**East West University**
**Department of Computer Science and Engineering**
**Spring 2025**
**CSE207 – Data Structures, Section – 6**
**Course Instructor:** Ahmed Abdal Shafi Rasel (AASR)

---

# Transition from C to C++, Dynamic Memory Allocation, Pointers, Structures, Class

## A. Transition from C to C++:

```c
#include<stdio.h>

int main() {
    int n;
    scanf("%d", &n);
    printf("The number is %d\n", n);

    return 0;
}
```

code.c

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

int main() {
    int n;
    cin >> n;
    cout << "The number is " << n << endl;

    return 0;
}
```

code.cpp

To learn C++ in depth, you can check the C++ documentation by clicking here. If you want to write scanf and printf in c++, just add the header `#include<bits/stdc++.h>` instead of `#include<iostream>`.

## B. Dynamic Memory Allocation:

Coding difference between C and C++

| *C* | *C++* |
|---|---|
| `int *ptr = (int*)malloc(sizeof(int));` | `int *ptr = new int;` |
| `free(ptr);` | `delete ptr;` |
| `int *arr = (int*)malloc(sizeof(int) * 5);` | `int *arr = new int[5];` |
| `free(arr);` | `delete[] arr;` |

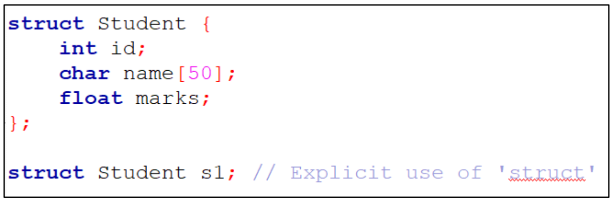
## C. Pointers:

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

int main() {
    int a = 10;
    int *ptr = &a;
    cout << "Value: " << *ptr << endl;
    cout << "Address: " << ptr << endl;
    return 0;
}
```

**Sample Output**

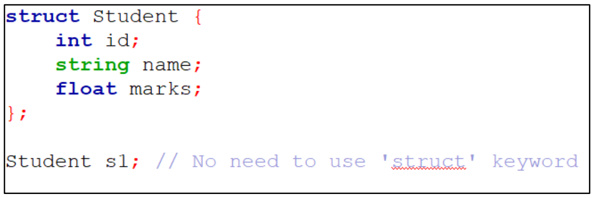
```
Value: 10
Address: 0x6ef13ffc14
```

## D. Structures:



```c
struct Student {
    int id;
    char name[50];
    float marks;
};

struct Student s1; // Explicit use of 'struct'
```

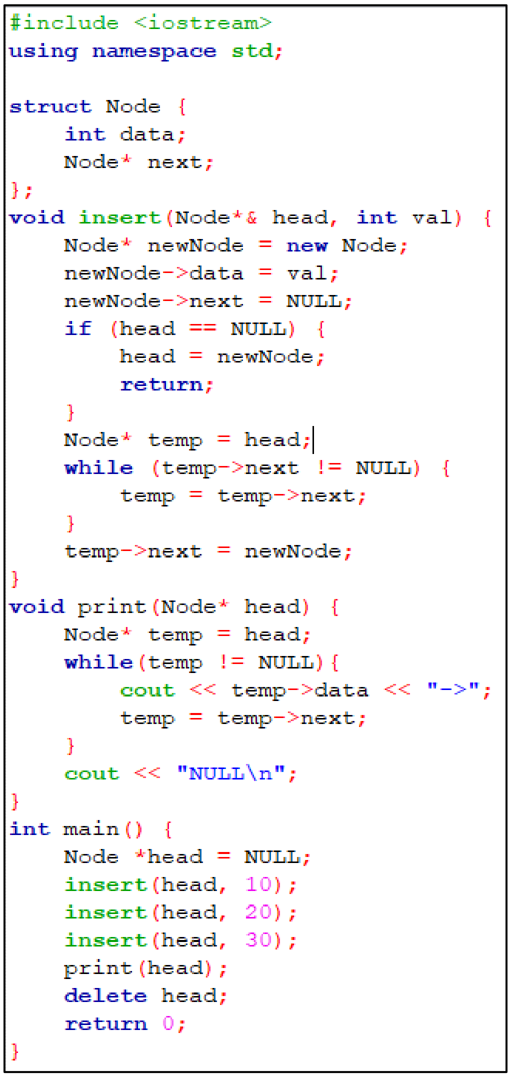
code.c



```cpp
struct Student {
    int id;
    string name;
    float marks;
};

Student s1; // No need to use 'struct' keyword
```

code.cpp

## E. LinkedList using Self-Referenced Structs:



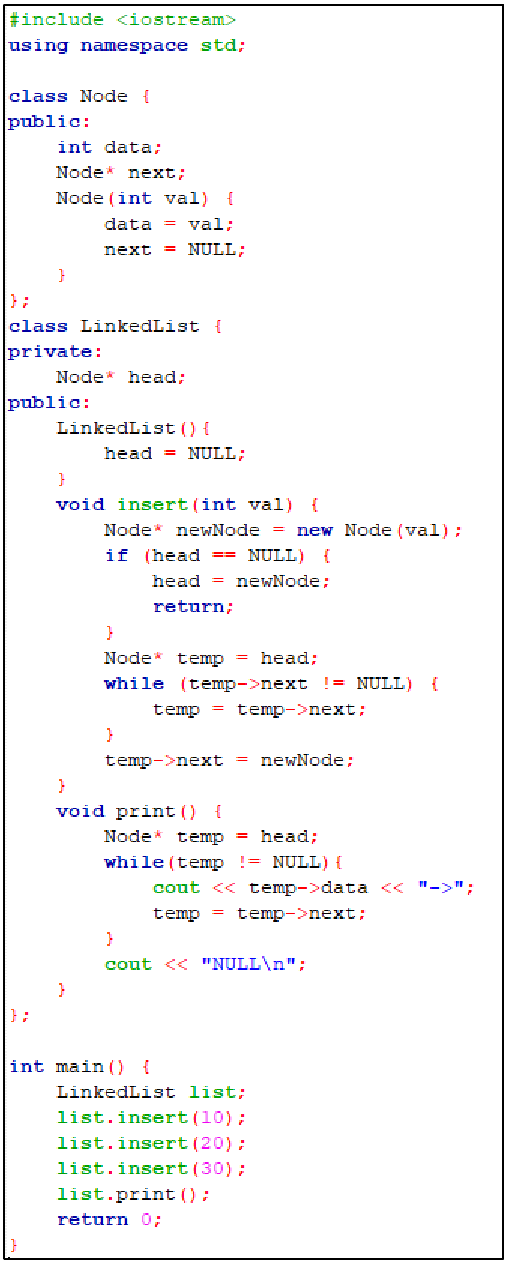
```cpp
#include <iostream>
using namespace std;

struct Node {
    int data;
    Node* next;
};
void insert(Node*& head, int val) {
    Node* newNode = new Node;
    newNode->data = val;
    newNode->next = NULL;
    if (head == NULL) {
        head = newNode;
        return;
    }
    Node* temp = head;
    while (temp->next != NULL) {
        temp = temp->next;
    }
    temp->next = newNode;
}
void print(Node* head) {
    Node* temp = head;
    while(temp != NULL){
        cout << temp->data << "->";
        temp = temp->next;
    }
    cout << "NULL\n";
}
int main() {
    Node *head = NULL;
    insert(head, 10);
    insert(head, 20);
    insert(head, 30);
    print(head);
    delete head;
    return 0;
}
```

**Output**

```
10→20→30→NULL
```

When using dynamic memory allocation for a structure in C++, the `new` keyword is needed. Since the allocated structure is accessed via a pointer, members of the structure should be assigned using → this sign.

## F. LinkedList using Self-Referenced Class:



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

class Node {
public:
    int data;
    Node* next;
    Node(int val) {
        data = val;
        next = NULL;
    }
};
class LinkedList {
private:
    Node* head;
public:
    LinkedList(){
        head = NULL;
    }
    void insert(int val) {
        Node* newNode = new Node(val);
        if (head == NULL) {
            head = newNode;
            return;
        }
        Node* temp = head;
        while (temp->next != NULL) {
            temp = temp->next;
        }
        temp->next = newNode;
    }
    void print() {
        Node* temp = head;
        while(temp != NULL){
            cout << temp->data << "->";
            temp = temp->next;
        }
        cout << "NULL\n";
    }
};

int main() {
    LinkedList list;
    list.insert(10);
    list.insert(20);
    list.insert(30);
    list.print();
    return 0;
}
```

**Output**

```
10→20→30→NULL
```

**Public:** Members declared as `public:` are accessible from anywhere outside the class.

**Private:** Members declared as `private:` are hidden from outside access. They can only be accessed within the class itself.

## Lab Task

1. Reverse an array using only pointers.

   **Sample Input:**

   ```
   5
   3 6 7 6 8
   ```

   **Sample Output:**

   ```
   8 6 7 6 3
   ```

2. Define a `struct` for `Book` with attributes `title`, `author`, and `price`. Take user input to initialize the `Book` array and return the maximum priced `Book` information.

   **Sample Input:**

   ```
   5
   Book1 Author1 3500
   Book2 Author2 5500
   Book3 Author3 4000
   Book4 Author4 7900
   Book5 Author5 500
   ```

   **Sample Output:**

   ```
   Book4 Author4 7900
   ```

3. Implement the linked-list, insert function and print function by taking user input (using both struct and class).

   **Sample Input:**

   ```
   8
   36 21 45 10 5 66 55 44
   ```

   **Sample Output:**

   ```
   36→21→45→10→5→66→55→44→NULL
   ```