ЛАБОРАТОРНАЯ РАБОТА № 19

ДИНАМИЧЕСКОЕ ВЫДЕЛЕНИЕ ПАМЯТИ

Цель работы: приобрести практические навыки по динамическому выделению памяти.

Краткие теоретические сведения

Указатели чаще всего используют при работе с динамической памятью, которую иногда называют «***куча***» (перевод английского слова *heap*). Это свободная память, в которой можно во время выполнения программы выделять место в соответствии с потребностями. Доступ к выделенным участкам динамической памяти производится только через указатели. Время жизни динамических объектов – от точки создания до конца программы или до явного освобождения памяти.

Динамическая переменная хранится в некоторой области ОП, не обозначенной именем, и обращение к ней производится через переменную-указатель.

Библиотечные функции

Функции для манипулирования динамической памятью в стандарте Си следующие:

void ****calloc***(*unsigned* ***n***, *unsigned* ***size***); – выделение памяти для размещения ***n*** объектов размером ***size*** байт и заполнение полученной области нулями; возвращает указатель на выделенную область памяти;

void ****malloc*** (*unsigned* ***n***) – выделение области памяти для размещения блока размером ***n*** байт; возвращает указатель на выделенную область памяти;

void ****realloc*** (*void* ****b***, *unsigned* ***n***) – изменение размера размещенного по адресу ***b*** блока на новое значение ***n*** и копирование (при необходимости) содержимого блока; возвращает указатель на перераспределенную область памяти; при возникновении ошибки, например, нехватке памяти, эти функции возвращают значение ***NULL***, что означает отсутствие адреса (нулевой адрес);

coreleft (*void*) – получение размера свободной памяти в байтах только для *MS DOS* (используется в *Borland* C++), тип результата: *unsigned* – для моделей памяти *tiny*, *small* и *medium*; *unsigned long* – для моделей памяти *compact*, *large* и *huge*;

void ***free*** (*void* ***b***) – освобождение блока памяти, адресуемого указателем ***b***.

Для использования этих функций требуется подключить к программе в зависимости от среды программирования заголовочный файл ***alloc.h*** или ***malloc.h***.

Создание одномерного динамического массива

В языке Си размерность массива при объявлении должна задаваться константным выражением.

Если до выполнения программы неизвестно, сколько понадобится элементов массива, нужно использовать динамические массивы, т.е. при необходимости работы с массивами переменной размерности вместо массива достаточно объявить указатель требуемого типа и присвоить ему адрес свободной области памяти (захватить память).

Память под такие массивы выделяется с помощью функций ***malloc*** и ***calloc*** во время выполнения программы. Адрес начала массива хранится в переменной-указателе. Например:

```
int n = 10;
double *b = (double *) malloc(n * sizeof (double));
```

В примере значение переменной *n* задано, но может быть получено и программным путем.

Обнуления памяти при ее выделении не происходит. Инициализировать динамический массив при декларации нельзя.

Обращение к элементу динамического массива осуществляется так же, как и к элементу обычного – например *a*[3]. Можно обратиться к элементу массива и через косвенную адресацию – *(*a* + 3). В любом случае происходят те же действия, которые выполняются при обращении к элементу массива, декларированного обычным образом.

После работы захваченную под динамический массив память необходимо освободить, для нашего примера *free*(*b*);

Таким образом, время жизни динамического массива, как и любой динамической переменной – с момента выделения памяти до момента ее освобождения.

Пример работы с динамическим массивом:

```
#include <alloc.h>
#include<stdio.h>
void main()
{  double *x;   int n;
printf("\nВведите размер массива – ");
scanf("%d", &n);
// Захват памяти
if ((x = (double*)calloc(n, sizeof(*x)))==NULL)
     {     puts("Ошибка ");
           return;      }
     ...
// Работа с элементами массива
     ...
free(x);     }    // Освобождение памяти
```

Создание двумерного динамического массива

Напомним, что имя двумерного массива – указатель на указатель. В данном случае сначала выделяется память на указатели, расположенные последовательно друг за другом, а затем каждому из них выделяется соответствующий участок памяти под элементы.

```
. . .
int  **m, n1, n2, i, j;
puts(" Введите размеры массива (строк, столбцов): ");
scanf("%d%d", &n1, &n2);
// Захват памяти для указателей – A (n1=3)
m = (int**)calloc(n1, sizeof(int*));
for (i=0; i<n1; i++)
    // Захват памяти для элементов – B (n2=4)
   *(m+i) = (int*)calloc(n2, sizeof(int));
for ( i=0; i<n1; i++)
   for ( j=0; j<n2; j++)
         m[i] [j] = i+j;              //  *(*(m+i)+j) = i+j;
```

```
. . .
for(i=0; i<n; i++)
   free(m[i]);                    // Освобождение памяти
free(m);
. . .
```

Порядок выполнения работы

1. Изучить теоретические сведения.
2. Ответить на контрольные вопросы.
3. Выполнить задание.

Контрольные вопросы

1. Что понимают под динамическим выделением памяти и в каких случаях оно используется ?
2. В какой области памяти ЭВМ осуществляется ее динамическое выделение?
3. Каково назначение функций ***calloc(), malloc()*** и ***free()***?
4. Опишите порядок выделения динамической памяти под переменную.
5. Что такое динамический массив?
6. Опишите порядок создания динамического массива.

Задания для выполнения

Выполнить задания из лабораторной работы № 17, используя динамическое выделение памяти для массивов.