ПЗ-09: **Работа с динамической памятью** (2 ч.)

***ДИНАМИЧЕСКИЕ МАССИВЫ***

Динамические массивы создаются с помощью функции **malloc( )** из библиотеки языка Си, при этом нужно указать тип и количество элементов массива.

**Пример**:

*Создается указатель* ***p*** *на вещественный тип, в ОП выделяется непрерывная область смежных ячеек памяти для размещения* ***100*** *элементов вещественного типа, и при этом адрес начальной ячейки записывается в указатель* ***р***

Функция **malloc(m)** из библиотеки языка Си выделяет непрерывную область памяти, длиной **m байтов**, поэтому для создания динамического массива из примера необходимо записать следующие операторы:

**int n = 100; m = sizeof (float);**

**float \*р = (float\*) malloc(n\*m);**

* В переменной **m** *вычислено и записано количество байт, необходимых для размещения одной переменной вещественного типа.*
* В аргументе функции **malloc( )** указано общее количество байт: **n\*m** для размещения **n=100** элементов вещественного типа.
* Доступ к элементам **динамического массива** осуществляется точно так же, как и к **статическим**, так как массив и указатель – одно и то же: **р[5].**
* Динамические массивы не обнуляются при создании (**!**)
* Память, выделенная для динамического массива, после использования должна быть освобождена. Для функции **malloc( )** посредством функции **free( )**:

**free ( р );** // освобождение памяти для указателя **р**

**Пример**: *Создать целочисленный динамический одномерный массив и подсчитать сумму его элементов*.

// #include<malloc.h>

*#include<stdlib.h>*

*#include<time.h>*

*void main( )*

*{ int \*p;*

*int sum = 0, i, n;*

*printf("Enter the length of the array:\n");*

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

***p = (int\*) malloc( n\*sizeof (int));*** /\* выделение памяти для n элементов целого типа \*/

*printf("Array fnd Sum:\n");*

*for (i = 0; i < n; i ++ )*

*{*

*p[i] = rand ()%10 - 5 ;*

*printf("p[%d] = %d\n", i, p[i]);*

*sum+=p[i];*

*}*

*printf("sum = %d\n", sum);*

***free(p);*** /\* освобождения выделенной памяти \*/

*printf("Click on Enter...");*

*getch();*

*}*

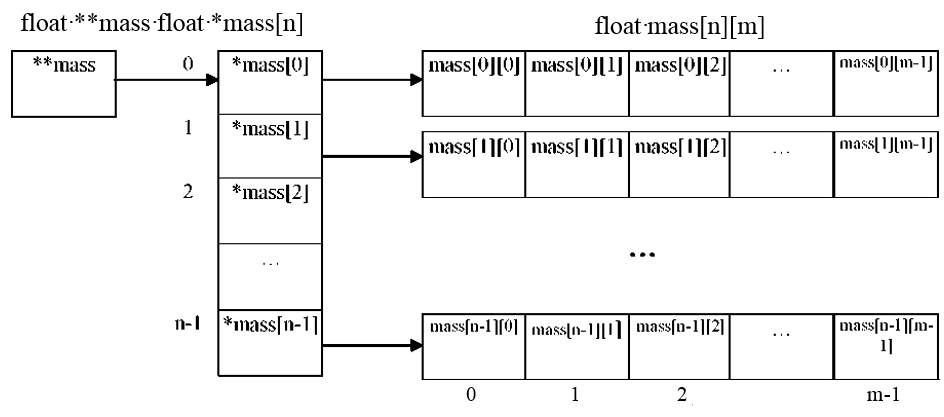
Для создания ***ДИНАМИЧЕСКОГО ДВУМЕРНОГО МАССИВА*** необходимо выполнить следующую последовательность действий:

1) **float \*\*mass;** //описать указатель на указатель;

2) **mass = (float\*\*) malloc( n\*sizeof (float\*));** /\* выделить память для одномерного массива указателей на будущие строки двумерного массива, состоящего из n элементов, где n - количество строк (см. рис. 1) \*/

3) **for (int i = 0; i < n; i ++)**

**{mass [i]= (float\*) malloc(m\*sizeof(float));}** /\* в цикле для каждого элемента массива указателей выделяется память под каждую строку двумерного массива. Причем каждая строка будет содержать m элементов вещественного типа, где m - количество столбцов (см. рис. 1) \*/



**Пример**: *Вычислить след матрицы, то есть произведение элементов главной диагонали матрицы с использованием двумерного динамического массива*.

|  |  |
| --- | --- |
| *#include<stdlib.h>*  *#include<time.h>*  *void main ( )*  *{*  *int \*\*matr;*  *int n, i, j, sled=1;*  *printf("Enter n:\n");*  *scanf("%d", &n);*  ***matr = (int\*) malloc(n\*sizeof(int\*));***  *for(int i = 0; i < n ; i ++)*  *matr[i] = (int\*) malloc (n\*sizeof(int));*  /*\* заполним матрицу случайными числами \**/  *srand(time(0));*  *for(i = 0 ; i < n ; i ++)*  *for (j = 0 ; j < n ; j ++)*  *{ matr[i][j] = rand()%10;*  *printf("matr[%d][%d] =* %*d*%*d\n", i, j, matr[i][j]);*  *}*  *for(i=0; i<n; i++)*  *sled \*= matr[i][i];*  *printf("sled =* %*d\n", sled);*    ***free(matr);***// *освобождения выделенной памяти*  *printf("Click on Enter...");*  *getch();*  *}* | *Заполнение массивов (на примере одномерного) случайными числами:*  //от 1 до 10  srand((unsigned)time(NULL));  for (int i = 0; i < n; i++ ) A[i] = rand()%10+1;  //от -20 до 20  srand((unsigned)time(NULL));  for (int i = 0; i < n; i++ ) A[i] = rand() % 41-20;  //на отрезке от a до b  srand(time(NULL)\*1000);  for (int i=0; i<n; i++)  {  A[i]=(rand()\*1.0/(RAND\_MAX)\*(b-a)+a);  } |

**Двумерные массивы. Обработка:**

|  |  |  |  |
| --- | --- | --- | --- |
| **//1. Заполнение (построчно, слева направо)**  for (int i = 0; i<n; i++)  for(int j = 0; j<m; j++)  a[i][j] = something;  **//2. Сумма всех элементов**  int sum=0;  for (int i = 0; i<n; i++)  for(int j = 0; j<m; j++)  sum += a[i][j]; | |  | |
| **//3. Сумма элементов строки**  int sum=0;  for(int j = 0; j<m; j++)  sum += a[3][j]; | | **//4. Сумма элементов столбца**  int sum=0;  for(int i = 0; i<n; i++)  sum += a[i][3]; | |
| **//5. Максимальный в каждом столбце**  int max[4];  for(int j = 0; i<m; j++)  {  max[j]=max[j,0];  for(int i=1;i<n; i++)  if (a[i][j] > max[j]) max[j]=a[i][j];  } | |  | |
| **//6. Сумма элементов главной диагонали**  for(int i = 0; i<n; i++) sum+=a[i][i]; | | **//7. Сумма элементов побочной диагонали**  for(int i = 0; i<n; i++) sum+=a[i][n-i-1]; | |
| **//8. Сумма элементов выше главной диагонали**  for(int i = 0; i<n; i++)  for(int j = 0; i<n; j++)  if (i < j) sum+=a[i][j]; | | **//9. Единицы ниже и на главной диагонали**  for (int i=0;i<n;i++)  for (int j=0; j<=i; j++) a[i][j]=1; |  |
| **//10. Единицы выше и на главной диагонали**  for (int i=0;i<n;i++)  for (int j=i; j<n; j++) a[i][j]=1; |  | **//11. Единицы выше и на побочной диагонали**  for (int i=0;i<n;i++)  for (int j=0; j < n-i; j++) a[i][j]=1; |  |
| **//12. Единицы ниже и на побочной диагонали**  for (int i=0;i<n;i++)  for (int j=(n-i-1); j < n; j++) a[i][j]=1; |  |  | |