cq

Tutorial Link https://codequotient.com/tutorials/Introduction to Pointers/5a006a24e63d6b7fd5dec29b

TUTORIAL

Introduction to Pointers

Chapter

1. Introduction to Pointers

   ### Topics

   1.2   Declaring a pointer variable in C/C++

   1.9   Video Solution

Every variable in our program when stored in memory has assigned a particular memory location. Each memory location has an address assigned to it. It means every variable has been assigned a address in memory. Depending on size requirement of a variable the number of memory locations assigned to it i.e. if a variable needs 4 bytes and it has been assigned the address 1000 then from locations 1000 to 1003 will be assigned to it. Now we can access a variable with its name or its address also. Address of a variable is a hexadecimal number which can be assigned to another variable also. This concept of address assignment is used in C/C++ as pointer concept. This is useful in various ways in C/C++ programming, we will look at these cases shortly.

Let's see the whole concept with the help of the following diagram: -

Variable names	Memory Contents	Memory Addresses
		0xFFFFFFFF
		…
		0x00002003
		0x00002002
		0x00002001
p	0x00001000	0x00002000
		…
		0x00001009
		0x00001008
b	50	0x00001007
		0x00001006
		0x00001005
		0x00001004
		0x00001003
		0x00001002
a	45	0x00001001
		0x00001000
		…
		0x00000001
		0x00000000

In the figure above, 'a' and 'b' are two variables stored at locations 0x00001000 and 0x00001006 respectively. 'p' is another variable having the value equals to the address of variable 'a'. Now the value 45 can be accessed with the help of variable 'a' and with the help of address of 'a' which is stored in variable 'p'.

A pointer variable is therefore a special variable that contains an address, which is a location of another variable as shown in below figure. Value of pointer variable will be stored in another memory location.

pointer variable pointing to
variable at address 1000.

variable var1 whose
address is 1000.
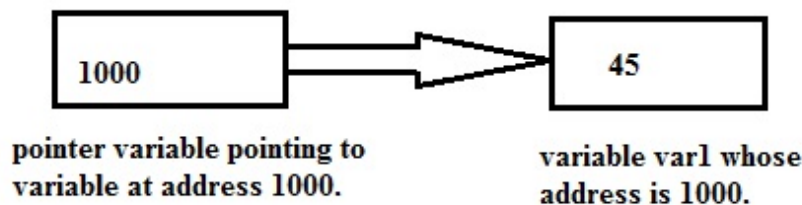
Declaring a pointer variable in C/C++

To use the pointer variables in C/C++, we will use two operators as
below: -

Operators	Name	Description
& (ampersand sign)	Address of operator	Return the address of a variable.
* (asterisk sign)	Indirection operator or De-referencing operator	Accesses the value at the address.

General syntax to declare a pointer variable is: -

```
type *pointer_name;
```

In this case, pointer_name is a pointer to a variable of data type 'type'.
For example: -

```
int *p;       // p is an pointer to an integer variable.
char *p1;     // p is an pointer to an character variable.
float *p2;     // p is an pointer to an float variable.
```

To use the pointers we have to follow the following sequence: -

- Declare a pointer variable.
- Assign the address of some variable to the pointer variable.
- Use the pointer variable to access the variable value.

To assign the address of a variable we will use the '&' (ampersand) operator called 'address of' operator. The following program will print the variable and its address: -

```c
#include <stdio.h>
int main ()
{
   int  var1=10;
   printf("Value of var1=%d , Address of var1= %p\n", var1, &var1);
   return 0;
}

```

```cpp
#include<iostream>
#include<cstdio>
#include<cmath>

using namespace std;

int main ()
{
   int  var1=10;
   cout<<"Value of var1="<< var1 <<", Address of var1= "<< &var1 <<endl;
   return 0;
}


```

The output of the program will be: -

```
Value of var1=10 , Address of var1=0xBFFC10B3
```

The address may vary from one system to another as memory address will depend on the underlying hardware. So to use the pointers we will use these operator as shown in below program:-

```c
#include <stdio.h>
int main()
{
  int *ptr;
  int i = 50;
  ptr = &i;          /* address of i is assigned to ptr */
  printf("value of i=%d address of i=%p value of p=%p\n", i, &i, ptr);
  printf("value of i=%d Value of variable pointed by ptr=%d\n", i, *ptr);
  return 0;
}
```

```cpp
#include<iostream>
#include<cstdio>

using namespace std;

int main()
{
  int *ptr;
  int i = 50;
  ptr = &i;          /* address of i is assigned to ptr */
  cout<<"value of i="<< i <<" address of i= "<< &i << " value of p=" << ptr << endl;
  cout<<"value of i="<< i <<" Value of variable pointed by ptr="<< *ptr << endl;
  return 0;
}
```

The output of the above program is

```
value of i=50 address of i=0xbfdfbd18 value of p=0xbfdfbd18
value of i=50 Value of variable pointed by ptr=50
```

A pointer variable can refer to multiple variables of valid type
throughout the program as shown below, but at any time it will point to
any one variable only: -

```c
#include <stdio.h>
int main()
{
  int *ptr;
  int i = 50, j = 10;
  ptr = &i;          /* address of i is assigned to ptr */
  printf("value of i=%d address of i=%p value of p=%p\n", i, &i, ptr);
  printf("value of i=%d Value of variable pointed by ptr=%d\n", i, *ptr);
  ptr = &j;          /* address of i is assigned to ptr */
  printf("value of j=%d address of j=%p value of p=%p\n", j, &j, ptr);
  printf("value of j=%d Value of variable pointed by ptr=%d\n", j, *ptr);
  return 0;
}
```

```cpp
#include<iostream>
#include<cstdio>

using namespace std;

int main()
{
  int *ptr;
  int i = 50, j = 10;
  ptr = &i;          /* address of i is assigned to ptr */
  cout<<"value of i="<< i <<" address of i="<< &i <<" value of p=" << ptr << endl;
```

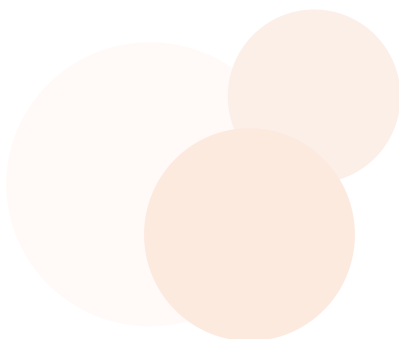```
12   cout<<"value of i="<< i <<" Value of variable
   pointed by ptr="<< *ptr << endl;
13   ptr = &j;   /* address of i is assigned to ptr
   */
14   cout<<"value of j="<< j <<" address of j="<< &j <<
   " value of p=" << ptr << endl;
15   cout<<"value of j="<< j <<" Value of variable
   pointed by ptr="<< *ptr << endl;
16   return 0;
17   }
18
```

The output of the above program will be

```
value of i=50 address of i=0xbfa0ecb4 value of p=0xbfa0ecb4
value of i=50 Value of variable pointed by ptr=50
value of j=10 address of j=0xbfa0ecb8 value of p=0xbfa0ecb8
value of j=10 Value of variable pointed by ptr=10
```

Video Solution

<iframe width="560" height="315" src="https://www.youtube.com/embed/k05KlNuDR4Q" title="YouTube video player" frameborder="0" allow="accelerometer; autoplay; clipboard-write; encrypted-media; gyroscope; picture-in-picture" allowfullscreen></iframe>