C++ Pointers

Dr. Pravin Zode

Assistant Professor
Department of Electronics Engineering
Yeshwantrao Chavan College of Engineering

August 2, 2024

1. Pointer Introduction

2. Symbols used in Pointers

3. Advantages of Pointers

4. Program using Pointer

Introduction

1. A pointer is a variable whose value is the address of another variable
2. General form of a pointer variable declaration is : - type *var-name;

Examples

```
int *ip;        // pointer to an integer
double *dp;     // pointer to a double
float *fp;      // pointer to a float
char *ch        // pointer to character
```

Symbols used in Pointers

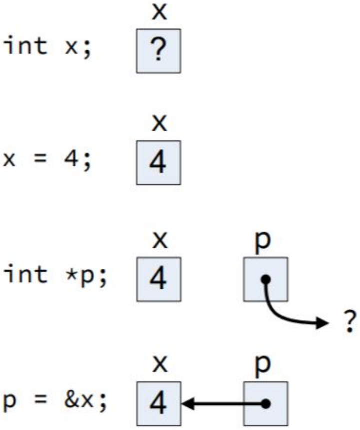
&(ampersand sign)

Address operator : It is used to determine the address of a variable.

*(asterisk sign)

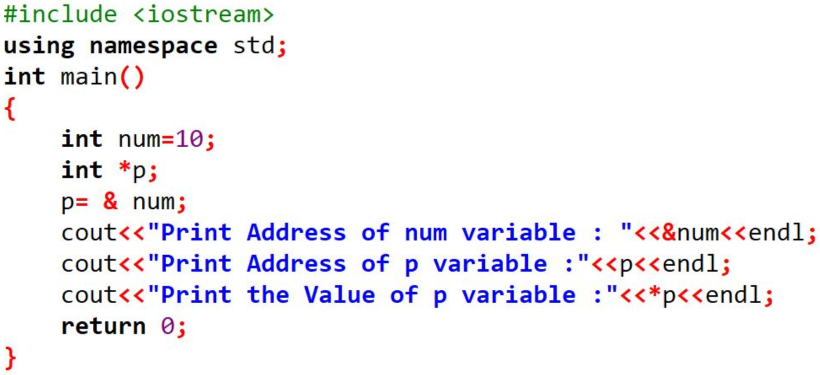
Indirection operator or Value at Address Operator : It is used to access the value of an address

Pointer Representation



```
int x;
x = 4;
int *p;
p = &x;
```

Pointer Program

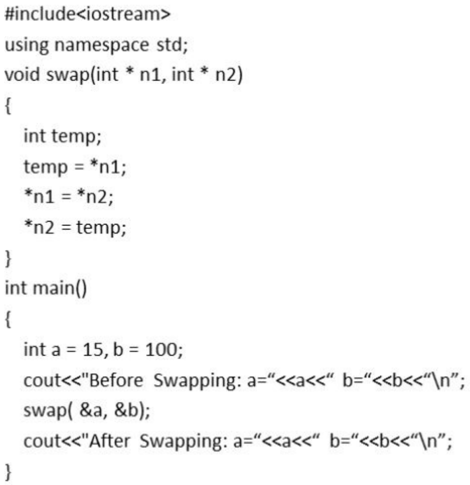


```
#include <iostream>
using namespace std;
int main()
{
    int num=10;
    int *p;
    p= & num;
    cout<<"Print Address of num variable : "<<&num<<endl;
    cout<<"Print Address of p variable :"<<p<<endl;
    cout<<"Print the Value of p variable :"<<*p<<endl;
    return 0;
}
```

Advantages of Pointers

1. It allows to use dynamic memory allocation
2. Helps to return more than one value from functdion
3. It provides direct access to memory
4. It reduces storage space of program
5. It improve exectuion speed of program
6. Help to build complex data structures sucha s linked list, trees, etc..

Swap Program Example



```
#include<iostream>
using namespace std;
void swap(int * n1, int * n2)
{
   int temp;
   temp = *n1;
   *n1 = *n2;
   *n2 = temp;
}
int main()
{
   int a = 15, b = 100;
   cout<<"Before  Swapping: a=“<<a<<“  b=“<<b<<“\n”;
   swap( &a, &b);
   cout<<"After  Swapping: a=“<<a<<“  b=“<<b<<“\n”;
}
```

Pointer and Arrays

Array Pointer

```
int *ptr;
int arr[5];
ptr = arr;
// stores the address of the first element of the array in variable ptr
```

Array Pointer

```
int *ptr;
int arr[5];
ptr = &arr[0];
//Notice that we have used arr instead of &arr[0] (both are same)
```

Pointer and Arrays

Element Pointer

int *ptr;
int arr[5];
ptr = arr;

ptr + 1 is equivalent to &arr[1];
ptr + 2 is equivalent to &arr[2];
ptr + 3 is equivalent to &arr[3];
ptr + 4 is equivalent to &arr[4];

Pointer and Arrays

Accessing Element using Pointer

// use dereference operator

*ptr == arr[0];

*(ptr + 1) is equivalent to arr[1];

*(ptr + 2) is equivalent to arr[2];

*(ptr + 3) is equivalent to arr[3];

*(ptr + 4) is equivalent to arr[4];

if we have initialized ptr = &arr[2]; then

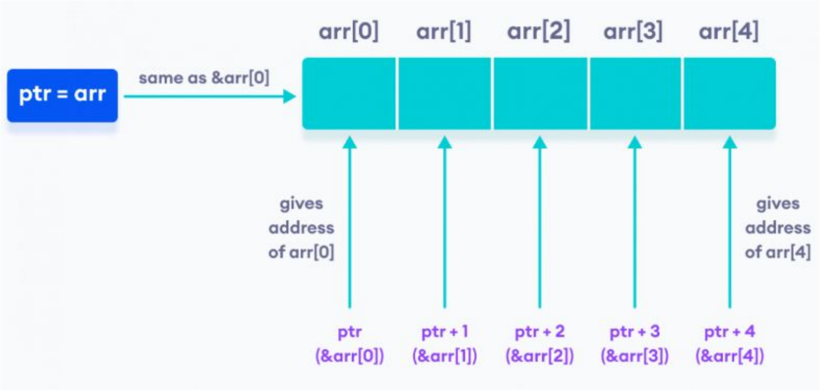
ptr - 2 is equivalent to &arr[0];

ptr - 1 is equivalent to &arr[1];

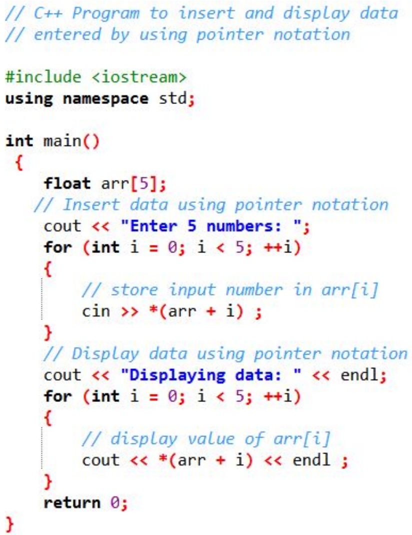
ptr + 1 is equivalent to &arr[3];

ptr + 2 is equivalent to &arr[4];

Pointers and Arrays



Example

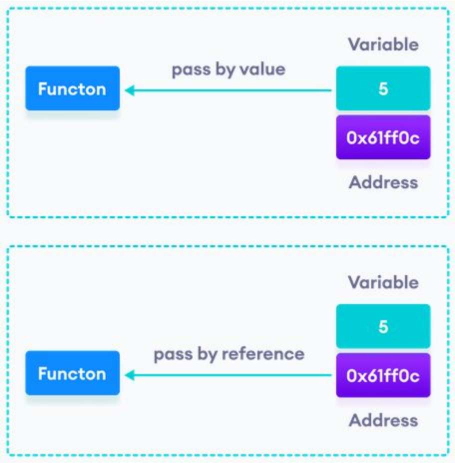


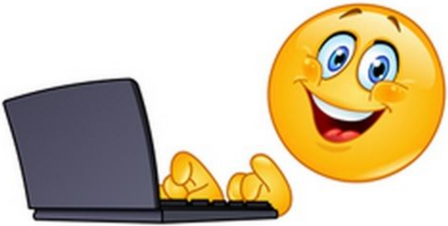
```
// C++ Program to insert and display data
// entered by using pointer notation

#include <iostream>
using namespace std;

int main()
 {
    float arr[5];
   // Insert data using pointer notation
    cout << "Enter 5 numbers: ";
    for (int i = 0; i < 5; ++i)
    {
        // store input number in arr[i]
        cin >> *(arr + i) ;
    }
    // Display data using pointer notation
    cout << "Displaying data: " << endl;
    for (int i = 0; i < 5; ++i)
    {
        // display value of arr[i]
        cout << *(arr + i) << endl ;
    }
    return 0;
}
```

Functions and Pointer





Thank you and Happy Coding

Problems with Normal Pointers

Some Issues with normal pointers in C++ are as follows:

- **Memory Leaks:** This occurs when memory is repeatedly allocated by a program but never freed. This leads to excessive memory consumption and eventually leads to a system crash.
- **Dangling Pointers:** It occurs at the time when the object is de-allocated from memory without modifying the value of the pointer
- **Wild Pointers:** Pointers that are declared and allocated memory but the pointer is never initialized to point to any valid object or address.
- **Data Inconsistency:** Data inconsistency occurs when some data is stored in memory but is not updated in a consistent manner.
- **Buffer Overflow:** When a pointer is used to write data to a memory address that is outside of the allocated memory block. This leads to the corruption of data which can be exploited by malicious attackers.