## **Виды памяти**

Существует 3 типа памяти: статический, автоматический и динамический.

**Статический** — выделение памяти до начала исполнения программы. Такая память доступна на протяжении всего времени выполнения программы. Во многих языках для размещения объекта в статической памяти достаточно задекларировать его в глобальной области видимости.

int id = 150; // определение статической глобальной переменной

int main()

{

std::cout << id + 8; // её использование

}

**Автоматический**, также известный как «размещение на стеке», — самый основной, автоматически выделяет аргументы и локальные переменные функции, а также прочую метаинформацию при вызове функции и освобождает память при выходе из неё.

Стек, как структура данных, работает по принципу LIFO («последним пришёл — первым ушёл»). Другими словами, добавлять и удалять значения в стеке можно только с одной и той же стороны.

Автоматическая память работает именно на основе стека, чтобы вызванная из любой части программы функция не затёрла уже используемую автоматическую память, а добавила свои данные в конец стека, увеличивая его размер. При завершении этой функции её данные будут удалены с конца стека, уменьшая его размер. Длина стека останется той же, что и до вызова функции, а у вызывающей функции указатель на конец стека будет указывать на тот же адрес.

Проще всего это понять из примера на С++:

int f (int n)

{

Int x;

}

Размер автоматической памяти, а он тоже фиксированный, определяется линковщиком (обычно — 1 мегабайт), максимальный размер зависит от конкретной системы и настроек компилятора/линковщика.

В Си работать с **динамической памятью** можно при помощи соответствующих функций распределения памяти (calloc, malloc, free), для чего необходимо подключить библиотеку  
malloc.h  
  
С++ использует новые методы работы с динамической памятью при помощи операторов new и delete:

new — для выделения памяти;

delete — для освобождения памяти.

Оператор new используется в следующих формах:

new тип; — для переменных

new тип[размер]; — для массивов

Память может быть распределена для одного объекта или для массива любого типа, в том числе типа, определенного пользователем. Результатом выполнения операции new будет указатель на отведенную память, или нулевой указатель в случае ошибки.

int \*ptr\_i;  
double \*ptr\_d;  
struct person \*human;  
…  
ptr\_i = new int;  
ptr\_d = new double[10];  
human = new struct person;

Память, отведенная в результате выполнения new, будет считаться распределенной до тех пор, пока не будет выполнена операция delete.  
  
Освобождение памяти связано с тем, как выделялась память – для одного элемента или для нескольких. В соответствии с этим существует и две формы применения delete:

delete указатель; — для одного элемента

delete[] указатель; — для массивов

Например, для приведенного выше случая, освободить память необходимо следующим образом:

delete ptr\_i;  
delete[] ptr\_d;  
delete human;  
  
Освобождаться с помощью delete может только память, выделенная оператором new.

Максимальный размер динамической памяти зависит от многих факторов: среди них ОС, процессор, аппаратная архитектура в целом, не говоря уже о самом очевидном — максимальном размере ОЗУ у конкретного устройства. Например x86\_64 процессоры используют только 48 бит для адресации виртуальной памяти, что позволяет использовать до 256 ТБ памяти.