ЛЕКЦИИ 11 – 12

Сложные типы данных: массивы

Сложные (векторные) типы данных

Значения сложного типа могут состоять из нескольких значений одного или различных типов данных (как простых, так и сложных).

В языке С присутствуют следующие сложные типы:

- массивы,
- строки,
- перечисления,
- структуры,
- объединения,
- поля бит.

Массивы

Массив – это сложный тип данных, представляющий собой упорядоченную совокупность элементов одного типа.

Упорядоченность проявляется в том, что доступ к каждому элементу массива осуществляется посредством его индекса (номера) в массиве. Индекс представляет собой одно или несколько целых чисел, в зависимости от размерности массива.

Массивы

Объявление массивов на языке С имеет следующий синтаксис:

тип имя[предел №1][предел №2]...[предел №N];

Примеры:

int a[10];

double b[5][20];

char c[5][5][10];

Массивы

Объявление массива в языке С является обычным оператором объявления, поэтому в одном операторе допускается объявлять несколько массивов и обычных переменных, например:

```
double x[5][10], y[10][10];
int a[10], i, j;
```

Обращение к элементам массива в языке С осуществляется путем указания имени массива и, следом за ним, индексов элемента в парных квадратных скобках. Индексация в языке С начинается с нуля.

Поэтому, в массиве *a* обращение к первому элементу будет иметь вид: *a[0]*, а к последнему – *a[9]*.

Соответственно в массиве *b*: *b[0][0]* и *b[4][19]*.

Массивы

На практике наиболее часто используются только:

- одномерные массивы (векторы);
- двумерные массивы (матрицы).

Одномерные массивы

Объявление одномерного массива (далее просто массив) имеет следующий синтаксис:

тип имя[размер];

В качестве размера массива может указываться любое положительное целочисленное значение. В стандарте **C89** значение могло являться только константой. В стандарте **C99** было введено понятие динамического массива.

Одномерные массивы

Динамический массив - массив, при создании которого в качестве размера указывается значение некоторого выражения, в которое входят переменные, объявленные и инициализированные ранее (выражение должно иметь положительный целочисленный результат).

Например:

```
int n;
printf(“Введите размер массива: ”);
scanf(“%d”,&n);
double x[n];
```

Одномерные массивы

При объявлении статических массивов допускается производить инициализацию элементов массива. Синтаксис такого объявления:

тип имя[размер] = {значение №1, ... значение №N};

Примеры:

int a[5] = **{**1,2,3,4,5**}**, b[5] = **{**1,2**}**;

double x[10] = **{**0.0**}**;

int c[] = {1,2,3,4,5};

char d[] = {’a’,’b’,’c’};

int e[5] = **{**1,2,,4,5**}**; //Неправильно - ошибка

Одномерные массивы

Объявление константных массивов (значения их элементов изменить нельзя) начинается с ключевого слова **const**, за которым следует объявление массива с инициализацией.

Примеры:

```
const int array[] = {1,2,3,4,5};
const double vector[5] = {1.0,2.0,3.0};
```

Одномерные массивы

Обращение к элементу массива осуществляется путем указания имени массива, а после имени в квадратных скобках индекса элемента:

имя[индекс]

Индексация в языке С начинается с нуля, поэтому для массива размером, например, десять элементов правильными будут индексы от нуля до девяти включительно.

Каждый отдельный элемент массива может рассматриваться как простая переменная и, соответственно, выступать в выражениях в качестве RValue или LValue значений.

Одномерные массивы

Ввод и вывод массивов в языке С осуществляется поэлементно в цикле. Например, ввод и вывод целочисленного массива из десяти элементов будет иметь вид:

```
int a[10];
for(int i=0;i<10;i++) scanf("%d",&a[i]);
...
for(int i=0;i<10;i++) printf("%d\t",a[i]);
```

Одномерные массивы

Присвоение массива массиву также осуществляется поэлементно. Например, необходимо присвоить вещественный массив *x* вещественному массиву *y*. Фрагмент программы:

```
double x[15], y[15];
...
for(int i=0;i<15;i++) y[i] = x[i];
...
```

Одномерные массивы

В языке С во время выполнения программы не производится контроль за допустимыми значениями индексов элементов. Поэтому, если индекс элемента выходит за рамки массива, то в программе возможно появление ошибок. Ошибки могут быть:

- простыми (например «случайное» изменение переменных);
- критическими (выход за пределы пространства памяти, отведенной для программы).

Например:

```
int a[10];
for(int i=0;i<=10;i++) a[i] = i;
```

Пример 1

Дан вещественный массив размера N. Размер массива и значения его элементов вводятся пользователем. Найти сумму всех локальных минимумов массива. Локальный минимум – элемент массива меньший по значению двух соседних элементов данного массива (исключая крайние элементы).

Пример 1

```
#include <stdio.h>
int main(int argc, char *argv[])
{
  int N;
  printf("Введите количество элементов: ");
  scanf("%d",&N);
  if(N<3) {printf("Мало элементов!\n"); return 0;}
  printf("Введите массив: ");
  double arr[N];
  for(int i=0;i<N;i++) scanf("%lf",&arr[i]);
  double summa = 0.0;
  for(int i=1;i<N-1;i++)
    if((arr[i]<arr[i-1])&&(arr[i]<arr[i+1]))
      summa += arr[i];
  printf("Сумма локальных минимумов: %.3lf\n",summa);
  return 0;
}
```

Пример 2

Дан целочисленный массив размера N. Найти элемент массива, значение которого наиболее близко к числу К и вывести его позицию на экран. Если таких элементов несколько, то только последний из них.

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

Пример 2

```
int main(int argc, char *argv[])
{
  int N,K;
  printf("Введите N: "); scanf("%d",&N);
  int arr[N];
  printf("Введите массив:\n");
  for(int i=0;i<N;i++) scanf("%d",&arr[i]);
  printf("Введите K: "); scanf("%d",&K);
  int pos = 0;
  for(int i=0;i<N;i++)
    if(abs(arr[i]-K)<=abs(arr[pos]-K)) pos = i;
  printf("Значение: %d, Позиция: %d\n",arr[pos],pos);
  return 0;
}
```

Пример 3

Дан целочисленный массив размера N. Размер массива и его элементы вводятся пользователем. Упорядочить все элементы, расположенные до максимального элемента массива по возрастанию, а после максимального элемента по убыванию.

Пример 3

```
#include <stdio.h>
int main(int argc, char *argv[])
{
  int N;
  printf("Введите количество элементов: ");
  scanf("%d",&N);
  if(N<3) {printf("Слишком мало элементов!\n"); return 0;}
  printf("Введите массив: ");
  int arr[N];
  for(int i=0;i<N;i++) scanf("%d",&arr[i]);
  int pos = 0;
  for(int i=1;i<N;i++)
    if(arr[i]>arr[pos])
      pos = i;
```

Пример 3

```
if(pos > 1){
    int flag = 1;
    while(flag){
      flag = 0;
        for(int i=0;i<pos-1;i++)
          if(arr[i]>arr[i+1]){
             int r = arr[i];
             arr[i] = arr[i+1];
             arr[i+1] = r;
             flag = 1;
        }
    }
  }
```

```
if(pos < N-2){
  int flag = 1;
  while(flag){
    flag = 0;
    for(int i=pos+1;i<N-1;i++)
      if(arr[i]<arr[i+1]){
        int r = arr[i];
        arr[i] = arr[i+1];
        arr[i+1] = r;
        flag = 1;
      }
  }
}
```

Пример 3

```
  printf("Результат: ");
  for(int i=0;i<N;i++) printf("%d ",arr[i]);
  printf("\n");
  return 0;
}
```

Пример 4

Дан вещественный массив размера N. Удалить из массива максимальный элемент. Если таких элементов несколько, то только первый из них.

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

Пример 4

```
int main(int argc, char *argv[])
{
  int N;
  printf("Введите N: "); scanf("%d",&N);
  double arr[N];
  printf("Введите массив:\n");
  for(int i=0;i<N;i++) scanf("%lf",&arr[i]);
  int pos = 0;
  for(int i=0;i<N;i++)
    if(arr[i]>arr[pos]) pos = i;
  for(int i=pos;i<N-1;i++) arr[i] = arr[i+1];
  N--;
  printf("Результат:\n");
  for(int i=0;i<N;i++) printf("%.2lf ",arr[i]);
  printf("\n");
  return 0;
}
```

Пример 5

Дан целочисленный массив размера N. Вставить перед вторым отрицательным элементом массива нулевое значение. Если такого элемента нет, то дописать нулевой элемент в конец массива.

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

Пример 5

```
int main(int argc, char *argv[])
{
  int N;
  printf("Input N: "); scanf("%d",&N);
  int arr[N+1];
  printf("Input array:\n");
  for(int i=0;i<N;i++) scanf("%d",&arr[i]);
  int pos = N;
  for(int i=0,j=0;i<N;i++){
    if(arr[i]<0) j++;
    if(j==2) {pos = i; break;}
  }
  if(pos!=N) for(int i=N;i>pos;i--) arr[i] = arr[i-1];
  arr[pos] = 0; N++;
  printf("Result:\n");
  for(int i=0;i<N;i++) printf("%d ",arr[i]);
  printf("\n");
  return 0;
}
```

Двумерный массив

Объявление двумерного массива (далее матрица) имеет следующий синтаксис:

тип имя[размер №1][размер №2];

Размеры матрицы указываются в отдельных парных квадратных скобках после имени и могут быть любыми положительными целочисленными значениями. На практике принято значение первой размерности называть строками, а второй – столбцами. Как и в случае одномерного массива, в стандарте **C89** регламентируется, что размеры матрицы должны быть целочисленными константами.

Двумерный массив

Стандарт **C99** допускает объявление динамических матриц, путем использования выражений при указании размеров матрицы, если в это выражение входят значения определенных ранее переменных (выражение должно иметь положительный целочисленный результат). Например:

```
int n,m;
printf(“Введите размеры матрицы: ”);
scanf(“%d %d”,&n,&m);
double a[n][m];
```

Двумерный массив

При объявлении матриц допускается производить инициализацию значений элементов матрицы:

```
тип имя[размер №1][размер №2] = {
  {значение № 11, ... значение № 1N},
  ...
  {значение № M1, ... значение № MN}
};
```

Двумерный массив

Примеры объявлений с инициализацией:

```
int a[2][4] = {  //Объявлена матрица
  {1,2,3,4},       //  1 2 3 4
  {5,6}};          //  5 6 0 0

double b[3][5] = {          //Объявлена матрица
  {1.0, 2.0, 3.0, 4.0, 5.0}, //   1 2 3 4 5
  {6.0, 7.0}                 //   6 7 0 0 0
};                           //   0 0 0 0 0
```

Двумерный массив

Пропускать значения инициализации строк нельзя. Например, следующий фрагмент кода программы неправильный:

```
int a[3][5] = {{1,2,3,4,5},,{6,7,8,9,0}};
```

Допускается не указывать количество строк в матрице (указываются пустые квадратные скобки). В таком случае размер массива будет определен по числу инициализирующих значений строк. Количество столбцов матрицы всегда необходимо указывать. Например:

```
double b[][4] = {{1,2,3,4},{5,6,7,8}};
```

Двумерный массив

Объявление константных матриц (значения их элементов изменить нельзя) начинается с ключевого слова **const**, за которым следует объявление матрицы с инициализацией. Пример:

```
const int matrix[][5] = {
  {1,2,3,4,5},
  {6,7,8,9}
};
```

Двумерный массив

Обращение к элементу матрицы осуществляется путем указания имени матрицы, а после имени в отдельных парных квадратных скобках индексы элемента (строка и столбец):

имя[строка][столбец]

Индексация в языке С начинается с нуля, поэтому для матрицы размером, например, пять строк и десять столбцов правильными будут индексы строк от нуля до четырех, а столбцов – от нуля до девяти включительно.

Каждый отдельный элемент матрицы может рассматриваться как простая переменная и, соответственно, выступать в выражениях в качестве RValue или LValue значений.

Двумерный массив

Ввод и вывод матриц в языке С осуществляется поэлементно. Так как матрица имеет двойную размерность, то ввод и вывод осуществляется во вложенных циклах. Например:

```
double a[5][10];
for(int i=0;i<5;i++)
  for(int j=0;j<10;j++)
    scanf("%lf",&a[i][j]);
...
for(int i=0;i<5;i++){
  for(int j=0;j<10;j++)
    printf("%8.2lf\t",a[i][j]);
  printf("\n");
}
```

Двумерный массив

Присвоение матрицы матрице также осуществляется поэлементно. Например, необходимо присвоить целочисленную матрицу *x* целочисленной матрице *y*. Фрагмент программы:

```
int x[5][10], y[5][10];
...
for(int i=0;i<5;i++)
  for(int j=0;j<10;j++)
    y[i][j] = x[i][j];
...
```

N-мерный массив

В языке С допускается создание массивов размерностью три и более. Например, объявление трехмерного целочисленного массива с инициализацией будет иметь вид:

```
int a[2][2][2]={
  {{1,2},{3,4}},
  {{5,6},{7,8}}
};
```

Ввод, вывод и прочая обработка такого массива осуществляется в трех вложенных циклах.

Пример 1

Дана квадратная вещественная матрица размера N. Вычислить среднее значение элементов матрицы, расположенных на главной диагонали и выше. Размер матрицы и ее элементы вводятся пользователем.

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

Пример 1

```
int main(int argc, char *argv[])
{
  int N;
  printf("Введите N: "); scanf("%d",&N);
  double arr[N][N];
  printf("Введите матрицу:\n");
  for(int i=0;i<N;i++)
    for(int j=0;j<N;j++)
      scanf("%lf",&arr[i][j]);
  double sum = 0.0;
  for(int i=0;i<N;i++) for(int j=i;j<N;j++) sum += arr[i][j];
  sum /= (N*(N+1)/2);
  printf("Среднее значение: %.2lf\n",sum);
  return 0;
}
```

Пример 2

Даны две вещественные матрицы. Вычислить произведение этих матриц, если это возможно. В противном случае вывести сообщение о невозможности перемножения матриц. Размеры матриц и их элементы вводятся пользователем.

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

Пример 2

```
int main(int argc, char *argv[])
{
  int N1, M1;
  printf("Введите размеры 1-ой матрицы: "); scanf("%d %d",&N1,&M1);
  double A[N1][M1];
  printf("Введите элементы 1-ой матрицы:\n");
  for(int i=0;i<N1;i++) for(int j=0;j<M1;j++) scanf("%lf",&A[i][j]);

  int N2, M2;
  printf("Введите размеры 2-ой матрицы: "); scanf("%d %d",&N2,&M2);
  double B[N2][M2];
  printf("Введите элементы 1-ой матрицы :\n");
  for(int i=0;i<N2;i++) for(int j=0;j<M2;j++) scanf("%lf",&B[i][j]);

  if(M1 != N2){
    printf("Матрицы перемножить нельзя!\n");
    return 0;
  }
```

Пример 2

```
  double C[N1][M2];
  for(int i=0;i<N1;i++) for(int j=0;j<M2;j++){
    C[i][j] = 0.0;
    for(int k=0;k<M1;k++) C[i][j] += A[i][k]*B[k][j];
  }
  printf("Результат:\n");
  for(int i=0;i<N1;i++){
    for(int j=0;j<M2;j++) printf("%6.2lf ",C[i][j]);
    printf("\n");
  }
  return 0;
}
```

Пример 3

Дана целочисленная матрица размера N×M. Размер матрицы и элементы вводятся пользователем. Поменять местами строки матрицы, содержащие максимальный и минимальный элементы матрицы. Если максимум и минимум находятся в одной строке, то строку обнулить. Полученную матрицу вывести на экран.

Пример 3

```
#include <stdio.h>
int main(int argc, char *argv[])
{
  int N,M;
  printf("Введите размеры матрицы: ");  scanf("%d %d",&N,&M);
  if((N<2)||(M<2)) {
    printf("Размеры введены некорректно!\n");
    return 0;
  }
  printf("Введите матрицу:\n");
  int matrix[N][M];
  for(int i=0;i<N;i++) for(int j=0;j<M;j++) scanf("%d",&matrix[i][j]);
  int max = matrix[0][0], min = matrix[0][0];
  int imax = 0, imin = 0;
    for(int i=0;i<N;i++) for(int j=0;j<M;j++){
      if(max < matrix[i][j]) {max = matrix[i][j]; imax = i;}
      if(min > matrix[i][j]) {min = matrix[i][j]; imin = i;}
  }
```

Пример 3

```
if(imin != imax){
   for(int j=0;j<M;j++){
     int tmp = matrix[imin][j];
     matrix[imin][j] = matrix[imax][j];
     matrix[imax][j] = tmp;
   }
 }else{
    for(int j=0;j<M;j++) matrix[imin][j] = 0;
 }
 printf("Результат:\n");
 for(int i=0;i<N;i++){
   for(int j=0;j<M;j++) printf("%4d ",matrix[i][j]);
   printf("\n");
 }
 return 0;
}
```

Пример 4

Дана вещественная матрица размера N на M. Продублировать столбец матрицы с минимальной суммой элементов. Значения N, M и элементы матрицы вводит пользователь.

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

Пример 4

```
int main(int argc, char *argv[])
{
  int N,M;
  printf("Введите N & M: "); scanf("%d %d",&N,&M);
  double matrix[N][M+1];
  printf("Введите матрицу:\n");
  for(int i=0;i<N;i++)
    for(int j=0;j<M;j++)
      scanf("%lf",&matrix[i][j]);
  double min = 0.0;
  int pos = 0;
  for(int j=0;j<M;j++){
    double sum = matrix[0][j];
    for(int i=1;i<N;i++) sum += matrix[i][j];
    if((j==0)||(sum < min)) {min = sum; pos = j;}
  }
```

Пример 4

```
  for(int j=M;j>pos;j--)
    for(int i=0;i<N;i++)
      matrix[i][j] = matrix[i][j-1];
  M++;
  printf("Результат:\n");
  for(int i=0;i<N;i++){
    for(int j=0;j<M;j++)
      printf("%6.2lf ",matrix[i][j]);
    printf("\n");
  }
  return 0;
}
```

Пример 5

Дана целочисленная матрица размера N на M. Удалить из матрицы строку, сумма элементов которой минимальна. Размер матрицы и ее элементы вводит пользователь.

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

Пример 5

```
int main(int argc, char *argv[])
{
  int N, M;
  printf("Введите размеры матрицы: "); scanf("%d %d",&N,&M);
  int A[N][M];
  printf("Введите элементы матрицы:\n");
  for(int i=0;i<N;i++) for(int j=0;j<M;j++)
    scanf("%d",&A[i][j]);

  int min = 0, pos = -1;
  for(int i=0;i<N;i++){
    int sum = 0;
    for(int j=0;j<M;j++) sum += A[i][j];
    if((pos==-1)||(min>sum)) {min = sum; pos = i;}
  }
```

Пример 5

```
 for(int i=pos;i<N-1;i++)
   for(int j=0;j<M;j++)
    A[i][j] = A[i+1][j];
  N--;

  printf("Результат:\n");
  for(int i=0;i<N;i++){
    for(int j=0;j<M;j++) printf("%5d ",A[i][j]);
    printf("\n");
  }
  return 0;
}
```