Introduction to Pointers
Lecture 2

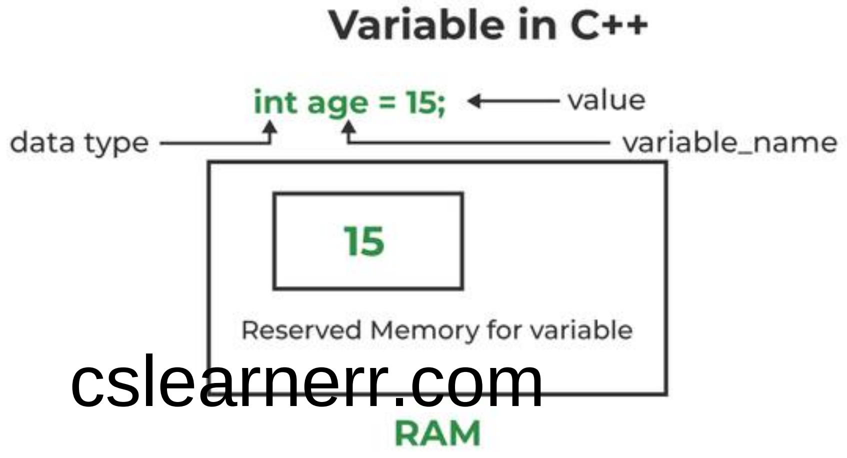
Background: Variables

Variable :

A variable is a name for a piece of memory that holds a value. When a program code is in its execution phase a free memory address is automatically assigned to a variable to store a specified type of data.

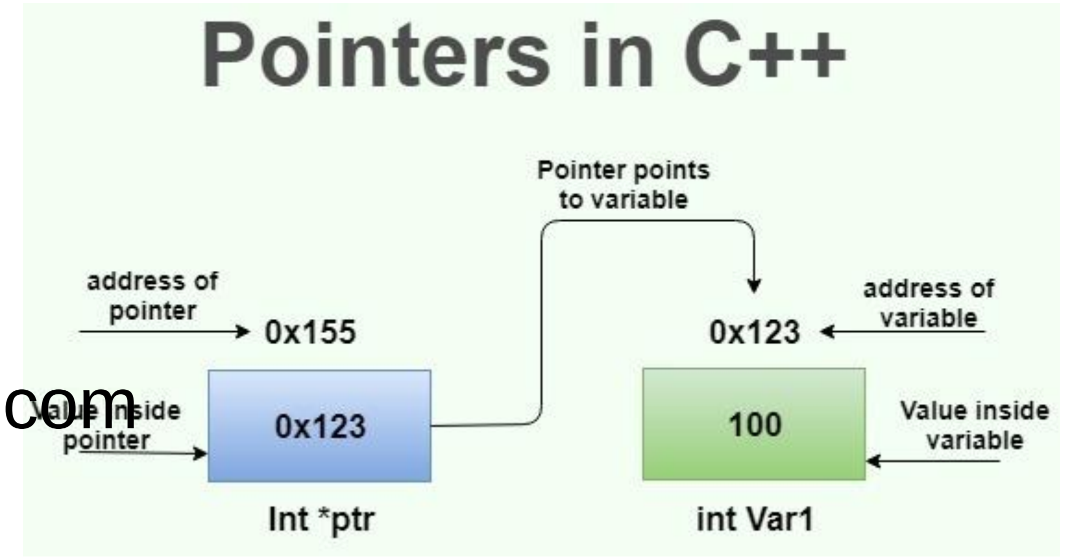
Properties attached to variable when it is created :

- Type
- Name
- Memory address assigned to it.



Pointers

1. A pointer is a variable that stores the memory address of an object.
2. Pointers are used to store the addresses of other variables or memory items.
3. A pointer is declared using a asterisk (*).
   1. For-example: int *p1;
      float *p2;
4. Each pointer point to a variable of only one type. For-example in above declaration p1 can point to integer type only and not float.



Address or Reference Operator(&):

- The address operator(&):

The memory address where the contents of a specified variable are stored can be accessed by the use of address operator.

Referencing to next slide#5:

```
cout<<"\nThe address of memory location of this variable is= "<<&x;
```

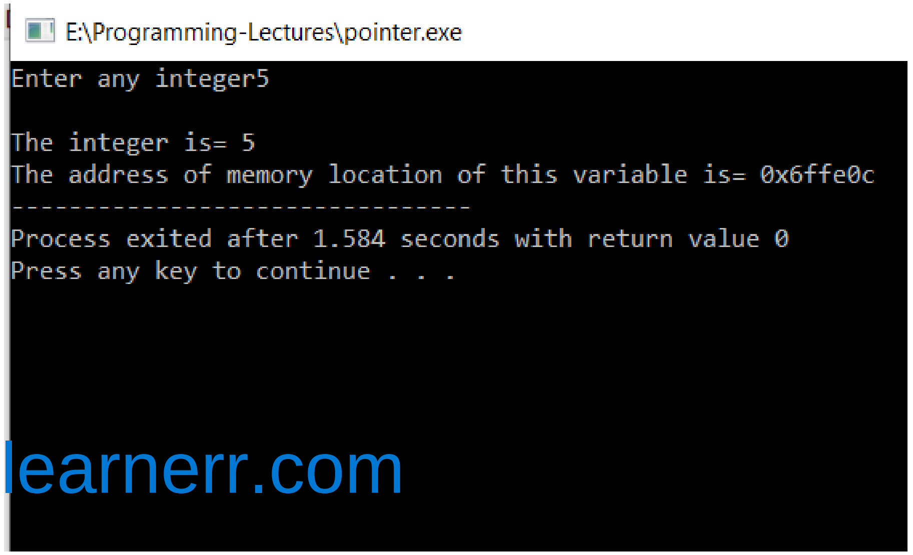
The address is printed in hexadecimal format

Example: Use of address operator(&)

PROGRAM CODE

```
#include<iostream>
using namespace std;
int main()
{
        int x;
        cout<<"Enter any integer";
        cin>>x;
        cout<<"\nThe integer is= "<<x;
        cout<<"\nThe address of memory location of this variable is= "<<&x;
        return 0;
}
```

OUTPUT

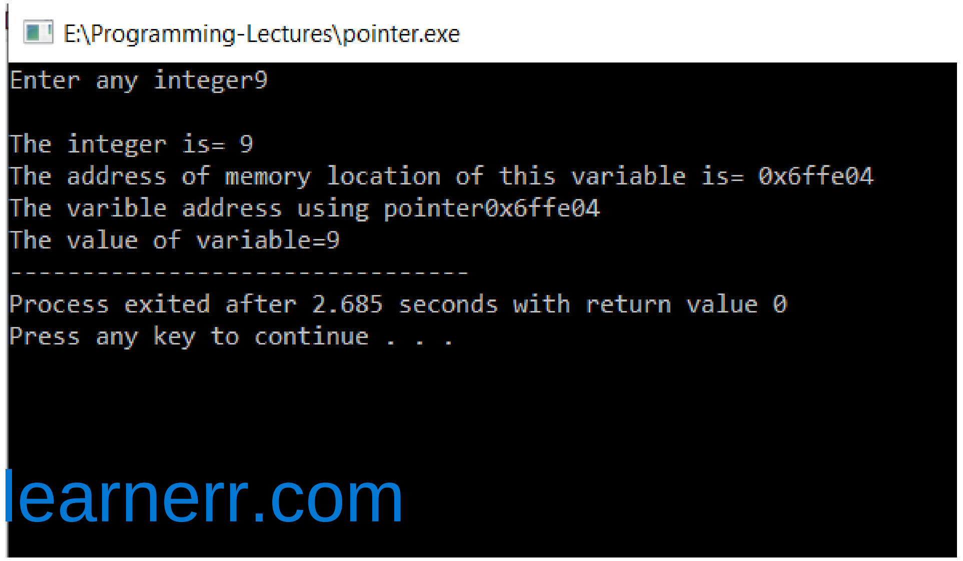


E:\Programming-Lectures\pointer.exe

```
Enter any integer5

The integer is= 5
The address of memory location of this variable is= 0x6ffe0c
--------------------------------
Process exited after 1.584 seconds with return value 0
Press any key to continue . . .
```

Example 2: Use of Pointers

```
#include<iostream>

using namespace std;

int main()

{

                    int x , *p1;

                    p1=&x;

                    cout<<"Enter any integer";

                    cin>>x;

                    cout<<"\nThe integer is= "<<x;

                    cout<<"\nThe address of memory location of this variable is= "<<&x;

                    cout<<"\nThe varible address using pointer"<<p1;

                    cout<<"\nThe value of variable="<<*p1;

                    return 0;

}
```



E:\Programming-Lectures\pointer.exe

```
Enter any integer9

The integer is= 9
The address of memory location of this variable is= 0x6ffe04
The varible address using pointer0x6ffe04
The value of variable=9
--------------------------------
Process exited after 2.685 seconds with return value 0
Press any key to continue . . .
```

Pointer Declaration Types

Pointers are declared as follows:

<type> * variable_name ;

int *p1;   //a pointer to data of type integer value

double *p2   //a pointer to data of type double value

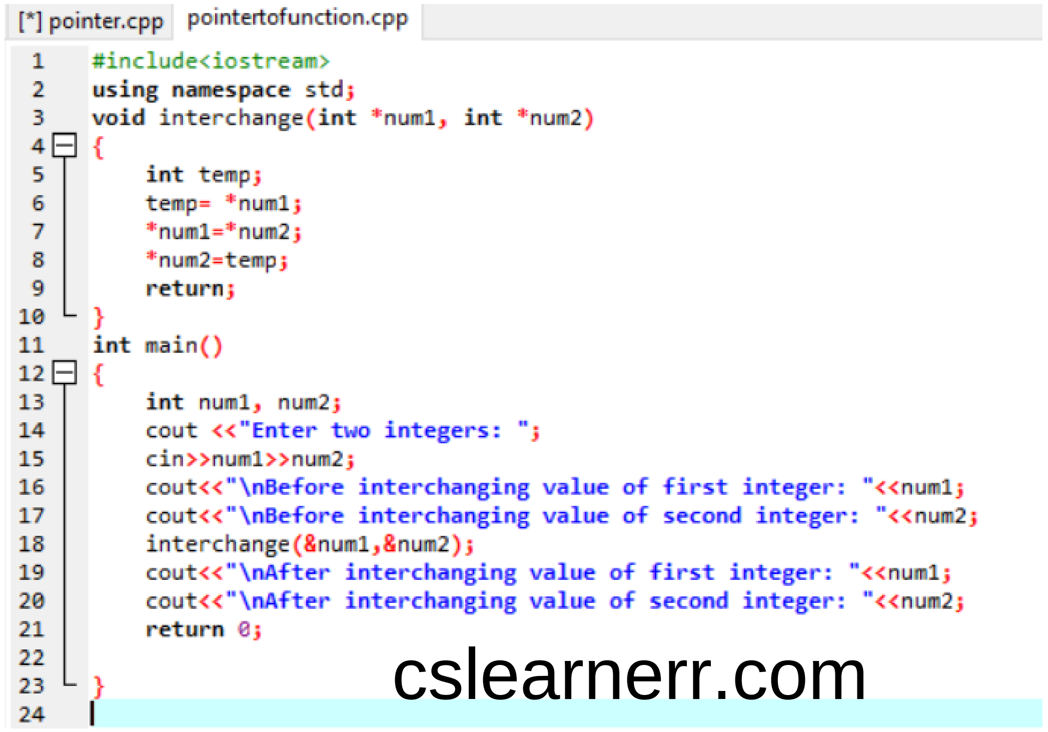
int* p1   //valid syntax(acceptable but not favored)

int *p1,*p2;   //declare two pointer to integer variable

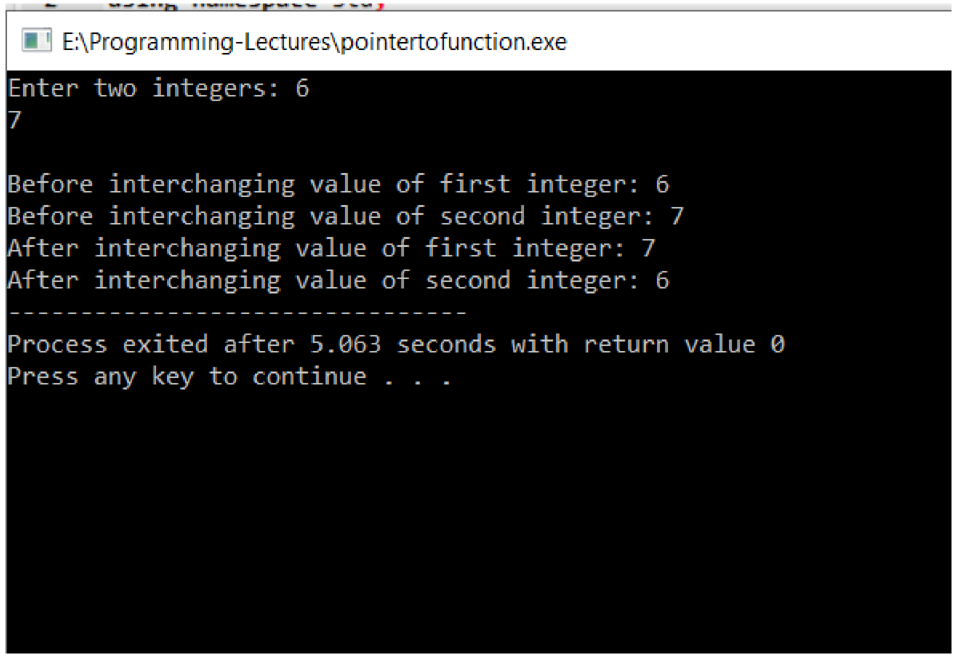
Pointers with Function

The **call by pointer** method of passing arguments to a function copies the address of an argument into the formal parameter. Inside the function, the address is used to access the actual argument used in the call. This means that changes made to the parameter affect the passed argument.

Example : The function parameters as pointer types as in the following function **interchange()**, which exchanges the values of the two integer variables pointed to by its arguments.


[*] pointer.cpp | pointertofunction.cpp

```cpp
#include<iostream>
using namespace std;
void interchange(int *num1, int *num2)
{
     int temp;
     temp= *num1;
     *num1=*num2;
     *num2=temp;
     return;
}
int main()
{
     int num1, num2;
     cout <<"Enter two integers: ";
     cin>>num1>>num2;
     cout<<"\nBefore interchanging value of first integer: "<<num1;
     cout<<"\nBefore interchanging value of second integer: "<<num2;
     interchange(&num1,&num2);
     cout<<"\nAfter interchanging value of first integer: "<<num1;
     cout<<"\nAfter interchanging value of second integer: "<<num2;
     return 0;

}

```


E:\Programming-Lectures\pointertofunction.exe

```
Enter two integers: 6
7

Before interchanging value of first integer: 6
Before interchanging value of second integer: 7
After interchanging value of first integer: 7
After interchanging value of second integer: 6
--------------------------------
Process exited after 5.063 seconds with return value 0
Press any key to continue . . .
```

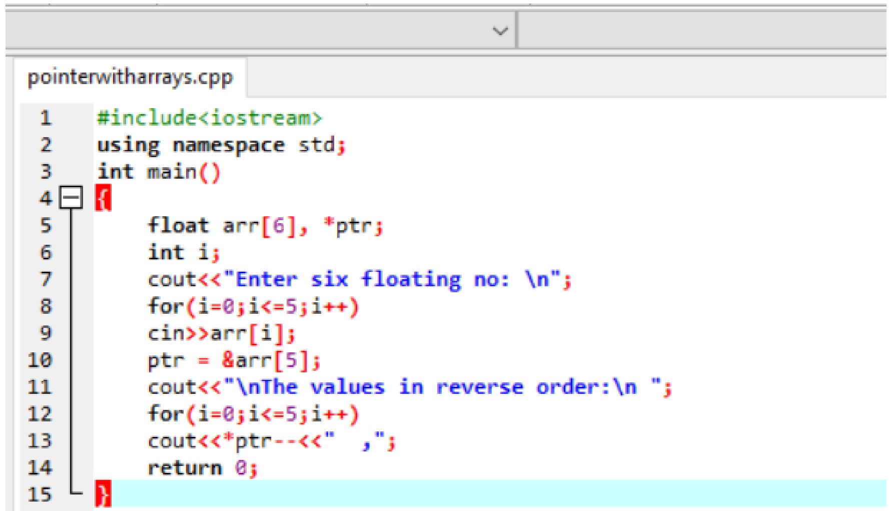
Pointer and Arrays

- Pointers can be used with array.
- A pointer can access all elements of an array if the address of the first element is assigned to it.
- The name of the array represents the address if its first element. If the name of the array is assigned to pointer then the pointer can access all elements of that array.
- Example:
  - int num[10], *ptr;
  - ptr=num;

- The pointer can be moved backward and forward by using increment ++ and decrement - - operator.
- Example :
  - int num[10], *ptr;
  - ptr=num;
  - cout<<*ptr;
  - ptr++;
  - cout<<*ptr;

Pointer with Array Example:

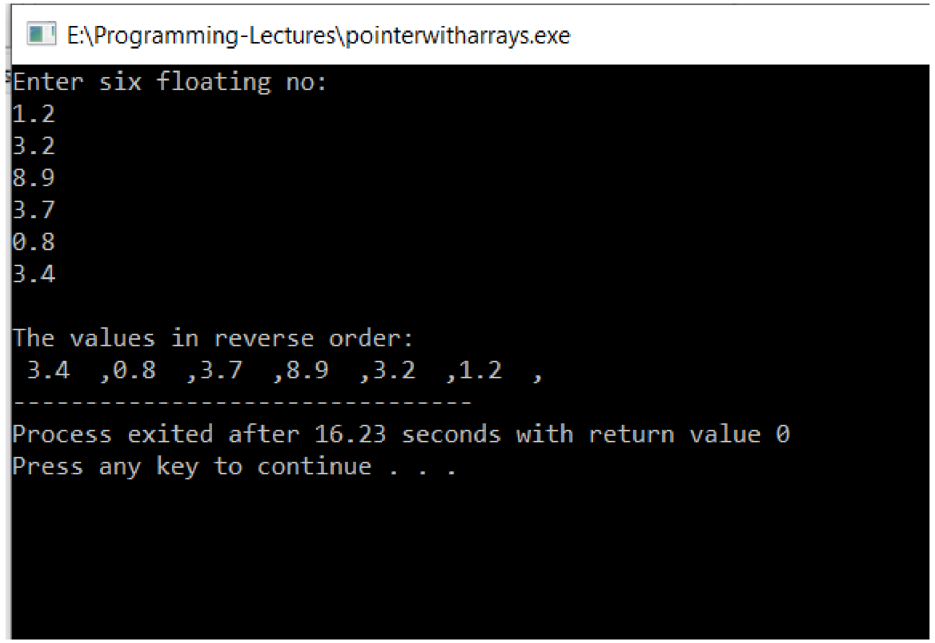
CODE: TO PRINT 6 FLOATING POINT NO. IN REVERSE ORDER USING POINTERS



pointerwitharrays.cpp

```cpp
#include<iostream>
using namespace std;
int main()
{
    float arr[6], *ptr;
    int i;
    cout<<"Enter six floating no: \n";
    for(i=0;i<=5;i++)
    cin>>arr[i];
    ptr = &arr[5];
    cout<<"\nThe values in reverse order:\n ";
    for(i=0;i<=5;i++)
    cout<<*ptr--<<"  ,";
    return 0;
}
```

OUTPUT:



E:\Programming-Lectures\pointerwitharrays.exe

```
Enter six floating no:
1.2
3.2
8.9
3.7
0.8
3.4

The values in reverse order:
 3.4  ,0.8  ,3.7  ,8.9  ,3.2  ,1.2  ,
--------------------------------
Process exited after 16.23 seconds with return value 0
Press any key to continue . . .
```