**Указатели и динамическая память**

Переменная – именованный участок оперативной памяти.

Байт – минимально адресуемый участок оперативной памяти.

Место размещения переменной в ОП – её адрес.

Указатель – переменная или константа, предназначенная для хранения адреса значения определённого типа. Значением указателя является адрес объекта, записанный в виде целого беззнакового числа, которое интерпретируется особым образом – адрес сообщает только о том, где размещена переменная и ничего не говорит о ней самой.

Сегментированный адрес состоит из сегмента и смещения внутри него, т.е. принимает вид, условно, 0002:00А4. Адрес невозможно присвоить в числовую переменную, хоть это и целое беззнаковое число.

Определение и использование указателя.

Тип \*идентификатор [ = инициализатор ];

Примеры определения указателя:

* double \*pf (неинициализированная переменная-указатель типа double)
* int \*p1, \*p2 (определение двух указателей на целое)
* long \*p, num = 10 (совмещение определения указателя на целое и переменной

Для получения доступа к значению в памяти необходимо знать не только адрес начального байта области памяти, в которой это значение находится, но и размер памяти, которое оно занимает, поэтому:

* указатель содержит адрес начального байта области памяти адресуемого значения
* тип определяет количество байт, занимаемое этим значением

Операция прямой адресации (получения адреса) - &имя\_переменной, результат – адрес операнда. Например:

* int a = 0x11223344;
* int \*pa;
* pa = &a;

Операция косвенной адресации (разыменования) - \*имя\_указателя, результат – значение, расположенное по адресу, содержащемуся в указателе. Например:

* int a = 0x11223344;
* int \*pa = &a;
* …
* int b;
* b = \*pa (переменная b содержит значение, расположенное по адресу, содержащемуся в указателе pa)

Инициализация указателей:

* Адресом переменной, уже размещённой в памяти:
  + int x = 5;
  + int \*p = &x;
* Нулевым указателем:
  + int \*p1 = 0;
  + int \*p2 = nullptr;
* Значением другого указателя, уже имеющего значение:
  + int x = 5;
  + int \*p1 = &x;
  + int \*p2 = p1;
* С помощью операции new динамического выделения ОП

Использование динамической памяти:

тип \*идентификатор = new тип[(инициализирующее\_значение)];

* int \*p;
* p = new int;
* \*p = 123;

ИЛИ

* int \*p = new int (123);

Синтаксис операции освобождения памяти: delete имя\_указателя;

* int \*p = new int (123);
* …
* delete p;

Проблема утечки памяти:

* double \*x = new double(1.234);
* double \*y = new double(5.678);
* …
* y = x; (потеря значения указателя у без освобождения памяти)

ИЛИ

* double \*x = new double(1.234);
* double \*y = new double(5.678);
* …
* delete y;
* y = x;

Основные правила использования new и delete:

* не используйте delete для освобождения памяти, которая не распределялась с помощью new
* не используйте delete для освобождения одного и того же блока памяти дважды подряд
* операцию delete можно применить к нулевому указателю

Использование модификатора const:

* Указатель на константный объект: int const \*p (or const int \*p) – значение нельзя изменить, но указателю можно присвоить другой адрес
* Константный указатель на объект: int \* const p – адрес в указателе нельзя изменить, но можно изменить указанное значение
* Константный указатель на константный объект: int const \* const p – как указатель, так и указанное значение нельзя изменить

Ссылки. В языке С++ ссылка определяется как другое имя уже существующего объекта, т.е. является его синонимом (псевдонимом). Обозначение ссылки: тип & имя\_ссылки = инициализатор

* Int x = 10;
* Int &rx = x;

Ограничения на использование ссылок:

* Инициализация ссылки обязательна
* После инициализации значение ссылки нельзя изменить (т.е. ссылка не может быть оторвана от объекта инициализации)
* Операции действуют на переменную, с которой она связана, а не на ссылку