C++- Pointers and References

Pointers

Every variable is a memory location and every memory location has its address defined which can be accessed using ampersand (&) operator which denotes an address in memory. Consider the following which will print the address of the variables defined:

```
#include <iostream>
using namespace std;
int main () {
   int  var1;
   char var2[10];
   cout << "Address of var1 variable: ";
   cout << &var1 << endl;
   cout << "Address of var2 variable: ";
   cout << &var2 << endl;
   return 0;
}
```

When the above code is compiled and executed, it produces the following result:

```
Address of var1 variable: 0xbfebd5c0
Address of var2 variable: 0xbfebd5b6
```

What are Pointers?

A pointer is a variable whose value is the address of another variable. Like any variable or constant, you must declare a pointer before you can work with it. The general form of a pointer variable declaration is:

```
type *var-name;
```

Here, type is the pointer's base type; it must be a valid C++ type and var-name is the name of the pointer variable. The asterisk you used to declare a pointer is the same asterisk that you use for multiplication.

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

Using Pointers in C++

```
#include <iostream>
using namespace std;
int main () {
   int  var = 20;   // actual variable declaration.
   int  *ip;        // pointer variable
   ip = &var;       // store address of var in pointer variable
   cout << "Value of var variable: ";
   cout << var << endl;
   // print the address stored in ip pointer variable
   cout << "Address stored in ip variable: ";
   cout << ip << endl;
   // access the value at the address available in pointer
   cout << "Value of *ip variable: ";
   cout << *ip << endl;
```

```
    return 0;
}
```

Dynamic memory

Programs need to dynamically allocate memory, for which the C++ language integrates the operators new and delete.

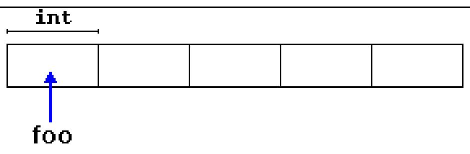
Operators *new* and *new[]*

Dynamic memory is allocated using operator new. new is followed by a data type specifier and, if a sequence of more than one element is required, the number of these within brackets []. It returns a pointer to the beginning of the new block of memory allocated. Its syntax is:

```
pointer = new type;
pointer = new type [number_of_elements];
```

The first expression is used to allocate memory to contain one single element of type *type*. The second one is used to allocate a block (an array) of elements of type *type*, where *number_of_elements* is an integer value representing the amount of these. For example:

```
int * foo;
foo = new int [5];
```



Operators *delete* and *delete[]*

In most cases, memory allocated dynamically is only needed during specific periods of time within a program; once it is no longer needed, it can be freed so that the memory becomes available again for other requests of dynamic memory. This is the purpose of operator delete, whose syntax is:

```
delete pointer;
delete[] pointer;
```

References

A reference variable is an alias, that is, another name for an already existing variable. Once a reference is initialized with a variable, either the variable name or the reference name may be used to refer to the variable.

References vs Pointers

References are often confused with pointers but three major differences between references and pointers are:

- You cannot have NULL references. You must always be able to assume that a reference is connected to a legitimate piece of storage.
- Once a reference is initialized to an object, it cannot be changed to refer to another object. Pointers can be pointed to another object at any time.
- A reference must be initialized when it is created. Pointers can be initialized at any time.

Creating References in C++

```
#include <iostream>
```

```
using namespace std;
int main () {
   // declare simple variables
   int    i;
   double d;
   // declare reference variables
   int&    r = i;
   double& s = d;
   i = 5;
   cout << "Value of i : " << i << endl;
   cout << "Value of i reference : " << r  << endl;
   d = 11.7;
   cout << "Value of d : " << d << endl;
   cout << "Value of d reference : " << s  << endl;
   return 0;
}
```

When the above code is compiled together and executed, it produces the following result:

```
Value of i : 5
Value of i reference : 5
Value of d : 11.7
Value of d reference : 11.7
```

Exercises

1. Write a program that asks the user to enter integers as inputs to be stored in the variables 'a' and 'b' respectively. There are also two integer pointers named ptrA and ptrB. Assign the values of 'a' and 'b' to ptrA and ptrB respectively and display them.
2. Write a program that allocate and free dynamically an array of 1000 integers and an array of 1234567 double.