'; i++)
{
    if (q[i]== '\0') return p;
    if (q[i]== p[i]) return p;
}
return q;
}

3.8.B. Сформулировать результат выполнения функции, определить назначение ("смысловую нагрузку") использованных переменных, написать вызов функции.
void F (char *p, char *q) {
for (*p='0'; p1++;)
for (*q='0'; p1+=*q; q++)
*p=q;
}

3.9.B. Сформулировать результат выполнения функции, определить назначение ("смысловую нагрузку") использованных переменных, написать вызов функции.
void F (char *p) {
char *q;
int n;
for (n=0, q=p; *p!= '\0'; p++)

```

28

29

5 строчных латинских букв; между соседними словами – не менее одного пробела, за последним словом точка. Напечатать слова, которые отличны от последнего слова и являются симметричными.

3.13С. Данна последовательность, содержащая от 1 до 30 слов, в каждом из которых от 1 до 5 строчных латинских букв; между соседними словами – не менее одного пробела, за последним словом точка. Напечатать слова, которые отличны от последнего слова и первая буква слова входит в него еще ровно один раз.

3.14С. Данна последовательность, содержащая от 1 до 30 слов, в каждом из которых от 1 до 5 строчных латинских букв; между соседними словами – не менее одного пробела, за последним словом точка. Напечатать слова, которые отличны от последнего слова и буквы слова упорядочены по алфавиту.

3.15С. Данна последовательность, содержащая от 1 до 30 слов, в каждом из которых от 1 до 5 строчных латинских букв; между соседними словами – не менее одного пробела, за последним словом точка. Напечатать слова, которые отличны от последнего слова и в слове нет повторяющихся букв.

3.16С. Данна последовательность, содержащая от 1 до 30 слов, в каждом из которых от 1 до 5 строчных латинских букв; между соседними словами – не менее одного пробела, за последним словом точка. Напечатать слова, которые отличны от последнего слова предварительно перенеся первую букву в конец.

3.17С. Данна последовательность, содержащая от 1 до 30 слов, в каждом из которых от 1 до 5 строчных латинских букв; между соседними словами – не менее одного пробела, за последним словом точка. Напечатать слова, которые отличны от последнего слова предварительно перенеся последнюю букву в начало.

3.18С. Данна последовательность, содержащая от 1 до 30 слов, в каждом из которых от 1 до 5 строчных латинских букв; между соседними словами – не менее одного пробела, за последним словом точка. Напечатать слова, которые отличны от последнего слова предварительно удалив из слова первую букву.

3.19С. Данна последовательность, содержащая от 1 до 30 слов, в каждом из которых от 1 до 5 строчных латинских букв; между соседними словами – не менее одного пробела, за последним словом точка. Напечатать слова, которые отличны от последнего слова предварительно перенеся последние две буквы.

3.20С. Данна последовательность, содержащая от 1 до 30 слов, в каждом из которых от 1 до 5 строчных латинских букв; между соседними словами – не менее одного пробела, за последним словом точка. Напечатать слова, которые отличны от последнего слова предварительно оставив в слове все последующие вхождения первой буквы.

3.21С. Данна последовательность, содержащая от 1 до 30 слов, в каждом из которых от 1 до 5 строчных латинских букв; между соседними словами – не менее одного пробела, за последним словом точка. Напечатать слова, которые отличны от последнего слова предварительно удалив из слова среднюю букву, если она нечетной длины.

3.24С. Дан текст из заглавных латинских букв, за которым следует пробел. Определить, является ли этот текст правильной записью римскими цифрами целого числа от 1 до 999, и, если является, распечатать это число арабскими цифрами.

30

3.25С. Задано шестнадцатеричное число. Напечатать таблицу умножения в шестнадцатеричной системе счисления от 1 до данного числа.

Тема 4. Структуры

Пример 1. Напишите программу, определяющую массив из структур с двумя целочисленными полями, содержащий три элемента. Напишите функцию, определяющую количество элементов массива с консоли. Напишите функцию, определяющую количество элементов массива, сумма полей которых неотрицательна.

Решение.

```
#include<stdio.h>
struct one{
    int x;
    int y;
}M[3]; // массив из структур с двумя целочисленными полями
void input(void);
void summn(void);
int main(void){
    input();
    summn();
    return 0;
}
----- функция, осуществляющая ввод значений в элементы массива с консоли
void input(void){
    int i;
    for(i=0;i<3;i++){
        printf("Введите поле x, элемента M[%i]: ",i);
        scanf("%d",&M[i].x);
        printf("Введите поле y, элемента M[%i]: ",i);
        scanf("%d",&M[i].y);
    }
}
----- функция, определяющая количество элементов массива, сумма полей которых неотрицательна
void summn(void){
    int i,s=0;
    for(i=0;i<3;i++){
        if(M[i].x+M[i].y>=0) s++;
    }
    printf("%d",s);
    return;
}
----- функция, определяющая количество элементов массива, сумма полей которых
//неотрицательна
void summn(void){
    int i,s=0;
    for(i=0;i<3;i++){
        if(M[i].x+M[i].y>=0) s++;
    }
    printf("%d",s);
    return;
}
Пример 2. Напишите программу, определяющую массив из структур с тремя целочисленными полями, содержащий три элемента. Напишите функцию, осуществляющую ввод значений в элементы массива с консоли. Напишите функцию, осуществляющую проход по массиву и запись в третью поле суммы значений первых двух полей.

```

Решение.

31

```
#include<stdio.h>
struct one{
    int x;
    int y;
    int z;
}M[3];
void input(void);
void summn(void);
int main(void){
    input();
    summn();
    return 0;
}
void input(void){
    int i;
    for(i=0;i<3;i++){
        printf("Введите поле x, элемента M[%i]: \n",i);
        scanf("%d",&M[i].x);
        printf("Введите поле y, элемента M[%i]: \n",i);
        scanf("%d",&M[i].y);
    }
}
----- функция сортировки
void order (one A[], int n){
for(int i=0;i<n;i++) // по принципу "каждый с каждым" просматриваем для i-ого
    //элемента все стоящие за ним
    for(int j=i+1;j<n;j++){
        if(A[i].x>A[j].x) {
            one temp = A[i]; A[i]=A[j]; A[j] = temp;
        }
    }
}
Уровень А:
4.1.А. Напишите программу, определяющую массив из структур с четырьмя полями, и имеющими четыре элемента. Далее необходимо организовать консольный ввод восьми чисел, которые заполняются в каждом элементе массива первыми два поля. Напишите функцию perimeter, которая перебирает элементы указанного массива и в третье поле каждого элемента вносит периметр прямоугольного треугольника, имеющего катеты, длины которых равны значениям первых двух полей. Напишите функцию volumet, которая перебирает элементы указанного массива и в четвертое поле каждого элемента вносит объем прямоугольного параллелепипеда, имеющего ребра, длины которых равны значениям первых трех полей элемента. Напишите функцию, которая упорядочивает массив вторым полем в порядке возрастания.
4.2.А. Напишите программу, определяющую массив из структур с тремя целочисленными полями, содержащий пять элементов. Напишите функцию, осуществляющую ввод значений в первые два поля элементов массива с консоли. Напишите функцию, упорядочивающую элементы массива по сумме первого и второго поля, которое записывается в третью поле, в порядке возрастания.
4.3.А. Напишите программу, определяющую массив из структур с двумя целочисленными полями, содержащий пять элементов. Напишите функцию, осуществляющую ввод значений элементов массива с консоли. Напишите функцию, вычисляющую сумму первых полей тех элементов, у которых произведение полей неотрицательно.
4.4.А. Напишите программу, определяющую массив из структур с двумя целочисленными полями, содержащий пять элементов. Напишите функцию, осуществляющую ввод значений элементов массива с консоли. Напишите функцию, вычисляющую произведение первых полей тех элементов, сумма полей которых четна.
```

Уровень В:
4.1В. Определить значения переменных после выполнения действий над статическими

32

33

данными. Написать программу, демонстрирующую работу функции.

```

struct student{
    char name[20];
    int dd, mm, yy;
    char *prof;
    student *next;
} vasya={"vasya", 1, 1, 1990, "кассир", NULL};
struct student rita = {"rita", 2, 2, 1990, "модель", & vasya},
    *p = & rita;
void ffff() {
    char c1, c2, c3, c4;
    c1 = vasya.name[2];
    c2 = rita.prof[3];
    c3 = p->name[3];
    c4 = p->next->prof[1];
}
4.2В. Определить значения переменных после выполнения действий над статическими данными. Написать программу, демонстрирующую работу функции.
```

```

struct student{
    char name[20];
    int dd, mm, yy;
    char *prof;
    student *next;
} M[3] = {"vasya", 1, 1, 1990, "кассир", NULL}, {"rita", 2, 2, 1990, "модель", & M[0]}, {"irma", 3, 3, 1990, "врач", & M[1]};
void fnc() {
    char c1, c2, c3, c4;
    c1 = M[0].name[2];
    c2 = M[1].prof[3];
    c3 = M[2].next->name[3];
    c4 = M[2].next->next->prof[1];
}
4.3В. Определить значения переменных после выполнения действий над статическими данными. Написать программу, демонстрирующую работу функции.
```

```

struct data{int dd,mm,yy;};
struct student{
    char name[20];
    data dd[3];
} M[2] = {"Лаптев", {{1, 10, 1990}, {8, 8, 1987}, {3, 2, 1988}}}, {"Романов", {{8, 12, 1999}, {3, 2, 1986}, {6, 6, 2005}}};
void f() {int i1,i2;
    i1=M[0].dd[0].mm;
    i2=M[1].dd[2].dd;
}
4.4В. Определить значения переменных после выполнения действий над статическими данными. Написать программу, демонстрирующую работу функции.
```

```

struct tree{int vv;
    tree *l,*r;
} A[4]={{1,NULL,NULL},{2,NULL,NULL},{3,&A[0],&A[1]},{4,&A[2],NULL}};
void f() {int i1, i2, i3, i4;
    i1=A[0].vv; i2=A[3].l->vv; i3=A[3].l->r->vv; i4=A[2].r->vv;
}
4.5В. Определить значения переменных после выполнения действий над статическими данными. Написать программу, демонстрирующую работу функции.
```

```

struct tree{int vv;
}

```

34

```

int *; } A[4]={{1,NULL,NULL},{2,NULL,NULL},{3,&A[0],&A[1]},{4,&A[2],NULL}};
*p=&A[3];
void f(){ int i1, i2, i3, i4;
    i1=A[0].vv; i2=A[3].l->vv; i3=p->l->r->vv; i4=p->vv;
}
4.6. Определить значения переменных после выполнения действий над статическими данными. Написать программу, демонстрирующую работу функции.
```

```

struct data{int dd,mm,yy;};
aa={17,7,1977}, bb={22,7,1982};
struct student{
    char name[20];
    data pd;
    data dd;
    char *zodiak;
} A={"Нескраба",&aa,{1,10,1969}, "Весы"}, B={"Лаптев",&bb,{8,9,1958}, "Скорпион"}, *p=&B;
void f(){int i1, i2, i3, i4;
    i1=A.dd.mm; i2=A.pd.yy; i3=p.dd.dd; i4=p->pd->yy;
}
4.7В. Определить значения переменных после выполнения действий над статическими данными. Написать программу, демонстрирующую работу функции.
```

```

struct data{int vv;
    data *pred;
    data *next;
} C,B,A;
data A={1,&C,&B}, B={2,&A,&C}, C={3,&B,&A}, *p=&A;
void f(){int i1, i2, i3, i4;
    i1=A.next->vv; i2=p->pred->pred->vv;
}
4.8В. Определить значения переменных после выполнения действий над статическими данными. Написать программу, демонстрирующую работу функции.
```

```

struct data{int dd,mm,yy;};
aa={17,7,1977}, bb={22,7,1982};
struct student{
    char name[20];
    data dd[3];
    data dd[3]; A[2]={("Нескраба", {1,10,1969}, {8,8,1988}, {3,2,1978})}, ("Лаптев", {8,12,1958}, {12,3,1976}, {3,12,1967})};
    void f(){int i1, i2;
        i1=A[0].dd[0].mm; i2=A[1].dd[2].dd;
}
4.9В. Определить значения переменных после выполнения действий над статическими данными. Написать программу, демонстрирующую работу функции.
```

```

struct student{
    char name[20];
}

```

35

char *zodiak;
student *next;
A={"Нескраба","Весы",NULL},
B={"Лаптев","Скорпион",&A},
***p[4]=(&B,&A,&A,&B);**
void f(){ char c1, c2, c3, c4;
c1=p[0]->name[2]; c2=p[2]->zodiak[3]; c3=p[3]->next->name[3];
c4=p[0]->next->zodiak[1];

4.10В. Определить значения переменных после выполнения действий над статическими данными. Написать программу, демонстрирующую работу функции.

```

struct data{int dd,mm,yy;};
struct student{
    char name[20];
    data dd[2]; S[2]={("Пушкин",{{6,6,1799},{10,2,1837}}), ("Лермонтов",{{15,10,1814},{27,7,1841}})};
    void f(){int i1, i2;
        i1=S[1].dd[0].mm; i2=S[0].dd[2].dd;
}
Уровень С: Структуры  

Программа должна запрашивать все необходимые параметры, отслеживая правильность формата ввода. Упорядочивание должно происходить после каждого дополнения вносимого в данные, а также позволять вводить не все записи сразу, а добавлять их по мере необходимости.
```

4.1С Описать структуру с именем STUDENT, содержащую следующие поля:

- фамилия и инициалы
- номер группы
- успеваемость (массив из пяти элементов)

Написать программу, выполняющую следующие действия:

- ввод с клавиатуры данных в массив, состоящий из десяти структур типа STUDENT; записи должны быть упорядочены по возрастанию номера группы;
- вывод на дисплей фамилий и номеров групп для всех студентов, включенных в массив, если средний балл студента больше 4.1; если таких студентов нет, вывести соответствующее сообщение.
- вывод полного списка студентов

4.2С Описать структуру с именем STUDENT, содержащую следующие поля:

- фамилия и инициалы
- номер группы
- успеваемость (массив из пяти элементов)

Написать программу, выполняющую следующие действия:

- ввод с клавиатуры данных в массив, состоящий из десяти структур типа STUDENT; записи должны быть упорядочены по возрастанию номера группы;
- вывод на дисплей фамилий и номеров групп для всех студентов, включенных в массив, имеющих только оценки 4 и 5; если таких студентов нет, вывести соответствующее сообщение.
- вывод полного списка студентов

4.3С Описать структуру с именем STUDENT, содержащую следующие поля:

36

- фамилия и инициалы
- номер группы
- успеваемость (массив из пяти элементов)

Написать программу, выполняющую следующие действия:

- ввод с клавиатуры данных в массив, состоящий из десяти структур типа STUDENT; записи должны быть упорядочены по алфавиту;
- вывод на дисплей фамилий и номеров групп для всех студентов, включенных в массив, имеющих хотя бы одну оценку 2; если таких студентов нет, вывести соответствующее сообщение.
- вывод полного списка студентов

4.4С Описать структуру с именем STUDENT, содержащую следующие поля:

- фамилия и инициалы
- номер группы
- успеваемость (массив из пяти элементов)

Написать программу, выполняющую следующие действия:

- ввод с клавиатуры данных в массив, состоящий из десяти структур типа STUDENT; записи должны быть упорядочены по убыванию среднего бала;
- вывод на дисплей фамилий и номеров групп для всех студентов, включенных в массив, не имеющих оценок 2; если таких студентов нет, вывести соответствующее сообщение.
- вывод полного списка студентов

4.5С Описать структуру с именем AEROFLOT, содержащую следующие поля:

- название пункта назначения рейса
- номер рейса
- тип самолета
- количество занятых мест
- общее количество мест

Написать программу, выполняющую следующие действия:

- ввод с клавиатуры данных в массив, состоящий из десяти структур типа AEROFLOT; записи должны быть упорядочены по алфавиту пункта назначения;
- вывод на дисплей номеров рейсов и типов самолета по называнию пункта назначения введенного с клавиатуры; если таких рейсов нет, вывести соответствующее сообщение.
- вывод полного списка рейсов

4.6С Описать структуру с именем AEROFLOT, содержащую следующие поля:

- название пункта назначения рейса
- номер рейса
- тип самолета
- количество занятых мест
- общее количество мест

Написать программу, выполняющую следующие действия:

- ввод с клавиатуры данных в массив, состоящий из десяти структур типа AEROFLOT; записи должны быть упорядочены по возрастанию номера рейса;
- вывод на дисплей номеров рейсов и количества занятых мест по называнию пункта назначения введенного с клавиатуры; если таких рейсов нет, вывести соответствующее сообщение.
- вывод полного списка рейсов

4.7С Описать структуру с именем AEROFLOT, содержащую следующие поля:

37

- вывод полного списка
- 4.23С** Описать структуру с именем ORDER, содержащую следующие поля:
- расчетный счет плательщика
 - расчетный счет получателя
 - перечисляемая сумма
 - вид валюты
- Написать программу, выполняющую следующие действия:
- ввод с клавиатуры данных в массив, состоящий из десяти структур типа ORDER; записи должны быть упорядочены по расчетному счету плательщика в порядке возрастания;
 - вывод на дисплей информации о сумме снятой с расчетного счета плательщика, введенной с клавиатуры; если таких платежей нет, вывести соответствующее сообщение.
 - вывод полного списка
- 4.24С** Описать структуру с именем ORDER, содержащую следующие поля:
- расчетный счет плательщика
 - расчетный счет получателя
 - перечисляемая сумма
 - вид валюты
- Написать программу, выполняющую следующие действия:
- ввод с клавиатуры данных в массив, состоящий из десяти структур типа ORDER; записи должны быть упорядочены по расчетному счету получателя в порядке убывания;
 - вывод на дисплей информации о сумме перечисленной на расчетный счет получателя, введенной с клавиатуры; если таких платежей нет, вывести соответствующее сообщение.
 - вывод полного списка
- 4.25С** Описать структуру с именем ORDER, содержащую следующие поля:
- расчетный счет плательщика
 - расчетный счет получателя
 - перечисляемая сумма
 - вид валюты
- Написать программу, выполняющую следующие действия:
- ввод с клавиатуры данных в массив, состоящий из десяти структур типа ORDER; записи должны быть упорядочены по величине платежей в порядке возрастания;
 - вывод на дисплей информации о платежах и получателе по перечисляемой сумме, введенной с клавиатуры; если таких платежей нет, вывести соответствующее сообщение.
 - вывод полного списка

Тема 5. Односвязный список

Пример 1. Написать функцию, создающую односвязный список для хранения целочисленных данных, состоящий из одного элемента со значением, вводимым с клавиатуры.

Решение:
`#include<stdio.h>`

`struct element { // структура, задающая элемент списка`

42

```
int data;// поле для хранения данных
element *next;// указатель на следующий элемент
} *start, *p;/определяем глобальные указатели на начало списка и текущий элемент
void create_list(int);
int main()
{
    int a;
    printf("Введите значение для первого элемента, создаваемого списка: ");
    scanf("%d", &a);
    create_list(a);
    printf("список создан, единственный элемент: %i \n", start->data);
    return 0;
}

void create_list(int a)
{
    p=new element;
    p->data=a;
    p->next=NULL;
    start=p;
    return;
}

Пример 2. Написать функцию, добавляющую элемент в уже существующий список следом за текущим элементом. Значение нового элемента передается как параметр функции.

Решение:
void new_element(int a)
{
    element *q;
    q=new element;
    q->data=a;
    q->next=p->next;
    p->next=q;
    return;
}

Пример 3. Написать функцию, удаляющую текущий элемент списка.

Решение:
void remove_element()
{
if(p == start){start=p->next;
}
}

43
```

```
delete p;
p=start;
return;
}

element *q;
q=start;
while(q->next != p){
    q=q->next;
}
q->next = p->next;
delete p;
p=q;
return;
}

Пример 4. Пример циклического списка (задача Иосифа)
Предположим, N человек решили выбрать главаря. Для этого они встали в круг и стали удалять каждого M-го человека в определенном направлении отсчета, смыкая ряды после каждого удаления. Задача состоит в определении, кто останется последним (потенциальный лидер с математическими способностями заранее определил выигрышную позицию в круге).

Решение:
Для представления людей, расположенных в круге, построим циклический связанный список, где каждый элемент (человек) содержит ссылку на соседний элемент против хода стрелки. Сначала создается список элементов от 1 до N. Для этого создается циклический список с единственным узлом для участника 1, затем устанавливаются узлы для участников от 2 до N с помощью цикла. Затем в списке отсчитывается M-1 элемент и удаляется следующий (устанавливаем значение ссылки (M-1)-го узла таким образом, чтобы пропустить M-ый узел). Этот процесс продолжается до тех пор, пока не останется только один узел (который будет указывать на самого себя).

#include <stdio.h>
struct node
{
    int item;
    node* next;
};

typedef node *link;
int main()
{
    int i, N, M;
    printf("сколько в банде?\n");
    scanf("%d", &N);
    printf("какой по счёту?\n");
    scanf("%d", &M);
    link l = new node;
    l->item = 1;
    l->next = l;
}

44
```

link x = t;
for (i = 1; i < N; i++) { link q = new node; q->item = i; q->next = t;
x = (x->next = q); }

```
while (x != x->next) {
    for (i = 1; i < M; i++) x = x->next;
    x->next = x->next->next;
    printf ("Остался %i -й человек \n", x->item);
    return 0;
}
```

Уровень А:

В задачах 5.1.А - 5.4.А надо написать программы, автоматически создающую линейный односвязный список, состоящий из N элементов. Значения элементов вводятся с консоли пользователем.

5.1.А. Напишите функцию, показывающую значение текущего элемента (пользователь указывает номер элемента).

5.2.А. Напишите функцию, показывающую значение элемента, предшествующего текущему (пользователь указывает номер элемента).

5.3.А. Напишите программу, удаляющую n элементов, начиная с текущего элемента (пользователь указывает номер элемента). Оставшиеся выводятся на экран.

5.4.А. Напишите программу, вычисляющую сумму значений n элементов, начиная с текущего (пользователь указывает номер элемента).

5.5.А. На основе примера 4 напишите программу, в которой создание циклического списка, удаление его элементов, осуществлялись бы в отдельных функциях, поле item предназначалось для хранения фамилии человека, опрос с консоли.

Уровень В:

5.1.В. Напишите функцию, которая возвращает количество узлов циклического списка для данного указателя одного из узлов списка.

5.2.В. Напишите функцию, которая определяет количество узлов в циклическом списке, находящихся между узлами, на которые ссылаются два данных указателя x и z.

5.3.В. Напишите функцию, которая по указателям x и z двух непересекающихся связных списков вставляет список, указываемый l, в список, указываемый x, в позицию, которая следует после узла x.

5.4.В. Для данных указателей x и z узлов циклического списка напишите функцию, которая переносит узел, следующий после l, в позицию списка, которая следует после узла x.

5.5.В. Тип элемента списка определён так:

```
struct list {
    int val;
    list *next;
};

Создаётся статический список:
list a={3, NULL}, b={2, &a}, c={1, &b}, *ph=&c;
Дана функция
int FF (list *z)
{
    int k; list *q;
    if(z == NULL) return 0;
    for (q=z, z=z->next, k=1; z!=q; k++, z=z->next);
    return k;
}
```

Определите выполняемое действие над списком, напишите вызов данной функции для

45

данного статического списка.

5.6B. Использовать программу примера 4 с целью определения значения функции Иосифа для $M=2, 3, 5, 10$ и $N = 1000, 10000, 100000$ и 1000000 .

5.7B. Тип элемента списка определён так:

```
struct list {
    int val;
    list * next;
}
```

Создаётся статический список:

```
list a={3, NULL}, b={2, &a}, c={1, &b}, *ph=&c;
```

Дана функция

```
void fukaz_zag (list **zag, int v)
{ list *q = new list;
  q->val=v;
  q->next=&zag;
  *zag=q;
}
```

Определите выполняемое действие над списком, напишите вызов данной функции для данного статического списка.

5.8B. Тип элемента списка определён так:

```
struct list {
    int val;
    list * next;
}
```

Создаётся статический список:

```
list a={3, NULL}, b={2, &a}, c={1, &b}, *ph=&c;
```

Дана функция

```
void fssil_ukaz(list *zag, int v)
{ list *q = new list;
  q->val=v;
  q->next=zag;
  zag=q;
}
```

Определите выполняемое действие над списком, напишите вызов данной функции для данного статического списка.

5.9B. Тип элемента списка определён так:

```
struct list {
    int val;
    list * next;
}
```

Создаётся статический список:

```
list a={3, NULL}, b={2, &a}, c={1, &b}, *ph=&c;
```

Дана функция

```
void fssil_ukaz(list *zag, int v)
{ list *q = new list;
  q->val=v;
  q->next=zag;
  zag=q;
}
```

Определите выполняемое действие над списком, напишите вызов данной функции для данного статического списка.

5.10B. Тип элемента списка определён так:

```
struct list {
    int val;
    list * next;
}
```

Создаётся статический список:

```
list a={3, NULL}, b={2, &a}, c={1, &b}, *ph=&c;
```

Дана функция

```
int f (list *z)
{ int n;
  for (n=0; z!=NULL; z=z->next, n++);
  return n;
}
```

Определите выполняемое действие над списком, напишите вызов данной функции для данного статического списка.

5.11B. Тип элемента списка определён так:

```
struct list {
    int val;
    list * next;
}
```

Создаётся статический список:

```
list a={3, NULL}, b={2, &a}, c={1, &b}, *ph=&c;
```

Дана функция

```
list* fun (list *z, int v)
{ list *p, *q = new list;
  p->val=v;
  p->next=q;
  if (z == NULL) return q;
  for (p=z; p->next!=NULL; p=p->next);
  p->next=q;
  return z;
}
```

Определите выполняемое действие над списком, напишите вызов данной функции для данного статического списка.

5.12B. Тип элемента списка определён так:

```
struct list {
    int val;
    list * next;
}
```

Создаётся статический список:

```
list a={3, NULL}, b={2, &a}, c={1, &b}, *ph=&c;
```

Дана функция

```
list* funk (list *z, int n)
{ list *p, *q, *pr;
  for (p=z; p!=NULL; n-=1, pr=p, p=p->next);
  if (p == NULL) return z;
  if (pr == NULL) { q=z->next; }
  else { q=p; pr->next=p->next; }
  delete q;
  return z;
}
```

Определите выполняемое действие над списком, напишите вызов данной функции для данного статического списка.

Уровень C:

Надо написать программу, автоматически создающую односвязный список, состоящий из заданного количества элементов. Значения элементов вводятся с консоли пользователем.

5.1C. Поиск первого элемента в линейном списке, совпадающего с заданным числом.

5.2C. Поиск минимального элемента в линейном списке.

5.3C. Поиск максимального элемента в линейном списке.

5.4C. Сравнение элементов двух линейных списков с занесением одинаковых в третий линейный список.

5.5C. Поиск первого элемента в циклическом списке, совпадающего с заданным числом.

5.6C. Поиск минимального элемента в циклическом списке.

5.7C. Поиск максимального элемента в циклическом списке.

46

47

5.8 C. Сравнения элементов двух циклических списков с занесением одинаковых в третий циклический список.

Тема 6. Двусвязный список

Пример 1. Написать функцию, добавляющую в двусвязный список, чьи элементы уже упорядочены по убыванию, новый элемент с сохранением порядка. Необходимо предусмотреть случаи вставки в пустой список, в начало, в конец и в середину списка.

Решение

/Тип элемента списка определён так:

```
struct list {
    int val;
    list * next, * pred;
}
```

//Функция вставки:

```
void* ins (list *ph, int v)
{ list *q, *p = new list; //новый элемент списка
  p->val=v;
  p->next=q->pred=NULL;
  //-----включение в пустой список
  if (ph == NULL) { ph=p; return; }
  // поиск места вставки (сохраняется в q)
  for (q=ph; q!=NULL && v>q->val; q=q->next);
  //-----вставка в конец списка
  if (q == NULL) { // приходится восстанавливать указатель на последний
    for (q=ph; q->next!=NULL; q=q->next);
    p->pred=q;
    q->next=p;
    return p;
  }
  //-----вставка перед текущим
  p->next=q; // следующий за новым - текущий
  p->pred=q->pred; // предыдущий нового - это предыдущий текущего
  // если это начало списка
  if (q->pred == NULL) ph=p;
  else { q->pred->next=p; } // включение в середину
  q->pred=p; // предыдущий текущего - новый
}
```

Уровень А:

В задачах 6.1А -6.4А надо написать программу, автоматически создающую линейный двусвязный список, состоящий из 8 элементов. Значения элементов вводятся с консоли пользователем.

6.1A. Напишите функцию, осуществляющую просмотр списка слева направо (пользователь поочередно подтверждает вывод на экран).

6.2A. Напишите функцию, осуществляющую просмотр списка справа налево (пользователь поочередно подтверждает вывод на экран).

6.3A. Напишите программу, удаляющую n элементов, начиная с текущего элемента (пользователь указывает номер элемента). Оставшиеся выводятся на экран.

6.4A. Напишите программу, вычисляющую сумму значений n элементов, начиная с текущего (пользователь указывает номер элемента).

48

Уровень В:

6.1B. Тип элемента списка определён так:

```
struct list {
    int val;
    list * next, * pred;
}
```

Создаётся статический список из трёх элементов:

```
list a, b, c;
list a={3,&b, NULL}, b={2, &c, &a}, c={1, NULL,&b}, *ph=&a;
```

Дана функция

```
void vstavok (list **z, int n)
{ list *p, *q=new list;
  q->val=n; q->next=q->pred=NULL;
  if (*z == NULL) { *z=p; return; }
  for (p=*z; p->next != NULL; p=p->next);
  p->next=q; q->pred=p;
}
```

Определите выполняемое действие над списком, напишите вызов данной функции для данного статического списка.

6.2B. Тип элемента списка определён так:

```
struct list {
    int val;
    list * next, * pred;
}
```

Создаётся статический список из трёх элементов:

```
list a, b, c;
list a={3,&b, NULL}, b={2, &c, &a}, c={1, NULL,&b}, *ph=&a;
```

Дана функция

```
list * funn (list *z, int n)
{ list *q=new list;
```

```
q->val=n; q->next=q->pred=q;
if (z == NULL) z=q;
else{
  q->next=z; q->pred=z->pred;
  z->pred=q; z=q;
}
```

return z;

Определите выполняемое действие над списком, напишите вызов данной функции для данного статического списка.

6.3B. Тип элемента списка определён так:

```
struct list {
    int val;
    list * next, * pred;
}
```

Создаётся статический список из трёх элементов:

```
list a, b, c;
list a={3,&b, NULL}, b={2, &c, &a}, c={1, NULL,&b}, *ph=&a;
```

Дана функция

```
list * fun (list *z, int n)
{ list *q;
```

```
for (q=z; n!=0; q=q->next, n--);
if (q->next==q){delete q; return NULL;}
```

```
if (q == z) z=q->next;
```

49

и высокого уровня – являются важнейшими с точки зрения

важности для дальнейшего профессионального становления и профессионального мастерства будущих специалистов в сфере информационных технологий.

Светлана Юрьевна Белим

Задания по программированию на языке высокого уровня.

(Учебно-методическое пособие)

Авторское редактирование

При подготовке к занятиям по курсу «Программирование на языке высокого уровня» вузом «Санкт-Петербургский государственный политехнический университет Петра Великого» в 2008 году было создано учебное пособие «Задания по программированию на языке высокого уровня». Пособие было разработано на основе лекционных материалов кафедры ИУТ и практикующими инженерами. В 2010 году в связи с изменениями в структуре и содержании курса было решено пересмотреть пособие. В результате проведенного пересмотра и дополнения пособия оно получило название «Задания по программированию на языке высокого уровня».