AMBITION GURU

C++ Programming (HCAC-154)

II semester, BCA

Compiled by Ankit Bhattarai
Ambition Guru College

Syllabus

AMBITION GURU

Unit	Contents	Hours	Remarks
1.	OOP Basics	9	
2.	Classes & Objects	10	
3.	Overloading and Inheritance	10	
4.	**Pointers**	**8**	
5.	Virtual function and polymorphism	8	
6.	Templates and Exception Handling	7	
7.	I/O stream	8	

Practical Works

Credit hours : 3

AMBITION GURU

Unit 4

Pointers

(8 hrs.)

Addresses and pointers: The Address of Operator, Pointer Variables, Syntax Quibbles, Accessing the Variable pointed to , Pointer to void, Pointer and Arrays, Pointers and Functions, Pointers to Objects, Linked List Example: A chain of pointers, Adding an Item to the list, Displaying the List Contents, Pointer to Pointer

Introduction

AMBITION GURU

- A **pointer** in C++ is a special variable that stores the **memory address** of another variable.
- Declared using *

Declaration:

```
int *ptr;     // declares a pointer to int
```

Access (Dereferencing):

```
cout << *ptr;    // prints value stored in x (20)
cout << ptr;     // prints address of x
```

Address of Operator (&)

The & operator gives the address of a variable.

Example:

```
int x = 10;
cout << &x;   // Prints the address of x
```

Syntax Quibbles

The symbol * is used in two contexts:

1. Declaration → means “pointer to”

```
int *p;
```

2. Dereferencing → means “value at”

```
cout << *p;
```

Example of using pointers:

AMBITION GURU

```
#include <iostream>
using namespace std;
int main() {
    int num = 50;        // normal variable
    int *p;              // pointer declaration
    p = &num;            // pointer initialization
    cout << "Value of num: " << num << endl;
    cout << "Address of num: " << &num << endl;
    cout << "Pointer p stores: " << p << endl;
    cout << "Value accessed using pointer: " << *p << endl;
    return 0;
}
```

Example of using pointers:

AMBITION GURU

```
Output:
Value of num: 50
Address of num: 0x7ffeefbff45c
Pointer p stores: 0x7ffeefbff45c
Value accessed using pointer: 50
```

Manipulation of pointers

AMBITION GURU

- We can manipulate a pointer with the indirection operator i.e. * which is also known as deference operator.
- With this operator, we can indirectly access the data variable content.
- It takes the following general form:

```
* pointer_variable
```

Manipulation of pointers

AMBITION GURU

```
#include<iostream>
using namespace std;
int main()
{
    int a = 10;
    int *ptr;
    ptr=&a;
    cout<<"The value of a is: "<<*ptr;
    *ptr = *ptr+a;
    cout<<"\n The revised value of a is"<<a;
    return 0;
}
```

Output:

The value of a is : 10

The revised value of a is : 20

Pointer to void

Pointer to void

- A void* (generic pointer) is a special type of pointer that can store the address of any data type.
- Since it has no specific type, it cannot be directly dereferenced.
- void* pointer can hold addresses of different data types, but before dereferencing, we must type-cast it.

AMBITION GURU

Pointer to void:
Example - Using void* with Different Data Types

```
#include <iostream>
using namespace std;
int main() {
    int a = 10;
    float b = 5.5;
    char c = 'Z';
    void *ptr;    // generic pointer
```

```
    // Pointing to int
    ptr = &a;
    cout << "Value of a = " << *((int*)ptr) << endl;

    // Pointing to float
    ptr = &b;
    cout << "Value of b = " << *((float*)ptr) << endl;

    // Pointing to char
    ptr = &c;
    cout << "Value of c = " << *((char*)ptr) << endl;

    return 0;
}
```

Output:

Value of a = 10

Value of b = 5.5

Value of c = Z

Pointer and arrays

Pointer and arrays

AMBITION GURU

- An array name acts as a constant pointer to its first element.
- We can assign an array to a pointer as follows:

```
int *ptr;
int arr[5] = {10, 20, 30, 40, 50};
ptr = &arr[0];
```

- We can use pointers to access and traverse arrays using arithmetic (p+1, p+2, …).

Example:
WAP for accessing array elements with pointers in C++

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

int main() {
    int arr[5] = {10, 20, 30, 40, 50};

    int *p = arr;

    cout << "Accessing elements using pointer arithmetic:" << endl;
```

AMBITION GURU

Example:
WAP for accessing array elements with pointers in C++

```
    for (int i = 0; i < 5; i++) {
        cout << "arr[" << i << "] = " << *(p + i) << endl;
    }

    cout << "\nAccessing elements using array notation:" << endl;

    for (int i = 0; i < 5; i++) {
        cout << "arr[" << i << "] = " << arr[i] << endl;
    }

    return 0;
}
```

Output:

```
Accessing elements using pointer arithmetic:
arr[0] = 10
arr[1] = 20
arr[2] = 30
arr[3] = 40
arr[4] = 50

Accessing elements using array notation:
arr[0] = 10
arr[1] = 20
arr[2] = 30
arr[3] = 40
arr[4] = 50
```

Note:

- This shows that arr[i] is equivalent to *(arr + i).
- The array name is not a normal variable pointer (you cannot reassign it), but it behaves like a pointer to the first element.

Pointer and Functions

Pointer and Functions

AMBITION GURU

- By default, C++ functions use *call by value* in which function gets a copy of the variable.
- Using pointers, we can achieve *call by reference* in which function directly accesses and modifies the original variable.
- In addition, C++ allows *function pointers* which is a pointer that can store the address of a function and call it indirectly.

AMBITION GURU

Example 1:
Passing Pointers to Functions (Call by Reference)

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

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

AMBITION GURU

Example 1:
Passing Pointers to Functions (Call by Reference)

```
int main() {
    int a = 10, b = 20;
    cout << "Before swap: a = " << a << ", b = " << b << endl;

    swap(&a, &b);  // pass addresses

    cout << "After swap: a = " << a << ", b = " << b << endl;
    return 0;
}
```

Output:

Before swap: a = 10, b = 20
After swap: a = 20, b = 10

Function Pointers

- The pointer to function is known as callback function.
- Using function pointers, we can allow a C++ program to select a function dynamically at run time.
- We can also pass a function as an argument to another function. Here, the function is passed as a pointer. The function pointers cannot be dereferenced.

We can declare a function pointer in C++:

```
return_type
(*pointer_name)(parameter_list);
```

AMBITION GURU

Example 2:
Declaring and Using a Function Pointer

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

int add(int x, int y) {
    return x + y;
}
```

```
int main() {
    // Declare a function pointer
    int (*fptr)(int, int);

    fptr = add;

    cout << "Sum = " << fptr(5, 7) << endl;
    cout << "Sum = " << (*fptr)(3, 4) << endl;
    return 0;
}
```

Output:

```
Sum = 12
Sum = 7
```

AMBITION GURU

Example 3:
Passing a Function Pointer to Another Function

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

int multiply(int x, int y)
{
    return x * y;
}
```

```
// Function that accepts a function pointer
void compute(int a, int b, int (*operation)(int, int)) {
    cout << "Result = " << operation(a, b) << endl;
}

int main() {
    compute(4, 5, multiply);
    return 0;
}
```

Output: Result = 20

Pointers to Objects

Pointers to Objects

AMBITION GURU

- A pointer to object is a pointer that stores the address of an object.
- Instead of using the dot operator (.), we use the arrow operator (->) to access members of the class through a pointer.

Syntax:

```
ClassName *ptr;           // Declare a pointer to class
ptr = &object;            // Store address of object
ptr->member_function();   // Access member function
ptr->data_member;         // Access data member (if public)
```

WAP in C++ to create a class Student with data members (name, marks) and demonstrate accessing members using pointer to object.

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

class Student {
    string name;
    int marks;

public:
   Student(string n, int m) {
        name = n;
        marks = m;
    }

    void display() {
        cout << "Name: " << name << ", Marks: " << marks << endl;
    }
};
```

```
int main() {
     Student s1(“Ram", 90);
    // Create a pointer to object
    Student *ptr;

    // Store address of object in pointer
    ptr = &s1;

    // Access function using arrow operator
    cout << "Accessing object using pointer:" << endl;
    ptr->display();
    return 0;
}
```

Output:

Accessing object using pointer:

Name: Ram, Marks: 90

Linked List Example: A chain of pointers, Adding an Item to the list, Displaying the List Contents

Introduction to Linked List

AMBITION GURU

- Linked List is a collection of nodes where each node consists of at least two parts:

  i. **Information field or info field** : It contains the actual element to be stored.

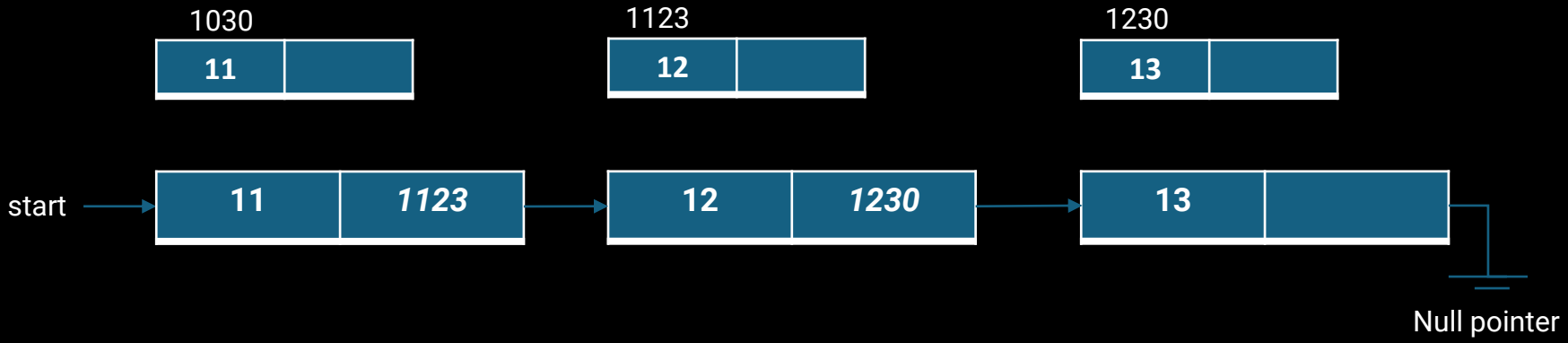
  ii. **Linked or address field**: It contains one or two links that points to the next node or previous node in the list.

Figure. A node

✓ The memory for nodes of the linked list can be allocated dynamically whenever required.

Introduction to Linked List

AMBITION GURU



Note:

- The first element of the list is known as **head** & the last element is known as **tail**.

Linked List: Structure

AMBITION GURU

Structure:

```
struct Node {
    int data;
    Node *next;
};
```

Linked List: Example

Add Item to Linked List & Display Contents

AMBITION GURU

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

// Define a Node
struct Node {
    int data;
    Node* next;
};
```

```
int main() {
    Node* head = NULL;
    Node* temp;

    // Adding first item
    Node* n1 = new Node;
    n1->data = 10;
    n1->next = NULL;
    head = n1;

    // Adding new item
    Node* n2 = new Node;
    n2->data = 20;
    n2->next = NULL;
    n1->next = n2;   // link first node to second
```

```
// Adding third item
    Node* n3 = new Node;
    n3->data = 30;
    n3->next = NULL;
    n2->next = n3;

    cout << "List contents: ";
    temp = head;
    while (temp != NULL) {
        cout << temp->data << " ";
        temp = temp->next;
    }
    cout << endl;

    return 0;
}
```

Pointer to Pointer

Pointer to Pointer

AMBITION GURU

- A *pointer to pointer* is a variable that stores the address of another pointer.
- It provides two levels of indirection:
  - *p → gives the address stored in the pointer.
  - **p → gives the actual value stored in the variable.

Pointer to Pointer

AMBITION GURU

Syntax:

```
datatype var = value;

datatype *ptr = &var;      // pointer to variable

datatype **pptr = &ptr;    // pointer to pointer
```

Basic Pointer to Pointer Example:

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

int main() {
    int a = 50;
    int *p = &a;     // pointer to int
    int **pp = &p;   // pointer to pointer
    cout << "Value of a  = " << a << endl;
    cout << "Address of a (&a) = " << &a << endl;
    cout << "Value of p (address of a) = " << p << endl;
    cout << "Value pointed by p (*p)   = " << *p << endl;
    cout << "Address of p (&p)         = " << &p << endl;
    cout << "Value of pp (address of p)= " << pp << endl;
    cout << "Value pointed by pp (*pp) = " << *pp << endl;
    cout << "Final value (**pp)        = " << **pp << endl;
    return 0;
}
```

AMBITION GURU

Questions:

1. Explain the concept of the address-of operator (&) and the dereference operator (*) with a simple example in C++.
2. What is a pointer variable? Write the syntax for declaring and initializing a pointer with an example.
3. Differentiate between an array name and a pointer. Give an example to illustrate how an array can be accessed using pointers.
4. Write a C++ program to demonstrate call by reference using pointers in functions.
5. What is a void pointer (void*)? Write a C++ program that stores and displays the address of an int and a float using a void pointer.
6. Explain pointer to pointer (double pointer) with a neat example program.
7. Write short notes on: (i) Pointer to objects, (ii) Arrow operator (->) in C++. Give an example program.
8. Write a program in C++ to add an item to a linked list and display the list contents using pointers.
9. Explain how function pointers are declared and used in C++. Give a small example program.
10. What is an array of pointers to objects? Write a simple C++ program to illustrate it.

THANK YOU

Any Queries ?