Язык программирования C++

Указатель

Массив и указатель

Арифметика для указателей

Одно из самых больших преимуществ языка C++ в том, что он позволяет писать высокоуровневые приложения, абстрагируясь от машинного уровня, и в то же время работать, по мере необходимости, близко к аппаратным средствам.

Понимание работы указателей и ссылок — один из этапов на пути к способности писать программы, эффективно использующие системные ресурсы.

Что такое указатель?

Указатель — это переменная, которая хранит адрес в памяти, по которому размещена другая переменная.

Указатель служит для того, чтобы косвенно указывать на содержимое некоторого участка памяти и для получения доступа к нему.

Для объявления указателя сначала определяется тип переменной, на которую он указывает, затем следует символ * и имя указателя.

char *char_pointer;

В переменной **char_pointer** хранится адрес в памяти, по которому располагается некоторая переменная типа **char**.

Указатель всегда имеет одинаковый размер, не важно, на данные какого типа он ссылается. В конце концов, он содержит всего лишь адрес памяти, который имеет одинаковый размер для любого типа. Он зависит только от архитектуры процессора компьютера.

Для 32-разрядной архитектуры указатель занимает 32 бита, то есть 4 байта.

Указатель, до инициализации содержит случайное значение. Неинициализированные указатели способны заставить вашу программу обратиться к недопустимой области памяти, приведя к аварийному отказу.

Указатель инициализируют значением **NULL**. Наличие значения NULL можно проверить, и оно не может быть адресом области памяти:

```
int  *kurs= NULL;
```

Адрес

Особая позиция в памяти называется ***адресом***.

Позиции памяти в компьютере пронумерованы, за адресом скрыто некоторое число. Чтобы выяснить адрес переменной, используется символ **&** (оператор ссылки).

```
char letter ;
char_pointer = &letter;
```

Косвенный доступ

После того, как указателю был присвоен адрес переменной, можно получить к ней доступ, если установить перед именем указателя символ * (косвенный оператор)

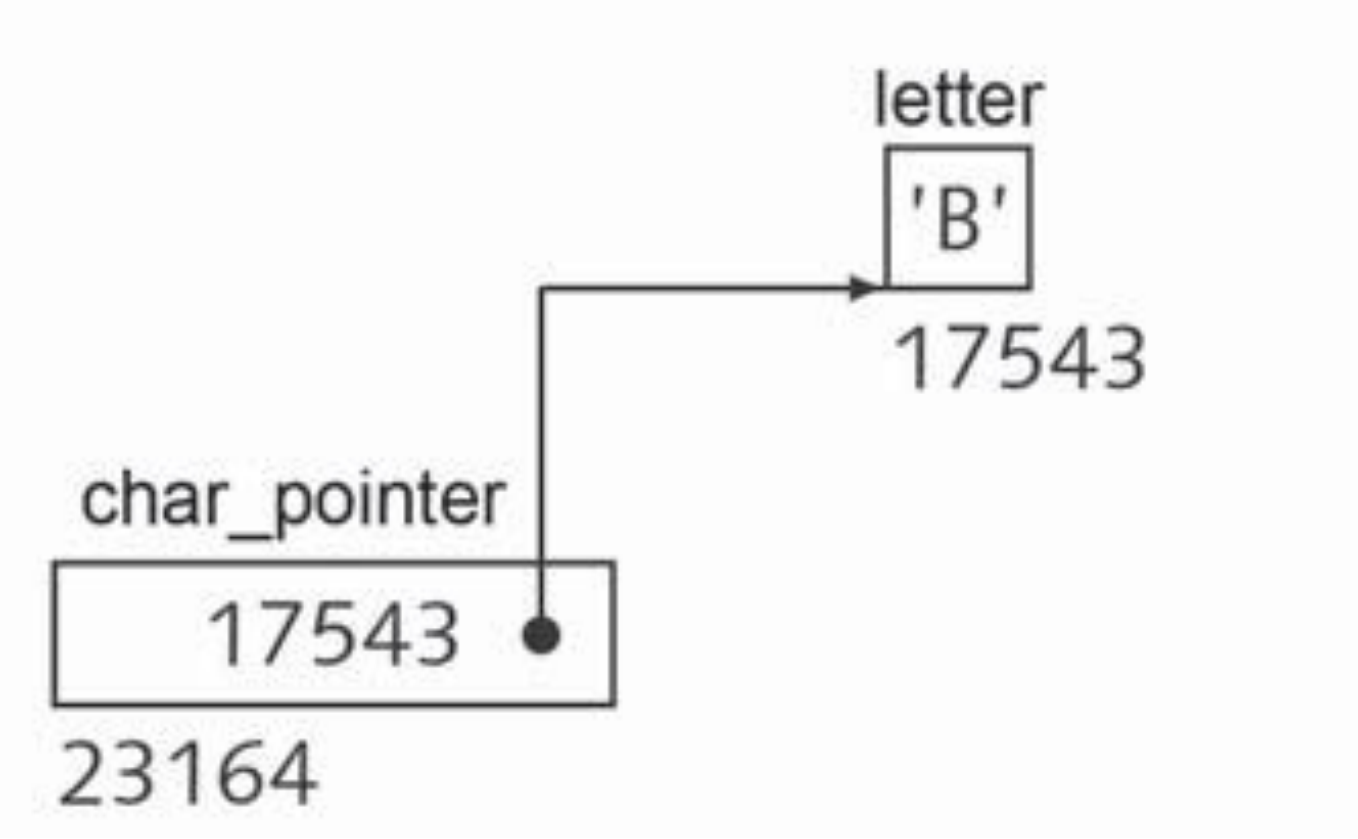
Что выводится на экран?

```
char letter='A' ;
char  *char_pointer;
char_pointer = &letter;
cout << *char_pointer;
```

С помощью символа * перед именем указателя четко обозначается, что обращение идет не к содержимому указателя, а к позиции в памяти, на которую ссылается указатель.

Переменной можно присвоить новое значение, используя указатель.

```
*char_pointer = 'B';
cout << letter;
```



Что выводится на экран?

```
int main()
{  int  *int_pointer = 0;
       int  intVar = 5;
   // указатель получает адрес переменной
       int_pointer =  &intVar;
  //меняем содержимое intVar
       *int_pointer = 1;
       intVar = *int_pointer + 1;
       cout >>intVar;
}
```

Указатели появились, прежде всего, для нужд системного программирования. Поскольку язык Си предназначался для "низкоуровневого" программирования, на нем нужно было обращаться, например, к регистрам устройств. У этих регистров вполне определенные адреса, т.е. необходимо было прочитать или записать значение по определенному адресу. Благодаря механизму указателей, такие операции не требуют никаких дополнительных средств языка.

Массивы и указатели

Указателем можно ссылаться на массив.

Для того чтобы поставить указатель на начало массива, надо написать:

```
int mas [4];
int  *mas_pointer = 0;

    mas_pointer = mas;
    //или
     mas_pointer= &mas[0];
```

Операция **&** перед одиноким именем массива недопустима!

mas_pointer= &mas;// нельзя

«Родство» указателей и массивов позволяет применять операцию * к имени массива:

value = *mas; //означает то же самое, что и
value = mas[0];

В процессе присваивания указателю массива квадратные скобки можно опустить. При этом переменная-указатель будет вести себя так, словно она изначально была массивом.

Пример использования указателя в качестве переменной массива:

Указатель можно использовать в качестве переменной массива:

```
int mas [4];
mas_pointer = mas;
mas_pointer[2]=10;
```

Арифметика для указателей

С указателями можно выполнять не только операции присваивания и обращения по адресу, но и ряд арифметических операций.

Указатель можно:

- инкрементировать
- декрементировать

При этом к нему будет добавляться столько байт, сколько требует тип переменной, на которую он указывает. Указатель на начало массива при инкрементировании будет указывать на следующий элемент.

Пример: присвоить элементам массива 0

```
int  values[MAX];
for(i=0;  i<MAX;  i++)
    values[i] = 0;
```

```
int  values[MAX];
int *pointer = values;
for( i=0;  i<MAX;  i++)
   {   *pointer = 0;   pointer++;     }
```

Второй вариант работает быстрее.

Сложение и вычитание

Увеличение указателя на единицу означает переход к следующей в памяти величине того же типа.

Прибавление или вычитание любого целого числа работает по тому же принципу, что и увеличение на единицу. Указатель сдвигается вперед (при прибавлении положительного числа) или назад (при вычитании положительного числа) на соответствующее количество объектов того типа, на который показывает указатель.

Следующее выражение указывает на два элемента позади того, на который ссылается указатель:

***(pointer+2) = 4;**

pointer[2] = 4;

Это однозначные команды (получен доступ ко второму элементу, расположенному после начального).

Сравнение указателей

Указатели одного и того же типа можно сравнивать.

```
int x = 10; int y = 10;
int* xptr = &x;    int* yptr = &y;
if (xptr == yptr)  // сравниваем указатели
   { cout << “равны"; }
   else { cout << "не равны"; }
if (*xptr == *yptr) // сравниваем значения
   { cout << “равны"; }
   else { cout << “не равны"; }
```