Week 9 Lecture 3

NWEN 241
Systems Programming

Jyoti Sahni

jyoti.sahni@ecs.vuw.ac.nz

Defining a class: Example

```
class Time {
public:
        void set(int, int, int);
        void print() const;
        Time();
        Time(int, int, int);

private:
        int hour;
        int minute;
        int second;
};
```

Member access specifiers

Possible specifiers:

- `private`
- `protected`
- `public`

Defining a class: Example

```
class Time {
public:
      void set(int, int, int);
      void print() const;
      Time();
      Time(int, int, int);

private:
      int hour;
      int minute;
      int second;
};
```

Constructors

- Named after class name
- Similar to Java.

When class performs dynamic memory allocation, **destructor** is also needed

Defining Member Functions: Example

- Member functions can be declared in 2 ways:
  - By specifying the function prototype
  - By specifying the function implementation

- Java allows only the second method

```
class Time {
public:
        void print() const;
        void set(int h, int m, int s) {
                hour = h;
                minute = m;
                second = s;
        }
        Time();
        Time(int, int, int);

private:
        int hour;
        int minute;
        int second;
};
```

Implementing Functions Separately: Example

- For member functions that are not implemented in the class declaration, they must be implemented separately

```
class Time {
public:
  void print() const;
  void set(int h, int m, int s) {
    hour = h;
    minute = m;
    second = s;
  }
  …
};
```

```
#include <cstdio>

void Time::print() const
{
  printf("%2d:%2d:%2d", hour,
    minute, second);
}
```

Revisit Structures in the context of C++

Structure in C Vs Structure in C++

- C++ structures adds **extra features** to C structures

- Same declaration syntax

- C++ structures can –
  - have functions as members
  - treated like a *built-in* data type
  - be extended (supports inheritance)
  - define access specifiers (public, private, protected)

Structure in C Vs Structure in C++

- C++ structures adds **extra features** to C structures
- Same declaration syntax
- C++ structures can –
  - have functions as members
  - treated like a *built-in* data type
  - be extended (supports inheritance)
  - define access specifiers (public, private, protected)

```
struct s
{
  int a;
  int b;
  void set()
    {
       a=10;
       b=20; } };
```

Structure in C Vs Structure in C++

- C++ structures adds **extra features** to C structures
- Same declaration syntax
- C++ structures can –
  - have functions as members
  - treated like a *built-in* data type
  - be extended (supports inheritance)
  - define access specifiers (public, private, protected)

```
struct s
{
  int a;
  int b;
  void set()
    {
       a=10;
       b=20; } };

 struct s s1;
 s s1;
```

Structure in C Vs Structure in C++

- C++ structures adds **extra features** to C structures
- Same declaration syntax
- C++ structures can –
  - have functions as members
  - treated like a *built-in* data type
  - be extended (supports inheritance)
  - define access specifiers (public, private, protected)

```
struct base
{
   :
   :
 };

struct derived: base {

}
```

Structure in C Vs Structure in C++

- C++ structures adds **extra features** to C structures

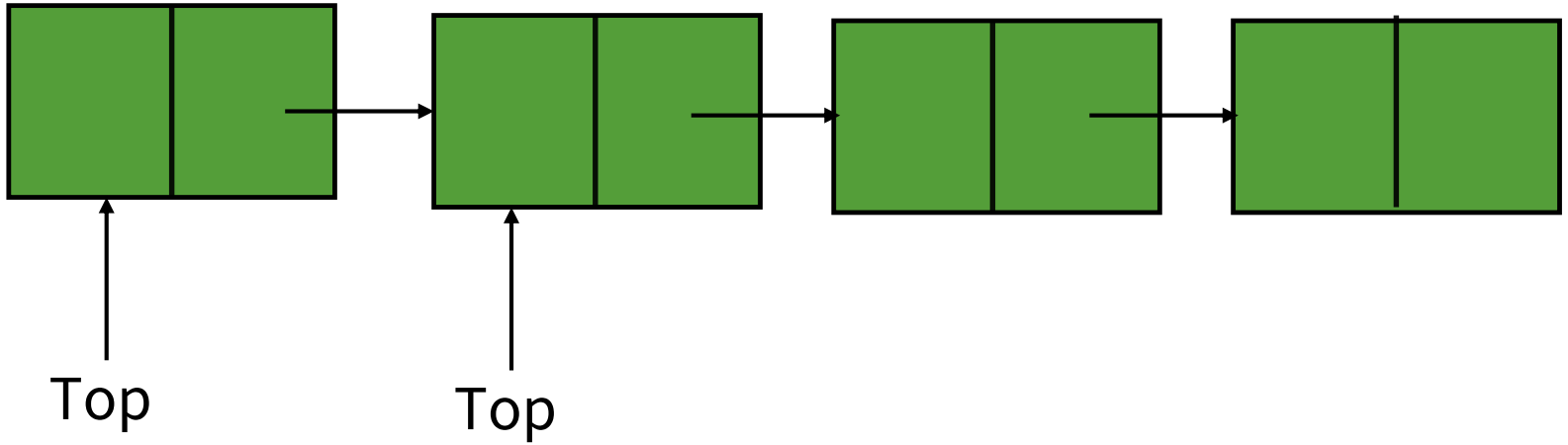
- Same declaration syntax

- C++ structures can –
  - have functions as members
  - treated like a *built-in* data type
  - be extended (supports inheritance)
  - define access specifiers (public, private, protected)

Members are public by default.

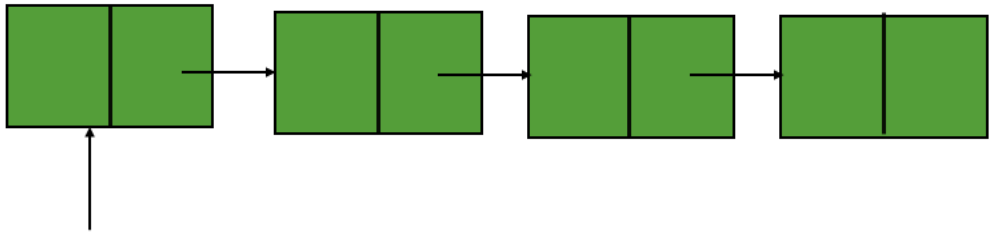
```
struct base
{
  public:
  int a;
  private:
  int b;
  protected:
  int c;
 };

struct derived: base {

}
```

Example: Implementing a Stack



Use a C++ Structure to maintain individual records.
Use a C++ class to maintain the stack

Example: Implementing a Stack

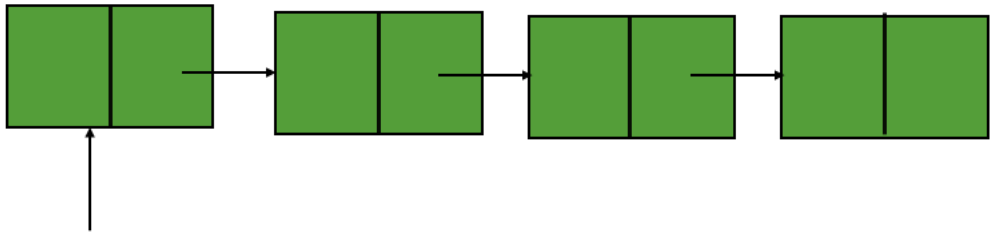


Use a C++ Structure to maintain individual records.
Use a C++ class to maintain the stack

```
struct LinkedNode {
  char value;
  LinkedNode *next;

  LinkedNode(char c, LinkedNode *n){
     value=c;
     next =n;
  }

  LinkedNode(){}
};
```

By default structure members are public

Example: Implementing a Stack



Use a C++ Structure to maintain individual records.
Use a C++ class to maintain the stack

```
struct LinkedNode {
  char value;
  LinkedNode *next;

  LinkedNode(char c, LinkedNode *n){
     value=c;
     next =n;
  }

  LinkedNode(){}
};
```

```
class LinkedStack {
    LinkedNode *top=nullptr;
    int count =0;

   void push(char c);
   char pop();
   void print();
~LinkedStack();       }
```