Pointers in C++

By – Mohammad Imran

Pointer

Pointer is a special type of variable that is used to store address of a variable. When we declare a variable it is preceded by **asterisk (*)**. While storing address of variable a variable is preceded by **ampersand (&)**.

A pointer can be incremented / decremented, i.e., to point to the next / previous memory location. The purpose of pointer is to save memory space and achieve faster execution time.

By – Mohammad Imran

Pointer

Key Points

1) Normal variable stores the value whereas pointer variable stores the address of the variable.
2) The content of the C++ pointer always be a whole number i.e. address.
3) Always C++ pointer is initialized to null, i.e. int *p = null.
4) The value of null pointer is 0.
5) & symbol is used to get the address of the variable.
6) * symbol is used to get the value of the variable that the pointer is pointing to.
7) If a pointer in C++ is assigned to NULL, it means it is pointing to nothing.

By – Mohammad Imran

8) Two pointers can be subtracted to know how many elements are available between these two pointers.
9) But, Pointer addition, multiplication, division are not allowed.
10) The size of any pointer is 8 byte (for 64 bit compiler).

NOTE: Using pointer we can find address of a variable as well as value of variable.

Pointer Declaration

```
Data_Type *Pointer_Name;
```

Example:

```
int *ptr;
```

Pointer Storing Address

```
int x;
ptr = &x;
```

```
int x = 10;
int *ptr = &x;
```

Pointer

Example - 1

```
#include<iostream>
using namespace std;
int main()
{
   int a = 20;
   cout << "Address of a = " << &a;
   cout << endl << "Value of a = " << a;
   return 0;
}
```

OUTPUT

```
Address of a = 648758
Value of a = 20
```

NOTE: Address without pointer

By – Mohammad Imran

Pointer

Example - 2

```
#include <iostream>
using namespace std;

int main()
{
   int *ptr, q;
   q = 50;
   ptr = &q;
   cout << "Value of q = " << q << endl;
   cout << "Address of q = " << ptr << endl;
   cout << "Value of q by pointer = " << *ptr;
   return 0;
}
```

OUTPUT

```
Value of q = 50
Address of q = 648758
Value of q by pointer = 50
```

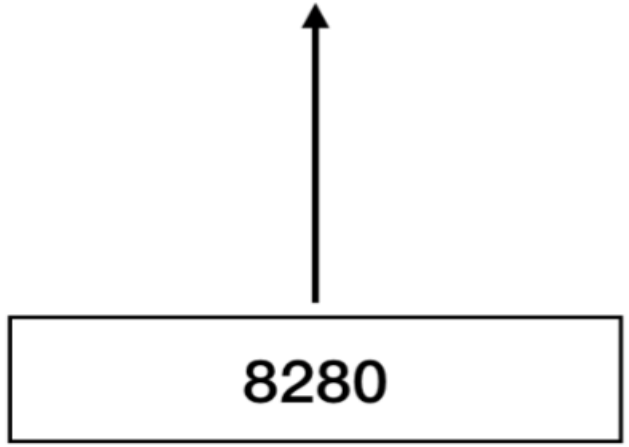
By – Mohammad Imran

Pointer

Graphical Representation

We can change the value of variable using address.

Code	Variable	Value	Address
`int num = 10;`	num	10	8280
`int *ptr = &num;`	ptr	8280	8272



By – Mohammad Imran

Pointer

Example - 3

```
#include <iostream>
using namespace std;
int main()
{
   int num = 10;
   int *ptr = &num;
   cout << "Value of num: "<< num << endl;
   cout << "Value of num: (using pointer) :" << *ptr << endl;
   *ptr = 20;
   cout << "value of num: " << num << endl;
   cout << "value of num (using pointer): " << *ptr;
   return 0;
}
```

OUTPUT

```
Value of num = 10
Value of num (using pointer = 10
Value of num = 20
Value of num (using pointer = 20
```

By – Mohammad Imran

Pointer

Types

There are eight different types of pointers which are as follows –

- ✓ Null pointer
- ✓ Void pointer
- ✓ Wild pointer
- ✓ Dangling pointer

By – Mohammad Imran

Pointer

Null Pointer

You create a null pointer by assigning the null value at the time of pointer declaration.

This method is useful when you do not assign any address to the pointer. A null pointer always contains value 0.

Null Pointer

Example - 1

```
#include <iostream>
using namespace std;
int main()
{
   int *ptr = NULL; //null pointer
   cout << "The value of ptr is: " << ptr;
   return 0;
}
```

OUTPUT

```
The value of ptr is: 0
```

By – Mohammad Imran

Pointer

Void Pointer

It is a pointer that has no associated data type with it. A void pointer can hold addresses of any type and can be typecast to any type.

It is also called a generic pointer and does not have any standard data type. It is created by using the keyword **void**.

Void Pointer

Example - 1

```
#include<stdio.h>
int main()
{
   int a = 2;
   char b = 'A';
   float f = 3.5f;
   void *ptr;
   ptr = &a;
   cout << "Typecasting a = "<< *(int *)ptr << endl;
   ptr = &b;
   cout<< "Typecasting b = "<< *(char *)ptr << endl;
   ptr = &f;
   cout << "Typecasting f = "<< *(float *)ptr;
   return 0;
}
```

OUTPUT

```
Typecasting a = 2
Typecasting b = A
Typecasting f = 3.5
```

By – Mohammad Imran

Pointer

Wild Pointer

If a pointer isn't initialized to anything, it's called a wild pointer. Wild pointers are also called uninitialized pointers.

By – Mohammad Imran

Wild Pointer

Example - 1

```
#include <iostream>
using namespace std;
int main()
{
   int *p; //wild pointer
   cout << *p;
   return 0;
}
```

OUTPUT

Segmentation fault
Or nothing will print

By – Mohammad Imran

Pointer

Dangling Pointer

A dangling pointer is a pointer that refers to a memory location that has been released or deleted.

By – Mohammad Imran

Dangling Pointer

Example - 1

```
#include <stdlib.h>
#include <iostream>
using namespace std;

int main()
{
   int *ptr;
   cout << "Size of *ptr is : " << sizeof(ptr) << endl;
   free(ptr);
   cout <<"Value of *ptr is "<< *ptr << endl;
   ptr = NULL;
   cout << "Value of *ptr is "<< *ptr << endl;
   return 0;
}
```

OUTPUT

```
Segmentation fault
```

By – Mohammad Imran

Call by value and call by reference

Functions can be invoked in two ways: **Call by Value** or **Call by Reference**. These two ways are generally differentiated by the type of values passed to them as parameters.

The parameters passed to function are called ***actual parameters*** whereas the parameters received by function are called ***formal parameters***.

Call By Value

In this parameter passing method, values of actual parameters are copied to function's formal parameters and the two types of parameters are stored in different memory locations. So any changes made inside functions are not reflected in actual parameters of the caller.

Call By Value

Example - 1

```
#include <iostream>
using namespace std;

void swap(int x, int y)

{
   int t;
   t = x;
   x = y;
   y = t;
   cout << "x = "<< x << endl;
   cout << "y = "<< y << endl;

}
```

OUTPUT

```
x = 20    y = 10
a = 10    b = 20
```

```
int main()
{
   int a = 10, b = 20;
   swap(a, b);
   cout << "a = "<< a << endl;
   cout << "b = "<< b << endl;
   return 0;
}
```

By – Mohammad Imran

Call By Reference

Both the actual and formal parameters refer to the same locations, so any changes made inside the function are actually reflected in actual parameters of the caller.

Call By Reference

Example - 1

```
#include <iostream>
using namespace std;
void swap(int*, int*);

int main()
{
   int a = 10, b = 20;
   cout << "a = " << a << endl;
   cout << "b = " << b << endl;
   swap(&a, &b);
   cout << "a = " << a << endl;
   cout << "b = " << b << endl;
   return 0;
}
```

OUTPUT

```
x = 20    y = 10
a = 10    b = 20
```

```
void swap(int *x, int *y)
{
   int t;
   t = *x;
   *x = *y;
   *y = t;
}
```

Formal Parameter

By – Mohammad Imran

Pointer Arithmetic

We can perform arithmetic operations on the pointers like addition, subtraction, etc. In pointer-from-pointer subtraction, the result will be an integer value. Following arithmetic operations are possible on the pointer in C++ language:

- ✓ Increment
- ✓ Decrement
- ✓ Addition
- ✓ Subtraction
- ✓ Comparison

By – Mohammad Imran

Pointer Incrementing

If we increment a pointer by 1, the pointer will start pointing to the immediate next location. This is somewhat different from the general arithmetic since the value of the pointer will get increased by the size of the data type to which the pointer is pointing.

The rule to increment the pointer is given below:

new_address = current_address + i * size_of(data type)

NOTE: Where i is the number by which the pointer get increased.

By – Mohammad Imran

Pointer Incrementing

Example - 1

```
#include<iostream>
using namespace std;
int main()
{
   int num = 50;
   int *p;
   p = &num;
   cout << "Address of num = " << p << endl;
   p = p + 1;
   cout << "After Increment : " << endl;
   cout << "Address of num = " << p <<endl;
   return 0;
}
```

OUTPUT

```
Address of p = 32100
After increment:
Address of p = 32104
```

By – Mohammad Imran

Pointer

Decrementing

Like increment, we can decrement a pointer variable. If we decrement a pointer, it will start pointing to the previous location. The formula of decrementing the pointer is given below:

```
new_address = current_address - i * size_of(data type)
```

NOTE: Where i is the number by which the pointer get decreased.

Pointer Decrementing

Example - 1

```
#include<iostream>
using namespace std;

int main()
{
   int num = 50;
   int *p;
   p = &num;
   cout << "Address of num = " << p << endl;
   p = p - 1;
   cout << "After Increment : " << endl;
   cout << "Address of num = " << p <<endl;
   return 0;
}
```

OUTPUT

```
Address of p = 32104
After decrement:
Address of p = 32100
```

By – Mohammad Imran

Pointer

Addition

Like increment and decrement to a pointer variable we can add value to the pointer variable.

```
new_address = current_address + (number * size_of(data type))
```

By – Mohammad Imran

Pointer Addition

Example - 1

```
#include<stdio.h>
int main()
{
   int number = 50;
   int *p;
   p = &number;
   cout << "Address of p = " << p;
   p = p + 3;
   cout << "After Addition:" << endl;
   cout << "Address of p = " << p;
   return 0;
}
```

OUTPUT

```
Address of p = 32104
After Addition:
Address of p = 32116
```

By – Mohammad Imran

Pointer

Illegal Arithmetic with Pointers

There are various operations which can not be performed on pointers. Since, pointer stores address hence we must ignore the operations which may lead to an illegal address.

- ✓ Address + Address = illegal
- ✓ Address * Address = illegal
- ✓ Address % Address = illegal
- ✓ Address / Address = illegal
- ✓ Address & Address = illegal
- ✓ Address ^ Address = illegal
- ✓ Address | Address = illegal
- ✓ ~Address = illegal

By – Mohammad Imran

Character Pointers in C++

By – Mohammad Imran

Character Pointer

In C++, different types of pointers can be printed, and the output varies depending on the type of pointer and how you print it.

int*, string*, char* Pointers:

- ✓ **int* and string*:** When you print pointers to these types, you're usually interested in seeing their address in memory. To achieve this, you should use the **%p** format specifier in C-style **printf** or use **std::cout** with proper **typecasting**.
- ✓ **char*:** When printing char* pointers, there's an overload for **operator <<** in C++'s **std::ostream** (like std::cout) that outputs the string

By – Mohammad Imran

Character Pointer

content rather than the memory address. This is because char* often points to a null-terminated string, and the stream operator is designed to print the string content.

Using **cout** we have different behaviors based on the type of pointer.

In the given example you can see address of character is not printing because **<<** operator is already overloaded to print the character.

Character Pointer

Example - 1

```
#include <iostream>
using namespace std;
int main()
{
   int num = 42;
   int *p = &num;
   cout << "Address integer p: " << p << endl;
   string str = "Hello";
   string *s = &str;
   cout << "Address of string s: " << s << endl;
   char ch = 'A';
   char *c = &ch;
   cout << "Address of character ch: " << c << endl;
   return 0;
}
```

OUTPUT

```
Address integer p: 0x6ffdf0
Address of string s: 0x6ffde0
Address of character ch: A?!.
```

By – Mohammad Imran

Character Pointer

Example - 1

```
#include <iostream>
using namespace std;

int main()
{
   int num = 42;
   int *p = &num;
   cout << "Address integer p: " << p << endl;
   string str = "Hello";
   string *s = &str;
   cout << "Address of string s: " << s << endl;
   char ch = 'A';
   char *c = &ch;
   cout << "Address of character ch: " << static_cast(void*)(&c) <<
   endl;
   return 0;
}
```

OUTPUT

```
Address integer p: 0x6ffdf0
Address of string s: 0x6ffde0
Address of character ch: 0x6ffde0
```

By – Mohammad Imran

Character Pointer

Some Unusual Behavior

In C++, **cout** shows different printing behavior with character pointers / arrays unlike pointers / arrays of other data types. So we will firstly explain how **cout** behaves differently with character pointers, and then the reason and the working mechanism behind it will be discussed.

Let's see different examples

By – Mohammad Imran

Character Pointer

Unusual Behavior - 1

```
#include <iostream>
using namespace std;
int main()
{
   int a[] = { 1, 2, 3 };
   char ch[] = "abc";
   cout << a << endl;
   cout << ch << endl;
   return 0;
}
```

OUTPUT

```
0x6ffe10
abc
```

By – Mohammad Imran

Unusual Behavior - 1

Explanation:

From the above code, it is clear that:

- ✓ When using the integer pointer to an array, **cout** prints the base address of that integer array.
- ✓ But when the character pointer is used, **cout** prints the complete array of characters (till it encounters a **null character**) instead of printing the base address of the character array.

Character Pointer

Unusual Behavior - 2

```
#include <iostream>
using namespace std;
int main()
{
   char b[] = "abc";
   char* c = &b[0];
   cout << c << endl;
   return 0;
}
```

OUTPUT

```
abc
```

By – Mohammad Imran

Unusual Behavior - 2

Explanation:

- ✓ In this example as well, the character type pointer **c** is storing the **base address** of the char array **b[]** and hence when used with **cout**, it starts printing each and every character from that base address till it encounters a **NULL** character.

Character Pointer

Unusual Behavior - 3

```
#include <iostream>
using namespace std;
int main()
{
   char c = '$';
   char* p = &c;
   cout << c << endl;
   cout << p << endl;
   return 0;
}
```

OUTPUT

```
$
$■o
```

By – Mohammad Imran

Unusual Behavior - 3

Explanation:

- ✓ In the above example, **c** is a simple character variable and it prints the value stored in it as expected. **p** being a character pointer when used with **cout**, results in the printing of each and every character till a null character is encountered. Thus, some garbage value is being printed after **'$'**.

Character Pointer

Unusual Behavior - 4

```
#include <iostream>
using namespace std;
int main()
{
   char c[] = ”Abc”
   cout << c << endl;
   cout << c[0] << endl;
   cout << *c << endl;
   return 0;
}
```

OUTPUT

```
Abc
A
A
```

By – Mohammad Imran

Unusual Behavior - 4

Explanation:

- ✓ Only ‘c’ with cout is treated as const char * and << operator overload for such input is called and thus every character is printed until null character.
- ✓ When using c[0], i.e., *(c + 0), it simply dereferences the particular memory location and prints the value stored at that location only.
- ✓ Similarly, for *c which is same as *(c + 0).

Character Pointer

Unusual Behavior - 5

```
#include <iostream>
using namespace std;
int main()
{
   char c[] = {'A','b','c','\0'};
   char *cptr = c;
   cout << cptr << endl;
   cout << (void*)cptr << endl;
   return 0;
}
```

OUTPUT

```
Abc
00fde1
```

NOTE: If you write char c[] = "Abc" it means null character is automatically placed at the end of the character string.

By – Mohammad Imran

Unusual Behavior - 4

Explanation:

- ✓ Only ‘c’ with cout is treated as const char * and << operator overload for such input is called and thus every character is printed until null character.
- ✓ When using c[0], i.e., *(c + 0), it simply dereferences the particular memory location and prints the value stored at that location only.
- ✓ Similarly, for *c which is same as *(c + 0).

Utilization of unusual behavior with character pointers:

```
void printPattern(char* ch)
{
   if (*ch == '\0')
      return;
   printPattern(ch + 1);
   cout << ch << endl;
}
int main()
{
   char ch[] = { "abcd" };
   printPattern(ch);
   return 0;
}
```

OUTPUT

```
d
cd
bcd
abcd
```

Pointer

Pointer to Pointers (Double Pointer)

As we know that, a pointer is used to store the address of a variable in C++. Pointer reduces the access time of a variable. However, In C++, we can also define a pointer to store the address of another pointer. Such pointer is known as a double pointer (pointer to pointer). The first pointer is used to store the address of a variable whereas the second pointer is used to store the address of the first pointer.

By – Mohammad Imran

Pointer

Syntax

```
Data_Type **Pointer_Name;
```

```
Example:     int **Ptr;
```

Pointer

Example - 1

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

OUTPUT

```
Address of a: 10234
Address of p: 20345
Value of p: 10
Value of pp: 10
```

By – Mohammad Imran

Pointer

Function Pointer

As we know that we can create a pointer of any data type such as int, char, float, we can also create a pointer pointing to a function. The code of a function always resides in memory, which means that the function has some address. We can get the address of memory by using the function pointer.

By – Mohammad Imran

Function Pointer

Simple Example

```
#include<iostream>
using namespace std;
int main()
{
    cout <<"Address of main() function is " << main;
    return 0;
}
```

By – Mohammad Imran

Function Pointer

Syntax

```
return_type (*pointer_name)(type1, type2);
```

Example:

```
int (*fptr) (int);
```

Function Pointer

Example - 1

```
#include<iostream>
using namespace std;
void fun(int a)
{
   cout << "Value of a is " << a;
}
int main()
{
   void (*fun_ptr)(int) = &fun;
   (*fun_ptr)(10);
   return 0;
}
```

By – Mohammad Imran

Function Pointer

Example - 2

```
#include <iostream>
using namespace std;
void printname(char *name)
{
    cout << "Name is :" <<name<< endl;
}
int main()
{
   char s[20];
   void (*ptr)(char*);
   ptr = printname;
   cout << "Enter the name of the person: ";
   cin >> s;
   ptr(s);
   return 0;
}
```

By – Mohammad Imran

QUIZ

By – Mohammad Imran

Quiz - 1

```
#include <iostream>
using namespace std;

void fun(int x)
{
    x = 30;
}

int main()
{
   int y = 20;
   fun(y);
   cout << y;
   return 0;
}
```

OUTPUT

20

Click here to see code

By – Mohammad Imran

Quiz - 2

```
#include <iostream>
using namespace std;

void fun(int *ptr)
{
   *ptr = 30;
   *ptr = *ptr + 5;
}
int main()
{
   int y = 20;
   fun(&y);
   cout << y;
   return 0;
}
```

OUTPUT

35

Click here to see code

By – Mohammad Imran

Coding Questions

By – Mohammad Imran

Question - 1

Write a program to take two input from the user and store the address of variable into pointer and find the sum of the numbers using pointer.

Click here to see code

By – Mohammad Imran

Question - 2

Write a program to take two input from the user and store the address of variable into pointer and find the sum of the numbers using **call by reference**.

Click here to see code

By – Mohammad Imran

Question - 3

Write a program that initializes integer values to an array and displays the value of the array on the screen using pointer notation and pointer arithmetic in C++:

Click here to see code

By – Mohammad Imran

Question - 4

Write a program that inputs values into an array and displays the odd value of array on screen using pointer notation and pointer arithmetic in C++:

Click here to see code

By – Mohammad Imran

Question - 5

Write a program that inputs value into array and display in reverse order on screen using pointer notation and pointer arithmetic in C++:

Click here to see code

By – Mohammad Imran

Question - 5

Write a C++ program to create a pointer array and display the element of an array using pointer.

Click here to see code

By – Mohammad Imran

Coding Questions

1. Write a C++ program to find cube of any number using function.
2. Write a C++ program to find diameter, circumference and area of circle using functions.
3. Write a C++ program to find maximum and minimum between two numbers using functions.
4. Write a C++ program to check whether a number is even or odd using functions.
5. Write a C++ program to check whether a number is prime, Armstrong or perfect number using functions.
6. Write a C++ program to find all prime numbers between given interval using functions.
7. Write a C++ program to print all strong numbers between given interval using functions.
8. Write a C++ program to print all Armstrong numbers between given interval using functions.
9. Write a C++ program to print all perfect numbers between given interval using functions.

By – Mohammad Imran

Coding Questions

10. Write a C++ program to find power of any number using recursion.
11. Write a C++ program to calculate area of square using return function.
12. Write a program to calculate area and parameter of rectangle.
13. Write a program to check if entered character is alphabet or not using return function with argument.
14. Write a C program to check entered number is even or odd using return with argument function.
15. Write a C program to print sizes of different types of pointer (Like integer, character, long int, float, double etc.
16. Write a program to demonstrate double pointer.
17. Write a program to change the value of a variable using pointer.

By – Mohammad Imran

Coding Questions

18. Write a program to print the Fibonacci series up to specified number.
19. Write a C++ program to call simple method using function pointer.
20.