Динамическое распределение памяти

Динамическое распределение памяти

Память, которую использует программа делится на три вида:

- Статическая память (static memory)
  - хранит глобальные переменные и константы;
  - размер определяется при компиляции.
- Стек (stack)
  - хранит локальные переменные, аргументы функций и промежуточные значения вычислений;
  - размер определяется при запуске программы (обычно выделяется 4 Мб).
- Куча (heap)
  - динамически распределяемая память;
  - ОС выделяет память по частям (по мере необходимости).

Динамически распределяемую память используют тогда, если заранее неизвестно, сколько памяти потребуется (например, размер массива, который вводится пользователем), или при работе с большими объемами данных.

Функции динамического распределения памяти в Си

`void *malloc(size_t size);`

Функция выделяет `size` байтов памяти и возвращает указатель на неё. Если память выделить не удалось, то функция возвращает `NULL`. Так как `malloc` возвращает указатель типа `void`, то его необходимо явно приводить к нужному нам типу.

```
int *p = NULL;
p = (int*) malloc(100); //выделено 100 байт
p = (int *) malloc(5*sizeof(int)); //выделена память под 5 intов
free(p);
```

После окончания работы с выделенной динамически памятью нужно освободить ее. Для этой цели используется функция **`free`**, которая возвращает память под управление ОС.

Функции динамического распределения памяти в Си

`void *calloc(size_t num, size_t size);`

Функция выделяет `num` объектов размером `size` и заполняет их нулями. Обычно она используется для выделения памяти под массивы. Если память не может быть выделена, возвращается `NULL`.

```
int *p = (int *) calloc(10, sizeof(int));
```

`p` будет указывать на начало массива из 10 `int`ов, инициализированных нулями.

Функции динамического распределения памяти в Си

```
void* realloc(void* ptr, size_t size);
```

Функция позволяет изменить размер ранее выделенной памяти и получает в качестве аргументов старый указатель и новый размер памяти в байтах. Содержание блока памяти сохраняется даже если новый блок имеет меньший размер, чем старый. Отбрасываются только те данные, которые не вместились в новый блок.

Если размер указанный в параметре `size` больше, чем тот, который был выделен под указатель `ptr`, то проверяется, есть ли возможность выделить недостающие ячейки памяти подряд с уже выделенными. Если новое значение `size` больше старого, то содержимое вновь выделенной памяти будет неопределенным. Если места недостаточно, то выделяется новый участок памяти размером `size` и данные по указателю `ptr` копируются в начало нового участка.

Динамическое распределение памяти в C++

В C++ для выделения и освобождения памяти используются операторы new и delete.

```
int *p = new int[10]; // выделение памяти под 10 intов
delete [] p;
```

Если требуется выделить память под один элемент, то можно использовать

```
int *p = new int;
```

или

```
int *p = new int(10); // выделенный int проинциализируется значением 10
delete p;
```