Урок 6. Указатели

Цель урока: изучить определение, назначение и описание указателей, приобрести навыки в решении задач с помощью указателей.

Указатель — это адрес поля памяти, занимаемого программным объектом. Пусть в программе определены три переменные разных типов:

```
int a = 5;
char c = 'G';
float r = 1.2E8;
```

Эти величины разместились в памяти компьютера следующим образом:

Память	FFC0	FFC1	FFC2	FFC3	FFC4	FFC5	FFC6
Переменные	a		c	r			
Значения	5		'G'	$1.2*10^8$			

Операция & — адрес. Применение этой операции к имени переменной дает в результате ее адрес в памяти. Для переменных из данного выше примера:

```
&a равно FFC0,  &c -  FFC2,  &r -  FFC3.
```

Описание указателей. Для хранения адресов используются переменные типа «указатель». Формат описания таких переменных следующий:

```
тип  *имя_переменной
```

Примеры описания указателей:

```
int *pti;  char *ptc;  float *ptf;
```

После такого описания переменная pti может принимать значение указателя на величину целого типа; переменная ptc предназначена для хранения указателя на величину типа char; переменная ptf — на величину типа float.

Указателям могут присваиваться значения адресов объектов только того типа, с которым они описаны. В нашем примере допустимы операторы

```
pti = &a;  ptc = &c;  ptf = &r;
```

В результате указатели примут следующие значения:

```
pti - FFCO, ptc - FFC2, ptf - FFC3.
```

Как и для других типов данных, значения указателей могут инициализироваться при описании. Например:

```
int    a = 5;          int    *pti = &a;
char   c = 'G';        char   *ptc = &c;
float  r = 1.2E8;      float  *ptf = &r;
```

В заголовочном файле stdio.h определена константа — нулевой указатель с именем NULL. Ее значение можно присваивать указателю. Например:

```
ptf = NULL;
```

Не надо думать, что после этого указатель ptf будет ссылаться на нулевой байт памяти. Нулевой указатель обозначает отсутствие конкретного адреса ссылки.

Использованный в описаниях указателей символ *(звездочка) в данном контексте является знаком операции **разадресации**. С ее помощью можно сослаться через указатель на соответствующую переменную.

После приведенных выше описаний в записи выражений этой программы взаимозаменяемыми становятся a и *pti, c и *ptc, r и *ptf. Например, два оператора

```
x = a + 2;  И  x = *pti + 2;
```

тождественны друг другу. В результате выполнения оператора

```
cout << *pti << a;
```

на экран выведется 55.

Операции над указателями. Записывая выражения и операторы, изменяющие значения указателей, необходимо помнить главное правило: единицей изменения значения указателя является размер соответствующего ему типа.

Продемонстрируем это правило на определенных выше указателях. Выполнение операторов

```
pti = pti + l;  ИЛИ  pti++;
```

изменит значение указателя pti на 2, в результате чего он примет значение FFC2. В результате выполнения оператора pti-- ; значение указателя уменьшится на 2 и станет равным FFBE. Аналогично для указателей других типов:

```
ptc++;  увеличит значение указателя на 1;
ptf++;  увеличит значение указателя на 4.
```

Указатели и массивы. ***Имя массива трактуется как указатель-константа на массив***. Пусть, например, в программе объявлен массив:

```
int X[10];
```

В таком случае x является указателем на нулевой элемент массива в памяти компьютера. В связи с этим истинным является отношение

```
X == &X[0]
```

Отсюда следует, что для доступа к элементам массива кроме индексированных имен можно использовать разадресованные указатели по принципу:

```
имя[индекс]  тождественно  *(имя  +  индекс)
```

Например, для описанного выше массива x взаимозаменяемы следующие обозначения элементов:

```
X[5],  ИЛИ  *(X + 5), ИЛИ  *(5 + X).
```

Напоминаем, что для указателей работают свои правила сложения. Поскольку X — указатель на величину целого типа, то X + 5 увеличивает значение адреса на 10.

В языке C/C++ символ [играет роль знака операции сложения адреса массива с индексом элемента массива. Из сказанного должно быть понятно, почему индекс первого элемента массива всегда нуль. Его адрес должен совпадать с адресом массива:

```
X[0] == *(X + 0)
```

Поскольку имя массива является указателем-константой, то его нельзя изменять в программе, т. е. ему нельзя ничего присваивать. Например, если описаны два одинаковых по структуре массива

```
int  X[10], Y[10];
```

то оператор присваивания X = Y будет ошибочным. Пересылать значения одного массива в другой можно только поэлементно.

Теперь рассмотрим ***двумерные*** массивы. Пусть в программе присутствует описание:

```
int  P[5][10];
```

Это матрица из пяти строк и десяти чисел в каждой строке.

Двумерный массив расположен в памяти в последовательности по строкам. Попрежнему P является указателем-константой на массив, т. е. на элемент p[0][0] . Индексированное имя p[i] обозначает i-ю строку. Ему тождественно следующее обозначение в форме разадресованного указателя:

```
*(P + I * 10)
```

Обращение к элементу массива p[2][4] можно заменить на *(P + 2 * 10 + 4). В общем случае эквивалентны обозначения:

```
P[i][j]  И  *(P + i * 10 + j)
```

Здесь дважды работает операция «квадратная скобка». Последнее выражение можно записать иначе, без явного указания на длину строки матрицы P:

```
*(*(P + i) + j).
```

Очевидно, что по индукции для ссылки на элемент трехмерного массива A[i][j][k] справедливо выражение

```
*(*(*(A + i) + j) +k)  и  т.д.
```

Упражнения:

1. В оперативной памяти вектор int X[10] расположен, начиная с адреса B7F0. Какие значения примут выражения:
   ```
   а)  x + 1;
   б)  x + 5;
   в)  x - 4?
   ```
2. В программе объявлен массив: int P[] = {0 , 2 , 4 , 5 , 6 , 7 , 9 , 12};
   Какие значения примут выражения:
   ```
   а)  P[3];
   б)  *p;
   в)  *(P + 4);
   г)  *(P + P[2])?
   ```
3. Составить функцию сортировки значений трех переменных a, b, c в порядке возрастания.
4. Составить функцию заполнения целочисленного одномерного массива случайными значениями в диапазоне от 0 до JV.
5. Составить функцию вычисления среднего значения элементов вещественного одномерного массива. Использовать эту функцию в основной программе, определяющей в матрице номер строки с наибольшим средним значением.

Информационные источники:

1. **Семакин И.Г., Шестаков А.П.** Основы программирования: Учебник.- М.: Мастерство, 2002.- 432 с.