Pointers

Introduction to Pointers

- *Pointer variables are also known as pointers. You can use a pointer to reference the address of an array, an object, or any variable.*

- *A pointer variable holds the memory address. Through the pointer, you can use the dereference operator * to access the actual value at a specific memory location.*

- *Pointer variables*, simply called *pointers*, are declared to hold memory addresses as their values.

Overview of Pointers

- Normally, a variable contains a data value—e.g., an integer, a floating-point value, and a character.
- However, a pointer contains the memory address of a variable that in turn contains a data value.
- Each byte of memory has a unique address. A variable's address is the address of the first byte allocated to that variable. Suppose three variables count, and letter are declared as follows:
  - int count = 5;
  - char letter = 'A';

Declaring Pointers

- Like any other variables, pointers must be declared before they can be used. To declare a pointer, use the following syntax:

  dataType* pVarName;

- For example, the following statements declare pointers named pCount,and pLetter, which can point to an int variable, and a char variable respectively.
  - int* pCount;
  - char* pLetter;

Assigning Address

- You can now assign the address of a variable to a pointer. For example, the following code assigns the address of variable count to pCount:

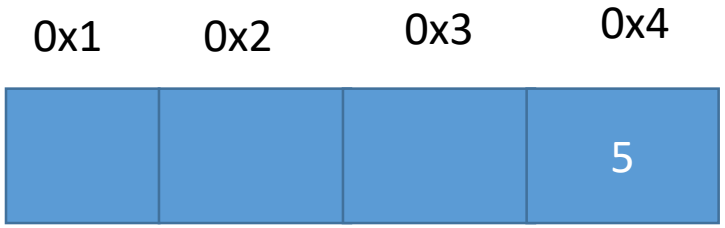
```
pCount = &count;
```

- The ampersand (&) symbol is called the *address operator* when placed in front of a variable.
- It is a unary operator that returns the variable's address. So, you may pronounce &count as the address of count.

Example 1

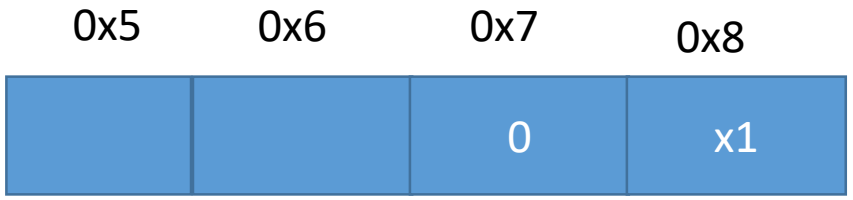
- int count = 5;

count



- int* pCount = &count
- After this statement pCount will store 0x1 in it.

pCount



Example 2

```
#include<iostream>
using namespace std;
main()
{
  int count = 5;
  int *pCount = &count;
  cout<<"address of count = "<<pCount<<endl;
  cout<<"access the value at the address available in pointer pCount = ";
  cout<<*pCount<<endl;
  cout<<"value hold by count variable = "<<count;

}
```

```
address of count = 0x6ffe04
access the value at the address available in pointer pCount = 5
value hold by count variable = 5
--------------------------------
Process exited after 0.241 seconds with return value 0
Press any key to continue . . .
```

Pointers and Arrays

- int arr[4] = {11,13,15,18};

0x32	0x36	0x40	0x44
11	13	15	18

int *ptr = arr

int *p1 = &arr[2];

ptr = 0x32

p1 = 0x40

Example 3

```
#include<iostream>
using namespace std;
main()
{

  int arr[3]={21,2,3};
  int *p2 = arr;
  for(int i =0;i<3;i++)
  cout<<"value at index "<<i<<" ="*p2++<<endl;


}
```

```
value at index 0 =21
value at index 1 =2
value at index 2 =3
```

Task 1

- Write a code to print an array in a reverse order using pointers.

Output:

7, 8, 5, 3

Solution: Task 1

```
#include<iostream>
using namespace std;
main()
{
    int arr[4] = {3,5,8,7};
    int *p = &arr[3];
    cout<<" element in reverse order = ";

       for(int i = 0;i<4;i++)
       cout<<*p--<<", ";
}
```

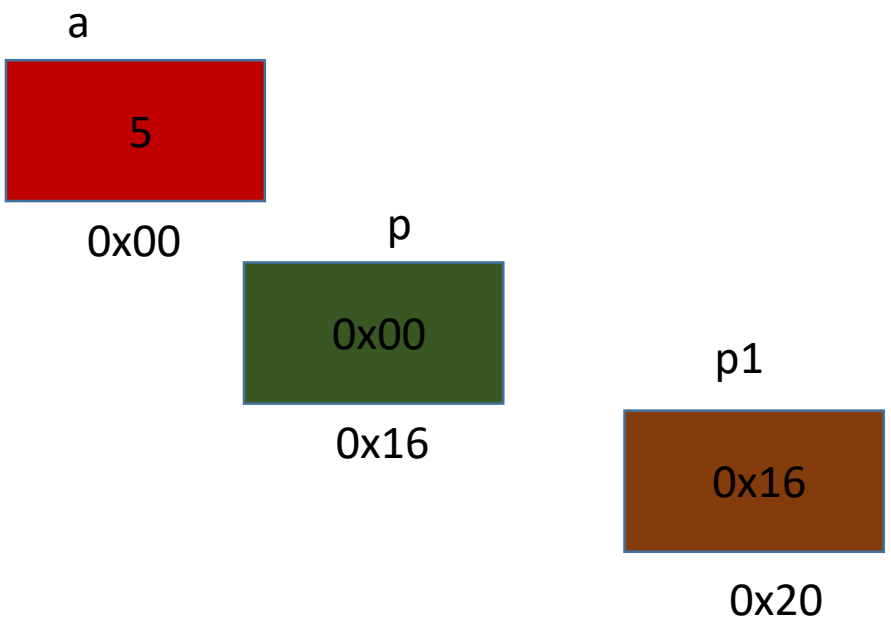
```
 element in reverse order = 7, 8, 5, 3,
--------------------------------
Process exited after 0.301 seconds with return value 0
Press any key to continue . . .
```

Double Pointers

```
int a = 5;
int *p = &a;
cout<<“address of a = “<<p<<endl;
int **p1 = &p;
cout<<“address of pointer p = ”<<&p<<endl;
//Same as
cout<<“address stored in pointer p1 = ”<<p1<<endl;
cout<<“ value pointed by pointer p = “<<*p<<endl;
cout<<“value pointed by pointer p1 = “<<*p1<<endl;
cout<<“value pointed by double pointer “<<**p1<<endl;
```

Double Pointers

Explanation:



Output:

address of a = 0x00

address of pointer p = 0x16

address stored in pointer p1 = 0x16

value pointed by pointer p = 5

=

value pointed by pointer p1 = 0x00

value pointed by double pointer = 5