Pointers

A pointer in C++ is a variable that holds address of another variable (of a specific type). Just like common variables, it is essential to declare pointer variable. Pointer declaration, address operator and indirection (dereferencing is explained in the following code)

les_09_code_01.cpp

```
1.    #include <iostream>
2.    using namespace std;

3.    int main()
4.    {
5.    int a; // a is an integer
6.    int *aPtr; // aPtr is an int * which is a pointer to an integer
7.
8.    a = 7; // assigned 7 to a
9.    aPtr = &a; // assign the address of a to aPtr
10.
11.   cout << "The address of a is " << &a
12.   << "\nThe value of aPtr is " << aPtr;
13.   cout << "\n\nThe value of a is " << a
14.   << "\nThe value of *aPtr is " << *aPtr;
15.   cout << "\n\nShowing that * and & are inverses of "
16.   << "each other.\n&*aPtr = " << &*aPtr
17.   << "\n*&aPtr = " << *&aPtr << endl;
18.   } // end main
```

Output

```
The address of a is 0x69feec
The value of aPtr is 0x69feec

The value of a is 7
The value of *aPtr is 7

Showing that * and & are inverses of each other.
&*aPtr = 0x69feec
*&aPtr = 0x69feec
```

Task: Declare pointer to char, bool, double, float data types.

les_09_code_02.cpp

```
1.    #include<iostream>
2.    using namespace std;

3.    int main()
4.    {
5.    double *num1Ptr = nullptr; //Initialize pointers to prevent
      pointing to unknown or uninitialized areas of memory.
6.    cout<<"num1Ptr holds : "<<num1Ptr<<endl;
7.    double num1 = 23.767;

8.    num1Ptr = &num1;
9.    cout<<"num1Ptr holds : "<<num1Ptr<<endl;

10.   return 0;
11.   }
```

`nullptr` is replacement of `NULL` in new `C++11`Std (You must enable C++11 in compiler settings)

Passing Pointer to Function

les_09_code_03.cpp

```
1.     #include<iostream>

2.     using namespace std;

3.     void cubeByReference( int * ); // prototype
4.     int main()
5.        {
6.        int number = 5;
7.        cout<<"number : "<<number;
8.        int *numPtr = &number;
9.        cubeByReference(numPtr);
10.       cout<<"\nnumber after cubing : "<<number;

11.       return 0;
12.       }

13.    void cubeByReference( int *nPtr )
14.       {
15.       *nPtr = *nPtr * *nPtr * *nPtr; // cube *nPtr
16.       }
```

Output

```
number : 5
number after cubing : 125
```

Relationship Between Array and Pointers

les_09_code_04.cpp

```
1.    #include<iostream>
2.    using namespace std;

3.    int main()
4.    {

5.       double array[10];

6.       for(int i=0; i<10;i++)
7.       {
8.       array[i] = (i + 2)/.512;
9.       }
10.      cout<<"Printing Array Directly"<<endl;

11.      for(int i=0; i<10;i++)
12.      {
13.      cout<<"\n Element "<<i<<" : "<<array[i];
14.      }

15.      cout<<"\nPrinting array via pointer\n";
16.      for(int i = 0; i<10; i++)
17.      {
18.      cout<<"\n Element "<<i<<" : "<<*(array + i);
19.      }
20.      return 0;

21.   }
```

Output

```
Printing Array Directly

 Element 0 : 3.90625
 Element 1 : 5.85938
 Element 2 : 7.8125
 Element 3 : 9.76562
 Element 4 : 11.7188
 Element 5 : 13.6719
 Element 6 : 15.625
 Element 7 : 17.5781
 Element 8 : 19.5312
 Element 9 : 21.4844
```

```
Printing array via pointer

 Element 0 : 3.90625
 Element 1 : 5.85938
 Element 2 : 7.8125
 Element 3 : 9.76562
 Element 4 : 11.7188
 Element 5 : 13.6719
 Element 6 : 15.625
 Element 7 : 17.5781
 Element 8 : 19.5312
 Element 9 : 21.4844
```

Pointer Arithmetic

les_09_code_05.cpp

```
1.    #include<iostream>
2.    using namespace std;

3.    int main()
4.    {
5.       double array[10];
6.       for(int i = 0; i<10; i++)
7.       {
8.       cout<<"\n Address of Element "<<i<<" : "<<(array + i);
9.       }
10.      return 0;
11.   }
```

Output

```
 Address of Element 0 : 0x69fea8
 Address of Element 1 : 0x69feb0
 Address of Element 2 : 0x69feb8
 Address of Element 3 : 0x69fec0
 Address of Element 4 : 0x69fec8
 Address of Element 5 : 0x69fed0
 Address of Element 6 : 0x69fed8
 Address of Element 7 : 0x69fee0
 Address of Element 8 : 0x69fee8
 Address of Element 9 : 0x69fef0
```