SLIIT ACADEMY

Higher Diploma in Information Technology
Year 1, Semester 1

Introduction to Programming(C++)

Lecture 09 : Pointers in C++

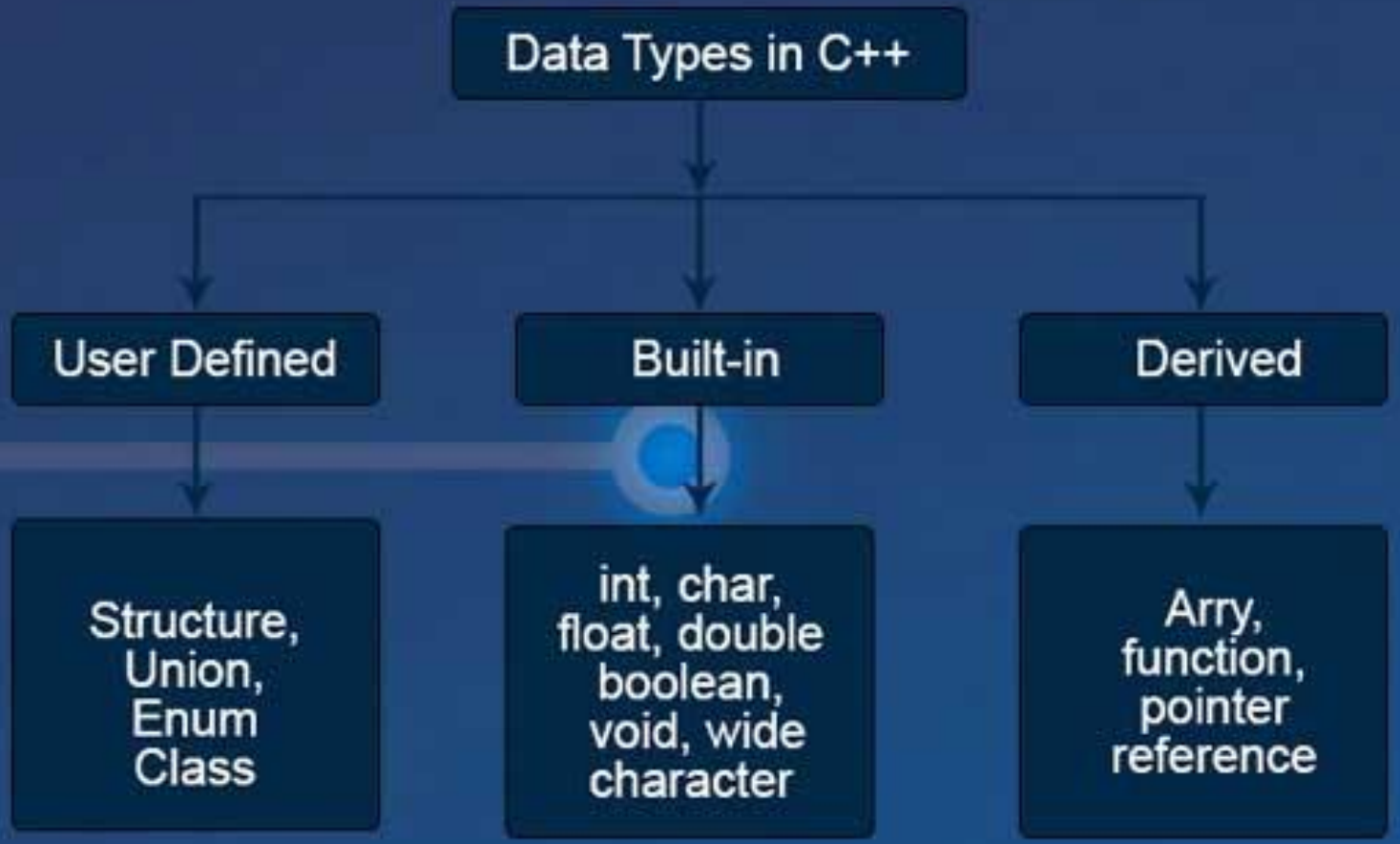
Intended Learning Outcomes

End of this lecture you will be able to learn ,

- **LO1 : Working with Derived data types in C++**
- **LO2 : Understand the usage of pointers in C++**
- **LO3 : Understand the how to work with pointers and arrays**

SLIIT ACADEMY

C++ Data Types



Memory Address in C++

- During program execution, each object (such as a variable or an array) is located somewhere in an area of memory. The location of an object in the memory is called its address.
- The memory address is the location of where the variable is stored on the computer.
- The & operator can also be used to get the memory address of a variable; which is the location of where the variable is stored on the computer.

SLIIT ACADEMY

Printing Variable Addresses in C++

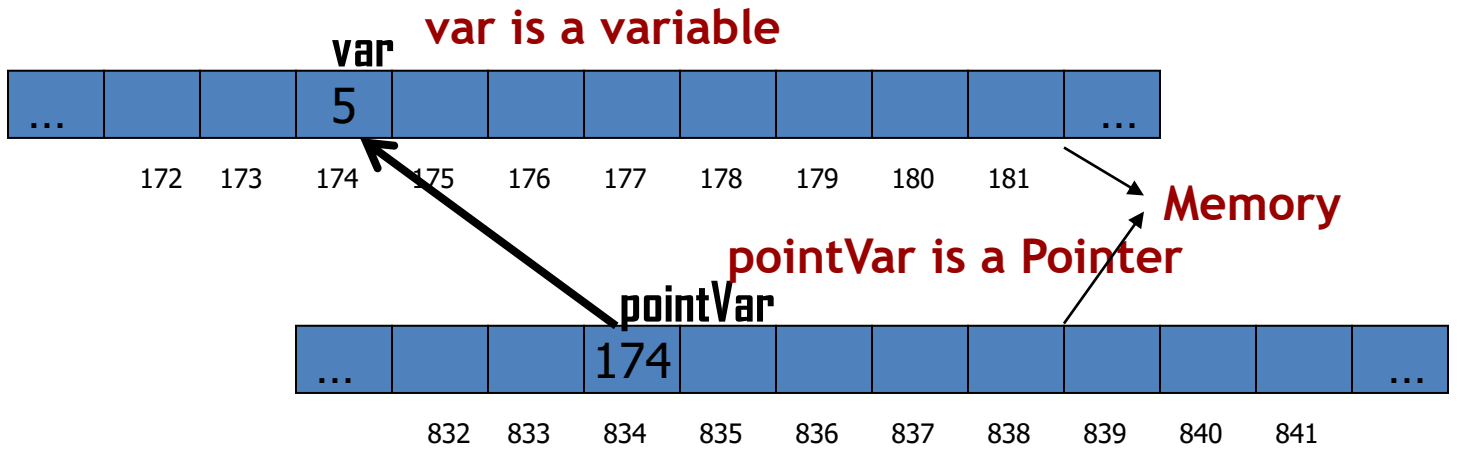
```
#include <iostream>
using namespace std;
int main(){
    // declare variables
    int var1 = 3;
    int var2 = 24;
    int var3 = 17;

    // print address of var1
    cout << "Address of var1: "<< &var1 << endl;

    // print address of var2
    cout << "Address of var2: " << &var2 << endl;

    // print address of var3
    cout << "Address of var3: " << &var3 << endl;
}
```

Memory Address in C++

- C++ offers two mechanisms for dealing with addresses:
  - **Pointers(*)**
  - **References (&)**
- Pointers are an older mechanism that C++ inherited from C, while references are a newer mechanism that originated in C++.
- Unfortunately, many C++ library functions use pointers as arguments and return values, making it necessary for programmers to understand both mechanisms.

Introduction to Pointes in C++

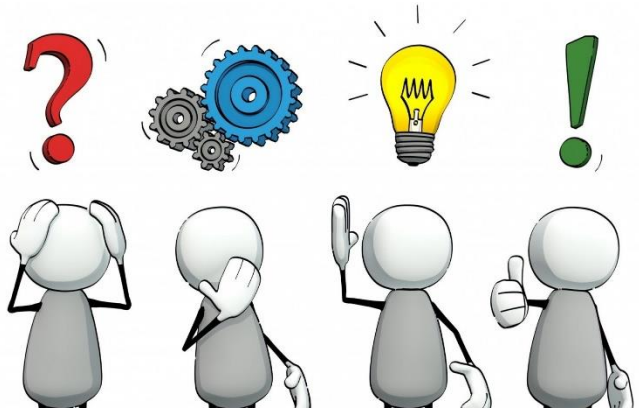
- The pointer in C++ language is a variable, it is also known as **locator or indicator** that points to an address of a value.
- The main difference between normal variables and pointers is that **variables contain a value**, while **pointers contain a memory address.**



SLIIT ACADEMY

Why we need Pointers ?

- Pointers give the ability to manipulate the data in the computer's memory ,which can reduce the code and improve the performance.



SLIIT ACADEMY

C++ Pointer Operators

- C++ provides two pointer operators, They are
  - **Address of Operator ➔ &**

    The & is a unary operator that returns the memory address of its operand.
  - **Indirection Operator ➔ ***

    Indirection Operator *, and it is the complement of &. It is a unary operator that returns the value of the variable located at the address specified by its operand.

SLIIT ACADEMY

C++ Pointers

Declare C++ Pointers

- A **pointer variable** must be declared before it can be used.

Method 01

```
int *pointVar;
```

Method 02

```
int* pointVar;
```

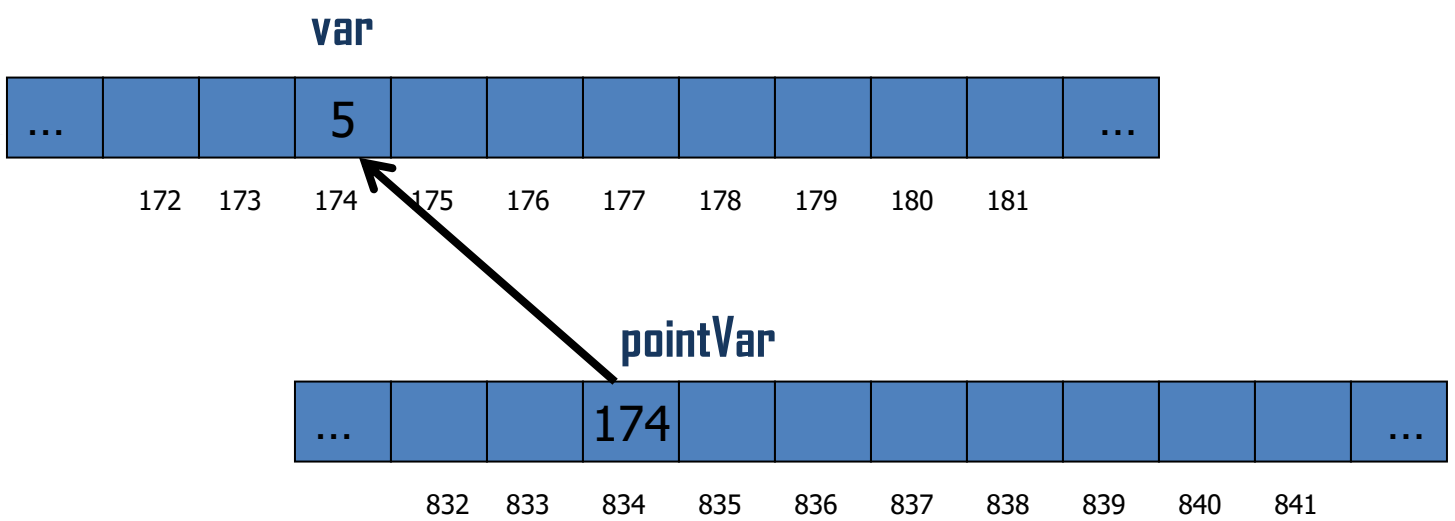
- The **asterisk(*)**, operator is used after the data type to declare pointers. It tells the compiler that the variable is to be a pointer, and the type of data (the value that we get indirectly) that the pointer points to.

SLIIT ACADEMY

Referencing in C++

- Assigning Addresses to Pointers

```
int *pointVar, var;
var = 5;
// assign address of var to pointVar pointer
pointVar = &var;
```



SLIIT ACADEMY

Dereferencing in C++

- To get the value pointed by a pointer, we use the * operator. When * is used with pointers, it's called the **dereference operator.**
- It operates on a pointer and gives the value pointed by the address stored in the pointer.

```
int *pointVar, var = 5;
// assign address of var to pointVar
pointVar = &var;

// access value pointed by pointVar
cout << *pointVar << endl;   // Output: 5
```

SLIIT ACADEMY

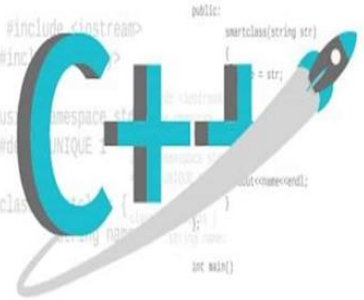
Practice Question 01

What is the output of the following C++ program?

```cpp
#include <iostream>
using namespace std;
int main()
{
    int a = 5;
    int *ptr = &a;

    *ptr = 20;
    cout<<a<<endl;

    return 0;
}
```

```cpp
#include <iostream>
using namespace std;
int main() {

    int m = 10;
    int *p = &m;
    m = *p+10;
    cout<<m<<endl;

    return 0;
}
```

```cpp
#include <iostream>
using namespace std;
int main()
{
    int *ptr;
    int x = 5;

    ptr = &x;
    cout<<x<<" "<<*ptr<<endl;

    *ptr += 5;
    cout<<x-1<<" "<<*ptr<<endl;

    (*ptr)++;
    cout<<x+1<<" "<<*ptr<<endl;

    return 0;
}
```



SLIIT ACADEMY

References in C++

- A reference variable is **an alias or 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.
- A reference variable is a "reference" to an existing variable, and it is created with the & operator.

```
string food = "Pizza";   // food variable
string &meal = food;     // reference to food
```

SLIIT ACADEMY

C++ Pointers and Arrays

- There is a close relationship between array and pointers in C++
- In C++, Pointers are variables that hold addresses of other variables.
- Not only can a pointer store the address of a single variable, but it can also store the address of cells of an array.

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

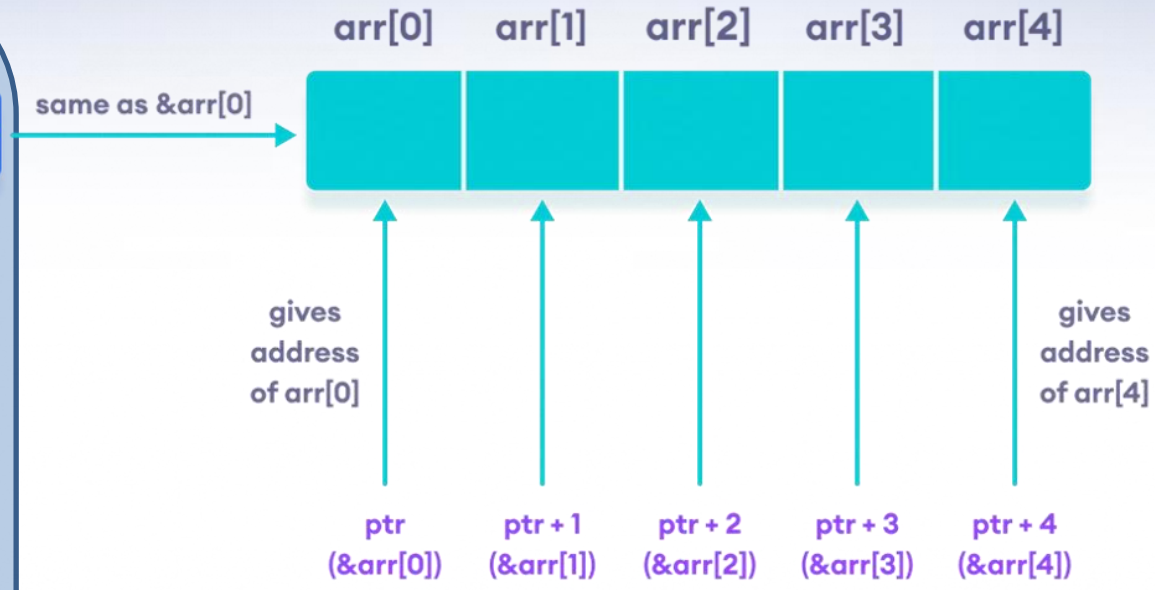
```
int *ptr;
int arr[5];
ptr = &arr[0];
```

SLIIT ACADEMY

Access Array Elements with pointers

```
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];
```

ptr = arr



SLIIT ACADEMY

Access Array Elements with pointers

Similarly, we can access the elements using the single 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];
```

SLIIT ACADEMY

C++ Pointers and Arrays

- When we declare an array, its name is treated as a constant pointer to the first element of the array.
- This is also known as the **base address of the array.**
- In other words, base address is the address of the first element in the array of the address of arr[0].

```
int arr[5]= {10,20,30,40,50};
cout<< *(arr) <<"\t";
cout<< *(arr+1) <<"\t";
cout<< *(arr+2) <<"\t";
cout<< *(arr+3) <<"\t";
cout<< *(arr+4) <<"\t";
```

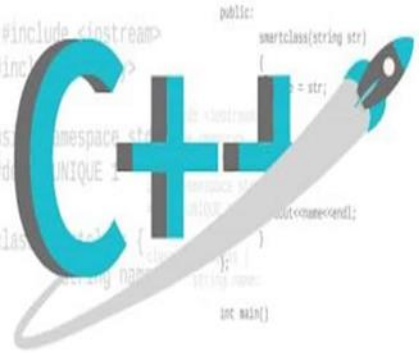
SLIIT ACADEMY

Practice Question 02

What is the output of the following C++ program?

```cpp
#include <iostream>
using namespace std;
int main() {

    int *ptr;
    int arr[] = {5, 10, 15};
    ptr = arr;
    cout<<*ptr<<endl;
    cout<<*(ptr+1)<<endl;
    cout<<*ptr+1<<endl;

    return 0;
}
```

```cpp
#include <iostream>
using namespace std;
int main() {

    int arr[] = {4, 5, 6, 7};
    int *p = arr;

     ++*p;
   cout<<*p<< endl;

    p += 2;
    cout<<*p<< endl;

    return 0;
}
```

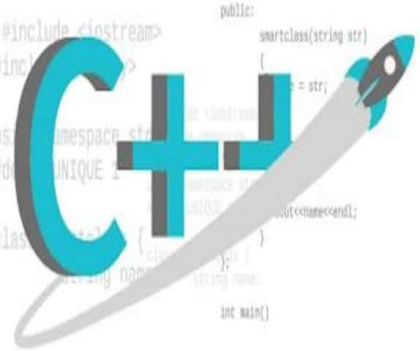


SLIIT ACADEMY

Practice Question 03

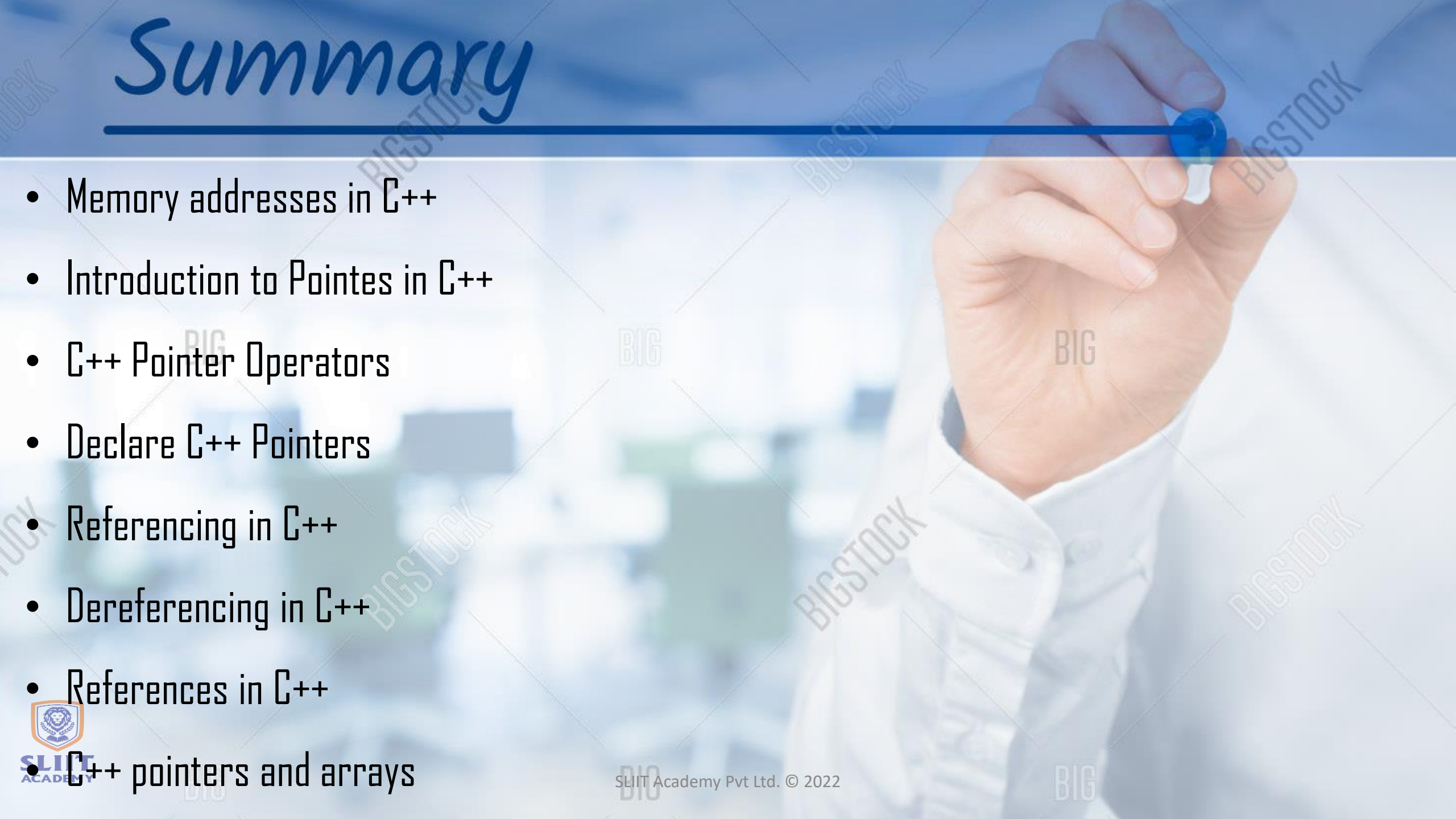
What is the output of the following C++ program?

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

int main()
{
    float arr[5] = {12.5, 10.0, 13.5, 90.5, 0.5};
    float *ptr1 = &arr[0];
    float *ptr2 = ptr1 + 3;

    cout<<*ptr2<<endl;
    cout<< ptr2 - ptr1<<endl;

   return 0;
}
```



SLIIT ACADEMY

Summary



- Memory addresses in C++
- Introduction to Pointes in C++
- C++ Pointer Operators
- Declare C++ Pointers
- Referencing in C++
- Dereferencing in C++
- References in C++
- C++ pointers and arrays