Динамические данные

Виды памяти

Существует три вида памяти: статическая, стековая и динамическая.

Статическая память выделяется еще до начала работы программы, на стадии компиляции и сборки.

Статическая память

Существуют два типа статических переменных:

Глобальные переменные (определенные вне функций):

```
…
int max_num = 100;
…
void main() {
…
}
```

Статические переменные (определенные со словом static):

```
void main() {
  …
static int max = 100;
…
}
```

Локальные переменные

Локальные (или стековые) переменные – это переменные определенные внутри функции (или блока):

```
void my_func() {
    …
    int n = 0;
    …
    if (n != 0) {
        int a[] = {0, 2, 4, 5};
        …
    }
    …
}
```

Память выделяется в момент входа в функцию или блок и освобождается в момент выхода из функции или блока.

Динамическая память

Недостаток статической или локальной памяти: количество выделяемой памяти вычисляется на этапе компиляции и сборки.

Использование динамической памяти позволяет избавиться от данного ограничения.

Выделение и освобождение памяти

Необходимая библиотека:

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

Выделение памяти:

```
void* malloc(size_t n);
void* calloc(size_t num, size_t size);
void* realloc(void *ptr, size_t size);
```

C++:

```
<тип> <имя> = new <тип>;
```

Освобождение памяти:

```
void free(void *p);
```

C++:

```
delete <имя>;
```

Динамические массивы

Пример. Ввести с клавиатуры n чисел (n задается пользователем) и вывести их в обратном порядке.

Неправильный способ решения задачи (с использованием локальной переменной массива) :

```
void main() {
  int n,i;
  scanf(“%d”, &n); /* вводим кол-во чисел */
  int a[n];
  /* ошибка. Нельзя создавать массив используя переменную-размер */
  for (i = 0; i < n; i++)
    scanf(“%d”, &a[i]);
  for (i = n-1; i >=0; i--)
    printf(“%5d”, a[i]);
}
```

Динамические массивы

Правильный способ решения задачи (с использованием динамической переменной массива):

```
void main() {
  int n,i;
  scanf(“%d”, &n); /* вводим кол-во чисел */

   /* выделяем память под массив */
  int *a = (int*)malloc(n * sizeof(int));

  for (i = 0; i < n; i++)
    scanf(“%d”, &a[i]);
  for (i = n-1; n >=0; i--)
    printf(“%5d”, a[i]);

  free(a); /* освобождаем память */
}
```

Динамические структуры

Выделение памяти под структуру:

```
struct <имя> *<имя переменной> =
    (struct <имя>*) malloc(sizeof(<имя>));
```

Освобождение памяти:

```
free(<имя переменной>);
```

Опишем структуру:

```
struct student {
    char name[50];
    int grade;
    int group;
};
```

Массивы динамически создаваемых структур

Пример. Формирование массива из динамически создаваемых структур.

```
 void main() {
     /* Объявляем массив студентов */
   struct student* students[100] = {NULL};
   int i,n;
   scanf("%d", &n); /* n - количество студентов */

   for (i = 0; i < n; i++) {
       /* резервируем память */
       students[i] = (struct student*)malloc(
                   sizeof(students));
       scanf("%50s %d %d", &students[i]->name,
              &students[i]->age, &students[i]->grade);
   }
   …
}
```

Динамические массивы структур

Пример. Формирование динамического массива из структур.

```
 void main() {
     /* Объявляем массив студентов */
   struct student* a;
   int i,n;
   scanf("%d", &n); /* n - количество студентов */
     /* резервируем память */
   a = (struct student* a) malloc( n * sizeof(struct
  student));

   for (i = 0; i < n; i++) {
     scanf("%50s %d %d", &a[i].name,
              &a[i].age, &a [i].grade);
   }
   …
}
```

Динамическая память, функции и двумерные массивы

```
#include <stdio.h>
#include <stdlib.h>
#define N 4
#define L 5

int **AllocateM(int Width, int Height)
{
    int **m=(int**)malloc(Height*sizeof(int*));//ðàçìåðíîñòü óêàçàòåëÿ
    int i;
    for(i=0;i<Height;i++)
    {
        m[i]=(int*)malloc(Width*sizeof(int));
    }
    return m;
}
```

```
void Func(int E[N][L],int p,int m)
{    int i,j,t,r,c=E[0][0];
 for(i=0;i<p;i++)
{        for(j=1;j<m;j++)
 {            if(E[i][j]<c)
       c=E[i][j],t=i+1,r=j+1;
}
}
printf("The smallest one:%d\nIts position: ROW - %d, COLLUMN - %d",c,t,r);}

int main()
{
    int m[N][L];    int i,j;    AllocateM(N,L);    puts("Enter massive:\n");
for(i=0;i<N;i++)    {        for(j=0;j<L;j++)        scanf("%d",&m[i][j]);    }
puts("Entered massive:\n");    for(i=0;i<N;i++)    {        for(j=0;j<L;j++)
printf(" %d",m[i][j]);            printf("\n");    }    Func(m,N,L);
    return 0;
}
```

Передачи двумерного СТАТИЧЕСКОГО массива в функцию: вариант1

```
#include <iostream>
using std::cout;
using std::cin;

void funArray(int mass[][6], const int nstr, const int nstb);

int main()
{
	int Arr[5][6]={
				{2,13,2,5,0,3},
				{12,3,5,0,7,5},
				{1,2,3,4,5,-4},
				{121,11,2,3,4,8},
				{3,5,3,7,9,12}
			};
	funArray(Arr,5,6);

	cin.get();
	cin.get();
	return 0;
}
```

```
void funArray(int mass[][6], const int nstr, const int nstb)
{
	for(int i=0; i<nstr; i++)
	{
		cout<<"\n";
		for(int j=0; j<nstb; j++)
			cout<<mass[i][j]<<"  ";
	}
}
```

Для динамических массивов используют
Вариант2:

```
#include <iostream>
using std::cout;
using std::cin;

void funArray(int **mass, const int nstr, const int nstb);

int main()
{     //ввод размера динамического двумерного массива:
      int N, M;
      cin >>N>>M;
      //выделение места в памяти под динамический двумерный массив:
      int **Arr=new int* [N];
      for(int i=0; i<N; i++)
            Arr[i]=new int [M];
```

```
//заполнение массива данными:
        for(int i=0; i<N; i++)
                for(int j=0; j<M; j++)
                        cin>>Arr[i][j];
        //передача массива в функцию:
        funArray(Arr,N,M);

        cin.get();
        cin.get();
        return 0;
}
```

```
void funArray(int **mass, const int nstr, const int nstb)
{
	for(int i=0; i<nstr; i++)
	{
		cout<<"\n";
		for(int j=0; j<nstb; j++)
			cout<<mass[i][j]<<"  ";
	}
}
```