Язык программирования C

Овсянникова Т.В.

9 декабря 2013 г.

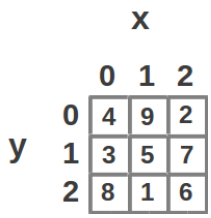
1 Программа на C.

1.1 Двумерные массивы. Файлы.

Определение. Магический квадрат

Магический, или волшебный квадрат — это квадратная таблица $n \times n$, заполненная n^2 числами, таким образом, что сумма чисел в каждой строке, каждом столбце и на обеих диагоналях оказывается одинаковой. Нормальным называется магический квадрат, заполненный целыми числами от 1 до n^2.

Задача C.1. Дан квадрат размером 3×3 заполненный числами от 1 до 9.



Написать программу, которая выясняет является ли данный квадрат магическим.

Решение: Для представления квадрата можно использовать двумерный массив. В языке C он описывается так:

```
char m1[3][2] \\ двумерный массив символов
              \\ 3 строки по 2 колонки
int dm[10][10] \\двумерный численный массив
...
m1[2][1]='*' \\присваивание
```

Для решения нужно прочитать числа квадрата, подсчитать и сравнить все необходимые суммы.

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

int main(){
/*
 * двумерный массив kvadrat
 * для целых чисел 3 строчки по 3 столбца
*/
  int kvadrat[3][3];
  int z;
/*
 * переменная ok равна 0 пока не установлено,
 * что сумма чисел в каком-нибудь столбце или
 * строке не равна сумме в остальных
*/
  int ok=0;
/*
 *координаты чисел в массиве: y - номер строки,
  x - номер столбца
*/
  int y,x;

  for(y=0;y<3;y++){
    for(x=0;x<3;x++){
/*
 * Читаем каждое следующее число
 * и присваиваем его элемету массива
*/
      scanf("%d",&z);
      kvadrat[y][x]=z;
    }
  };

/*
 * Напечатали квадрат для проверки
 */
  for(y=0;y<3;y++){
    for(x=0;x<3;x++){
```

```
      printf("%d ",kvadrat[y][x]);
    }
    printf("\n");
  };

/*
 * csum - это сумма с которой будем сравнивать
 * sum - сумма чисел в строке с номером y
 */
  int csum=0,sum=0;

  for(y=0;y<3;y++){
    csum=0;
    for(x=0;x<3;x++){
       csum+=kvadrat[y][x];
    }
    if(y==0)
      sum=csum;
    else{
/*
 * Как только суммы не равны, ok присваиваем 1
 * и выходим из цикла
 */
       if (sum!=csum){
        ok=1;
        break;
       }
    }
  }
  printf("ok=%d\n",ok);

/*
 * Такой же цикл, только считаем сумму
 * в столбцах
*/
  for(x=0;x<3;x++){
    csum=0;
    for(y=0;y<3;y++){
       csum+=kvadrat[y][x];
    }
    if(x0)
```

```
        sum=csum;
      else{
         if (sum!=csum){
          ok=1;
          break;
         }
      }
   }

 printf("ok=%d\n",ok);
 csum=sum=0;
/*
 * Подсчет суммы чисел на диагонали
 * верхний левый угол <-> правый нижний
*/
   for(x=0;x<3;x++){
      csum+=kvadrat[x][x];
   }
/*
 * Подсчет суммы чисел на диагонали
 * верхний правый угол <-> левый нижний
*/
   for(x=0;x<3;x++){
      sum+=kvadrat[x][2-x];
   }
   printf("sum=%d␣csum=%d\n",sum,csum);

/*
 * Проверяем магический квадрат или нет
 * Если ok не стал 1 и суммы по двум
 * диагоналям равны, значит магический
*/
   if(ok!=1 && csum==sum)
     printf("magic\n");
   else
     printf("no\n");
   return 0;
}
```

Задача, которую мы решали требует введения только 9 чисел. Однако каждый раз вручную вводить в программу большое количество данных

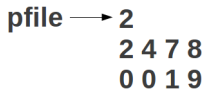
неудобно и долго. Чтобы избежать этого можно воспользоваться данными, записанными в файл.

Для работы с файлом используется **файловый указатель**. Он связывается с конкретным файлом, а затем используется для чтения данных из файла или записи в него.

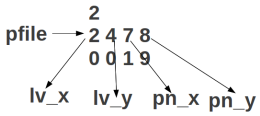
Рассмотрим файл **my.txt**. Пусть в нем лежат только числа: первое число - количество строк, далее строки по 4 числа в каждой

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

int main(){
// Описание файлового указателя
 FILE *pfile;
//Открытие файла my.txt для чтения
 pfile=fopen("my.txt","r");
```

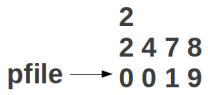



```c
 int n;
int lv_x,lv_y,pn_x,pn_y;
// Чтение первого числа
 fscanf(pfile,"%d",&n);
//Файловый указатель "смотрит"
// на следующее за ним число
// Чтение  четырех чисел
fscanf(pfile,"%d%d%",&lv_x,&lv_y);
fscanf(pfile,"%d%d",&pn_x,&pn_y);
```




```c
// n=2
//lv_x=2, lv_y=4,pn_x=7,pn_y=8
//Файловый указатель "смотрит"
//на следующую строчку

// закрываем файл
  close(pfile);
 return 0;
}
```



Задача C.2. В файле заданы координаты левого верхнего и правого нижнего угла прямоугольника. Все четыре числа записаны в одну строку: сначала координаты левого верхнего угла, затем правого нижнего.

Например:

2 1 8 8

Требуется «изобразить» этот прямоугольник на рисунке 10×10 сим-

волов. При этом символ «.» - белый цвет, а символ «*» - черный.

Напримммер для координат (2,1) и (8,8) :

```
......
..***.
..***.
..***.
..***.
......
```

Решение:

```
#include <stdio.h>
#include <stdlib.h>
#include <errno.h>
/*
 * глобальный двумерный массив pict
 * для символов
 * 10 строчек по 10 столбцов
*/
char pict[10][10];

  void write(){
     int x,y;
     for(y=0;y<10;y++){
       for(x=0;x<10;x++)
         printf("%c",pict[y][x]);
       printf("\n");
     }
  } ;

int main(){
/*
 * Координаты левого верхнего и
 * правого нижнего угла прямоугольника
*/
 int lv_x,lv_y,pn_x,pn_y;

  int i,x,y;
 /*
 * Чтобы прочитать содержимое файла нужно:
 * 1) объявить файловый
 *    указатель (назовем его pfile)
```

```
*/
  FILE *pfile;
/*
 * 2) присоединить файловый указатель к файлу
 *    pfile -> coord.txt
 *    для этого, например, нужно открыть
 *    файл "на чтение"
*/
   pfile=fopen("coord.txt","r");
  if(errno){
    perror(":(");
  }

/*
 В строчке записаны по 4 числа:
 * координаты левого верхнего угла и
 * правого нижнего
*/

/*
 * Читаем  координаты углов
 *  прямоугольника:
*/

     fscanf(pfile,"%d%d%d%d",&lv_x,&lv_y,&pn_x,&pn_y);

/*
* "Нарисуем" прямоугольник
*  Сначала заполним массив ".",
*  а затем "*" обозначим прямоугольник
*/
   for(y=0;y<10;y++)
     for(x=0;x<10;x++){
       pict[y][x]='.';
    }

/*
 * "Рисование" прямоугольника:
*/
   for(y=lv_y;y<=pn_y;y++)
     for(x=lv_x;x<=pn_x;x++)
```

```
        pict[y][x]='*';

    write();
return 0;
}
```

Задача C.3. В файле записаны числа (неизвестно сколько). Известно, что их сумма не превышает 2^{63}.

Написать программу, которая подсчитывает среднее арифметическое этих чисел.

Решение: Для решения этой задачи не нужно запоминать в массив все прочитанные числа. Их можно сразу складывать и подсчитывать сколько чисел удалось прочитать.

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

int main(){
  long long sum=0;
  int d;
// Количество чисел
  int n=0;
// Для среднего значения
  float aver;

  FILE *pfile;
// Открытие файла
  pfile=fopen("chisla","r");
  if(errno){
    perror(":(");
    exit(1);
  }
/*
 fscanf может читать из файла числа пока не достигнут конец
 или вдруг там записано что-то другое.
 Когда встречается конец файла (EOF) fscanf его проверяет.
 Таким образом можно читать файл неизвестного размера
*/
  while(fscanf(pfile,"%d",&d)!=EOF){
```

```
      sum+=d;
      n++;
   }
   aver=(float)sum/n;
   printf("среднее: %0.2f\n",aver);
 return 0;
}
```

Задача C.4. Изображение задано в файле. В первой строке заданы два целых числа (N и M) - размер изображения в символах. Далее в файл записаны N строк по M символов в каждой. Символ «.» - белый цвет, символ «*» - черный. Черным «нарисованы» прямоугольники. Никакие прямоугольники при этом не пересекаются и не соприкасаются. Одинокая закрашенная черным клетка тоже считается прямоугольником.

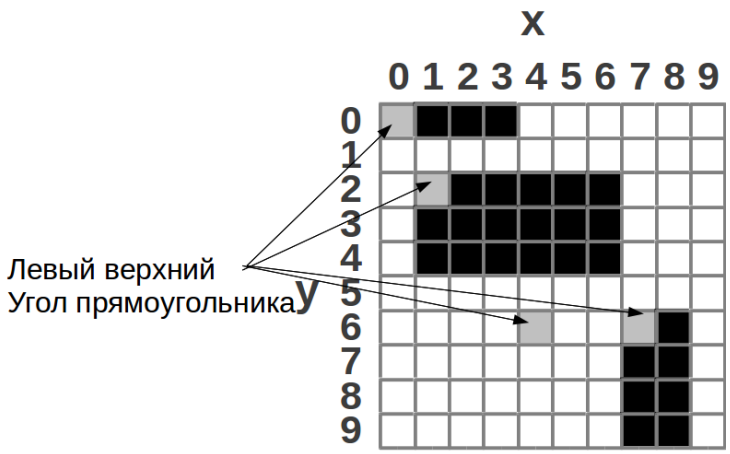
Написать программу, которая подсчитывает сколько всего черных прямоугольников изображено на рисунке.

Решение:

Заметим, что прямоугольники можно считать по встретившимся левым верхним углам. Можно подсчитать все встретившиеся левые верхние углы.

Определение. Левым верхним углом прямоугольника *является черная клетка если:*

- *она имеет координату (0,0);*
- *она расположена в ряду с номером* 0 *и клетка слева (координата x меньше) - белая;*
- *она расположена в столбце с номером* 0 *и клетка сверху (координата y меньше) - белая;*
- *она имеет координаты (y,x), а клетки с координатами (y,x-1) и (y-1,x) - белые ;*



Решение:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <errno.h>
/*
 * глобальный двумерный массив pict
 * для символов
 * 10 строчек по 10 столбцов
*/
char pict[10][10];
/* Функция, копирующая символы из строки buf в
   заданную строку (y) массива pict
*/
void copy(int y, int m, char *buf){
  int i;
  for (i=0;i<m;i++)
    pict[y][i]=buf[i];
};

// Функция печати массива pict

void write(){
    int x,y;
  for(y=0;y<10;y++){
    for(x=0;x<10;x++)
      printf("%c",pict[y][x]);
```

```
    printf("\n");
  }
};

int main(){
// Строка символов для считывания из файла
  char buf[11];

  int x,y,i;
// Размер картинки
  int N,M;

// Файловый указатель
  FILE *pfile;
// Открываем файл для чтения
  pfile=fopen("pict.dat","r");
  if(errno){
    perror(":(");
    exit(1);
  }

/*
Считываем первые два числа - размер картинки
*/
  fscanf(pfile,"%d%d",&N,&M);

/*
 Считываем из файла N  строк
*/
  for(y=0;y<N;y++){
// Чтение строки из файла
    fscanf(pfile,"%s",buf);
// Копируем то, что прочитали в массив
    copy(y,M,buf);
  };

// Напечатали картинку для проверки
  write();
// Ищем углы прямоугольников:
 int count=0;
// Проверка первого символа рисунка
```

```
  if (pict[0][0]=='*'){
    count++;
   }
// Проверка первой строки
  for (x=1;x<M;x++)
    if(pict[0][x]=='*' && pict[0][x-1]=='.'){
      count++;
    }

// Проверка остального рисунка
   for(y=1;y<N;y++){
      for(x=0;x<M;x++){
        if(pict[y][x]=='*')
// Проверка первого столбца
          if (x==0){
            if (pict[y-1][x]=='.')
              count++;
          }else{
// Проверка остального рисунка
             if(pict[y][x-1]=='.' && pict[y-1][x]=='.'){
               count++;
             }
          }
      }
   }

  printf("n=%d\n",count);

  return 0;
}
```

Задачи для самостоятельного решения

Задача C.5. (*) В файле (**numbers.txt**) записаны целые числа.

Написать программу, которая печатает минимальное число из файла.

Задача C.6. (*) Изображение задано в файле. В первой строке заданы два целых числа (N и M) - размер изображения в символах. Далее в файл записаны N строк по M символов в каждой. Символ «.» - белый цвет, символ «*» - черный.

Написать программу, которая выясняет каких клеток больше бе-

лых или черных.

Задача C.7. В файле записана позиция игры «крестики-нолики».

Написать программу, которая выясняет закончена ли игра (выиграли крестики, выиграли нолики или все клеточки заполнены и ничья) или можно сделать ход. Если игра закончена, программа пишет кто выиграл или ничья, а если нет, то предлагает сделать ход.

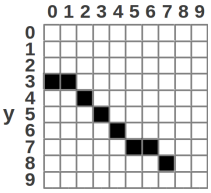
Задача C.8. В файле заданы начальная и конечная координаты отрезка, нарисованного на клетчатом поле размером 10×10 клеток. Уравнение прямой для координат задается формулой

$$y = \frac{y_2 - y_1}{x_2 - x_1} x + \frac{x_2 y_1 - x_1 y_2}{x_2 - x_1}$$

.

Написать программу, которая «закрашивает» черным («*») все оставшиеся клетки отрезка. При вычислении координаты точки округлять по правилам округления.

Например, для координат (0,3) и (7,8) рисунок будет такой:



Задача C.9. Изображение задано в файле. В первой строке заданы два целых числа (N и M) - размер изображения в символах. Далее в файл записаны N строк по M символов в каждой. Символ «.» - белый цвет, символ «*» - черный. Черным «нарисованы» прямоугольники. Никакие прямоугольники при этом не пересекаются и не соприкасаются. Одинокая закрашенная черным клетка тоже считается прямоугольником.

Написать программу, которая печатает координаты верхних левых и углов и правых нижних для всех прямоугольников.

1.2 Структуры данных (записи)

Задача C.10. Дана карта звездного неба. Звезды на ней имеют свое название, цвет и многие другие свойства. Кроме того у каждой звезды есть двумерные координаты (относительно карты). Точка наблюдения на карте имеет координаты $(0, 0)$

Написать программу, которая вычисляет расстояние между двумя звездами на карте и выводит его с указанием названия звезд и их цветов.

Решение: Заметим, что в этой задаче для каждой звезды необходимо запоминать множество значений, а значит использовать множество переменных. Языки программирования позволяют собрать все свойства объекта (в данном случае звезды) в одну сложную преременную, называемую в C структурой (struct).

Определение. Структура

Структура - это ТИП переменных, который программист создает самостоятельно. Для каждой структуры определяются ***поля****: их количество, тип*

В языке C для задачи про звезды тип переменной, описывающий все свойства звезды может быть определен например так:

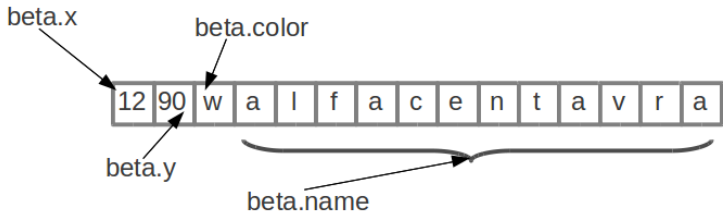
```
typedef struct{
    int x,y; //Координаты звезды на карте.
    char color;  //Цвет звезды
    char name[100];  //Название звезды
 } Star;
```

Таким образом опредлен только тип переменной - это еще не переменная, это только новое понятие такое же ка и *int*, *float*, *char* и т.д.

Чтобы пользоваться переменной этого типа ее нужно объявить как и все остальные пременные:

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
//Объявление нового типа переменных
 typedef struct{
    int x,y; //Координаты звезды на карте.
    char color;  //Цвет звезды
    char name[100];  // Название звезды
 } Star;
```

```
int main(){
//Объявление переменных типа Star (созданный нами тип)
  Star beta,vega;
/*
  Переменные beta и vega имеют поля:
  x,y - целого типа
  color - типа char
  name - типа string
*/
```




```
  float rasst;

// Присваивание значений полям переменной beta.
  beta.x=0;
  beta.y=0;
  beta.color='w';

//Чтение значений полей переменной vega.
  scanf("%d%d",&(vega.x),&(vega.y),&(vega.color));
/*
  Заметим, что vega и  beta - разные переменные. Поле x
  переменной vega и поле x переменной beta - РАЗНЫЕ.
  ДЛЯ КАЖДОЙ ПЕРЕМЕННОЙ - СВОИ
*/

//Вычисление рассояния между двумя звездами.
  rasst=(vega.x-beta.x)*(vega.x-beta.x)+(vega.y-beta.y)*(vega.y-beta.y);
  rasst=sqrt(rasst);
  printf("%0.2f между звездой %s цвета %c", rasst,beta.name,beta.color);
  printf( " и звездой %s цвета %c\n",vega.name, vega.color);
  return 0;
}
```

Таким образом объявленные переменные можно присваивать друг другу.

```
//Объявление нового типа переменных
```

```
 typedef struct{
     int x,y; //Координаты звезды на карте.
     char color;  //Цвет звезды
     char name[100]; // Название звезды
  } Star;

int main(){
//Объявление переменных типа Star (созданный нами тип)
  Star:beta,vega;
begin
    scanf("%d%d%c\n",&(vega.x),&(vega.y),&(vega.color));
//Теперь присвоим vega ->beta
  beta=vega
  return 0;
}
```

В этом случае все поля переменной *beta* примут значения, которые имеют поля переменной vega. То есть одной операцией копирования удалось скопировать столько же занчений, сколько было бы в четырех отдельных переменных

Можно создавать массивы из переменных, описанных как «структура»:

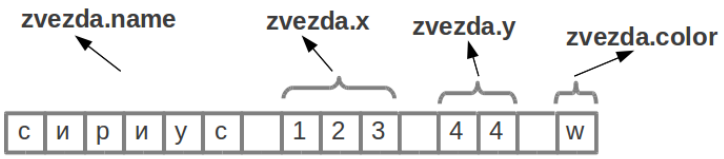
```
...
  Star nebo[4];
...
// Обращение к элементам такого массива:
   nebo[0].x=10;
  i=3;
  nebo[i].color='g';
```

Задача C.11. В файле записано информация о звездах (с карты):

4	Количество звезд (дальше будут 4 строки)
alpha_centavra 4 12 w	название звезды x y цвет
algol 10 33 w	
polarnaya 0 0 w	
vega 31 22 c	

Написать программу, которая читает информацию из файла и печатает ее на экран.

Решение: Заметим, что все записано в одну строку, причем вначале строки идет текст - название звезды. Значит нужно разбить строку на: текст, число, число и символ.




```c
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
/*
   Объявление нового типа переменных (struct)
   для хранения информации о звездах
*/
typedef struct{
      int x,y; //Координаты звезды на карте.
      char color;  //Цвет звезды
      char name[100];  // Название звезды
   }Star;

int main(){

   FILE *fil;
   float rasst;
   char buf[100];
   int i;
   int n;
   int ok;
   Star beta;
// Объявление массива переменных типа Star
   Star nebo[4];

   fil=fopen("stars.dat","r");

   for(i=0;i<4;i++){
// Чтение информации и строки файла
      fscanf(fil,"%s%d%d␣%c\n",z.name,&(z.x),&(z.y),&(z.color));
// Присваивание элементу массива (тип Star)
// прочитанной перменной z (тип Star)
    nebo[i]=z;
    printf("%s␣%d␣",nebo[i].name,nebo[i].x);
  // Печать значений на экран.
```

```
    printf("%d␣%c\n",nebo[i].y,nebo[i].color);
  }
  return 0;
}
```

Задача C.12. Два треугольника заданы своими координатами.

Написать программу, которая выясняет равны ли эти треугольники.

Решение: Рассмотрим часть решения задачи. Введем переменную «треугольник». Тогда при сравнении можно пользоваться представлениями: «вершина треугольника», «сторона треугольника» и т.д.

```
#include <stdio.h>
#include <stdlib.h>
// Опишем тип Vershina - координаты вершин треугольника
typedef struct{
  x,y:int;
}Vershina;

// Тип Triangle - треугольник
// Состоит из вершин и длин сторон.
 typedef struct{
    Vershina ugl[3];
    int storona[3];
}Triangle

int main(){
//Опишем два треугольника - синий и красный
  Triangle siniy,kracny;
  int x2,y2;

int main()

// Обращение к координатам первой вершины синего
    siniy.ugl[0].x=1;
    siniy.ugl[0].y=3;
// Обращение к координатам второй вершины синего
    siniy.ugl[1].x=0;
    siniy.ugl[1].y=0;
// Обращение к координатам первой вершины красного
    kracny.ugl[2].x=3;
    kracny.ugl[2].y=5;
```

```
// Подсчет квадрата длины первой строны синего
    x2=(siniy.ugl[0].x-siniy.ugl[1].x);
    y2=(siniy.ugl[0].y-siniy.ugl[1].y);
    siniy.st[0]=x2*x2+y2*y2;
    printf("%d\n",siniy.st[0]);

end.
```

Задачи для самостоятельного решения

Задача C.13. Доделать предыдущую задачу. Сравнить два треугольника.

Задача C.14. Прямоугольник на плоскости задается своими вершинами: левый верхний угол и правый нижний. Кроме того, заданы координаты N (≤ 100) точек.

Написать программу, которая выясняет сколько точек лежит внутри прямоугольника.

Задача C.15. Точки на плоскости (не более 100) заданы своими координатами (x,y - целые числа). Найти координаты двух наиболее близко расположенных точек.

Задача C.16. Рациональная дробь - это целая часть, числитель и знаменатель. Написать функции сложения, вычитания, умножени и деления дробей в виде:

```
// Тип Drob для хранения времени
 typedef struct{
   int cl,ch,zn;
  }Drob;
...
Drob add(Drob a,Drob b); // сложения
Drob sub(Drob a,Drob b);// вычитания
Drob mult(Drob a,Drob b):Drob;// умножения
Drob divis(Drob a,Drob b);// деления
```

Дроби представить в несократимом виде. Для этого написать функцию поиска наибольшего общего делителя