УДК 281

Ярош Н.С.

ПРИМЕНЕНИЕ УКАЗАТЕЛЕЙ В С И С++

БНТУ, Минск
Научный руководитель: Дробыш А.А.

Память можно представить по-разному.

Объяснение для военных на примере взвода. Есть взвод солдат. Численность – 30 человек. Построены в одну шеренгу. Если отдать им команду рассчитаться, у каждого в этой шеренге будет свой уникальный номер. Обязательно у каждого будет и обязательно уникальный. Этот взвод – доступная нам память. Всего нам здесь выделено для работы 30 ячеек. Можно использовать меньше. Больше – нельзя. К каждой ячейке можно обратиться и быть уверенным, что обратился именно к ней. Любому солдату можно дать что-то в руки. Например, цветы. То есть поместить по адресу данные.

В памяти хранятся числа. Ни с чем кроме чисел компьютер работать не умеет. Если была помещена в память какая-то комплексная структура, она все равно будет представлена числами. Даже если работать с ней как со структурой. Примером комплексной структуры в терминах языков С и С++ может быть, например, экземпляр структуры или объект класса.

Наименьшей адресуемой величиной в памяти типового компьютера является байт. Это означает, что каждый байт имеет собственный адрес. Для того, чтобы обратиться к полубайту, придется обратиться сначала к байту, а затем выделить из него половину. Если минимальная адресуемая величина – байт, то всю доступную программе память можно представить в виде последовательности байтов.

Система в компьютере двоичная. В 1 байте 8 бит. Английское bit означает binary digit, то есть двоичный разряд. Получается, что байт может принимать числовые значения от 0 до 2 в 8 степени

без единицы. То есть от 0 до 255. Если представлять числа в шестнадцатеричной системе, то от 0x00 до 0xFF.

Указатель – это переменная. Такая же, как и любая другая. Со своими «можно» и со своими «нельзя». У нее есть свое значение и свой адрес в памяти. Значение переменной-указателя – адрес другой переменной. Адрес переменной-указателя свой и независимый. Переменная-указатель объявляется также, как и любые другие переменные, но после имени типа ставится звездочка.

```
int *pointerToInteger;
```

Здесь объявляется переменная pointerToInteger. Ее тип – указатель на переменную типа int.

Обычно функция возвращает одно значение. А как вернуть больше одного? Рассмотрим код функции, которая меняет местами две переменные.

```
int swap(double a, double b) {
  double temp = a;
  a = b;
  b = temp;}
```

Пусть есть переменные x и y с некоторыми значениями. Если выполнить функцию, передав в нее x и y, окажется, что никакого обмена не произошло. И это правильно.

При вызове этой функции в стеке будут сохранены значения x и y. Далее a и b получат значения x и y. Будет выполнена перестановка. Затем функция завершится, и значения x и y будут восстановлены из стека.

Чтобы заставить функцию работать так, как нужно, следует передавать в нее не значения переменных x и y, а их адреса. Но и саму функцию тогда нужно адаптировать для работы с адресами.

```
void swap(double* a, double* b) {
  double temp = *a;
  *a = *b;
  *b = temp;}
```

Не стоить забывать о том, что и вызов функции теперь должен выглядеть иначе.

swap(&x, &y);

Теперь в функцию передаются адреса. И работа ведется относительно переданных адресов.

Если функция должна вернуть несколько значений, необходимо передавать в нее адреса.

Если функция должна менять значение переменной, нужно передавать ей адрес этой переменной.

У тех, кто только начинает программировать на С, есть одна распространенная ошибка. При вводе с клавиатуры с помощью функции scanf() они передают значение переменной, а не ее адрес. А ведь scanf() должна менять значение переменной.

Еще один важный случай, когда указатели крайне полезны – это передача большого объема данных. Пусть нам нужно передать в функцию целое число типа int. Таким образом, мы передаем в функцию sizeof(int) байт. Обычно это 4 байта (размер будет зависеть от архитектуры компьютера и компилятора). 4 байта – не так много. 4 байта уйдут в стек. Потому что имеет место передача по значению.

Теперь нам нужно передать 10 таких переменных. Это уже 40 байт. Тоже невелика задача.

Когда нужно передать большой объем данных, его передают не копированием, а по адресу. Все массивы, даже из одного элемента, передаются по адресу.

Указатели – это мощный инструмент. Указатели эффективны и быстры, но не слишком безопасны. Потому как вся ответственность за их использования ложится на разработчика. Разработчик – человек. А человеку свойственно ошибаться.

Представим ситуацию.

int x;

int *p;

В большинстве компиляторов С и С++ неинициализированные локальные переменные имеют случайное значение. Глобальные обнуляются.

Если мы захотим разыменовать указатель и присвоить ему значение, скорее всего, будет ошибка.

*p = 10;

Неинициализированный указатель p хранит случайный адрес. Мы можем попытаться получить значение по этому адресу и что-то туда записать. Но совсем не факт, что нам можно что-то делать с памятью по этому адресу.

Указатели можно и нужно обнулять. Для этого есть специальное значение NULL.

int *p = NULL;

Это запись больше соответствует стилю C. В C++ обычно можно инициализировать указатель нулем.

int *p = 0;

Ловкость рук и никакого мошенничества. На самом деле, если изучить библиотечные файлы языка, можно найти определение для NULL.

#define NULL (void*)0

Для C NULL – это нуль, приведенный к указателю на void. Для C++ все немного не так. Стандарт говорит: «The macro NULL is an implementation-defined C++ null pointer constant in this International Standard. Possible definitions include 0 and 0L, but not (void*)0». То есть это просто 0 или 0, приведенный к long.

УДК 621.762.4

Ярош Н.С.

ПРОБЛЕМА РАЗВИТИЯ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ В РЕСПУБЛИКЕ БЕЛАРУСЬ

БНТУ, Минск

Научный руководитель: Липень С.Г.

На мой взгляд данная тема является очень интересной и ее актуальность не вызывает у меня сомнения. Далее я попытаюсь обосновать свое мнение.

В истории человечества можно выделить несколько этапов, которые человеческое общество последовательно проходило