



Module 12

Partha Pratim
Das

Objectives &
Outline

Access
Specifiers

public and
private

Information
Hiding

Stack (public)
Stack (private)

Get-Set Idiom

Summary

Module 12: Programming in C++

Access Specifiers

Partha Pratim Das

Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

ppd@cse.iitkgp.ernet.in

Tanwi Mallick
Srijoni Majumdar
Himadri B G S Bhuyan



Module Objectives

Module 12

Partha Pratim
Das

Objectives & Outline

Access
Specifiers

public and
private

Information
Hiding

Stack (public)
Stack (private)

Get-Set Idiom

Summary

- Understand access specifiers in C++ classes to control the visibility of members
- Learn to design with Information Hiding



Module Outline

Module 12

Partha Pratim
Das

Objectives &
Outline

Access
Specifiers

public and
private

Information
Hiding

Stack (public)
Stack (private)

Get-Set Idiom

Summary

- Access specifiers
 - public Access Specifier
 - private Access Specifier
- Information Hiding
 - Stack with public data
 - Stack with private data
- Get-Set Idiom



Module 12: Lecture 21

Module 12

Partha Pratim
Das

Objectives & Outline

Access
Specifiers

public and
private

Information
Hiding

Stack (public)

Stack (private)

Get-Set Idiom

Summary

- Access specifiers
 - public Access Specifier
 - private Access Specifier
- Information Hiding
 - Stack with public data
 - Stack with private data



Program 12.01/02: Complex Number: Access Specification

Module 12

Partha Pratim
Das

Objectives &
Outline

Access
Specifiers

public and
private

Information
Hiding

Stack (public)
Stack (private)

Get-Set Idiom

Summary

Public data, Public method

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

class Complex { public: double re, im;
public: double norm() {
    return sqrt(re*re + im*im);
}
};

void print(const Complex& t) { // Global fn.
    cout << t.re << "+j" << t.im << endl;
}

int main() {
    Complex c = { 4.2, 5.3 }; // Okay

    print(c);
    cout << c.norm();
    return 0;
}
```

- public data can be accessed by any function
- norm (method) can access (re, im)
- print (global) can access (re, im)
- main (global) can access (re, im) & initialize

Private data, Public method

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

class Complex { private: double re, im;
public: double norm() {
    return sqrt(re*re + im*im);
}
};

void print(const Complex& t) { // Global fn.
    cout << t.re << "+j" << t.im << endl;
    // 'Complex::re': cannot access private
    // member declared in class 'Complex'

    // 'Complex::im': cannot access private
    // member declared in class 'Complex'
}

int main() {
    Complex c = { 4.2, 5.3 }; // Error
    // 'initializing': cannot convert from
    // 'initializer-list' to 'Complex'
    print(c);
    cout << c.norm();
    return 0;
}
```

- private data can be accessed *only* by methods
- norm (method) can access (re, im)
- print (global) cannot access (re, im)
- main (global) cannot access (re, im) to initialize



Access Specifiers

Module 12

Partha Pratim
Das

Objectives &
Outline

Access
Specifiers

public and
private

Information
Hiding

Stack (public)
Stack (private)

Get-Set Idiom

Summary

- Classes provide **access specifiers** for members (data as well as function) to enforce **data hiding** that separates **implementation** from **interface**
 - **private** - accessible inside the definition of the class
 - member functions of the same class
 - **public** - accessible everywhere
 - member functions of the same class
 - member function of a different class
 - global functions
- The keywords **public** and **private** are the *Access Specifiers*
- Unless specified, the access of the members of a class is considered **private**
- A class may have multiple access specifier. The effect of one continues till the next is encountered



Information Hiding

Module 12

Partha Pratim Das

Objectives & Outline

Access Specifiers

public and private

Information Hiding

Stack (public)

Stack (private)

Get-Set Idiom

Summary

- The private part of a class (*attributes* and *methods*) forms its **implementation** because the class alone should be concerned with it and have the right to change it
- The public part of a class (*attributes* and *methods*) constitutes its **interface** which is available to all others for using the class
- Customarily, we put all *attributes* in private part and the *methods* in public part. This ensures:
 - The **state** of an object can be changed only through one of its *methods* (with the knowledge of the class)
 - The **behavior** of an object is accessible to others through the *methods*
- This is known as **Information Hiding**



Information Hiding

Module 12

Partha Pratim
Das

Objectives &
Outline

Access
Specifiers

public and
private

Information
Hiding

Stack (public)
Stack (private)

Get-Set Idiom

Summary

- For the sake of efficiency in design, we at times, put *attributes* in public and / or *methods* in private. In such cases:
 - The public *attributes should not* decide the *state* of an object, and
 - The private *methods* cannot be part of the behavior of an object

We illustrate information hiding through two implementations a stack



Program 12.03/04: Stack: Implementations using public data

Module 12

Partha Pratim Das

Objectives & Outline

Access Specifiers

public and private

Information

Hiding

Stack (public)

Stack (private)

Get-Set Idiom

Summary

Using dynamic array

```
#include <iostream> #include <cstdlib>
using namespace std;
class Stack { public:
    char *data_; int top_;
public:
    int empty() { return (top_ == -1); }
    void push(char x) {data_[++top_] = x; }
    void pop() { --top_; }
    char top() { return data_[top_]; }
};
int main() {
    Stack s; char str[10] = "ABCDE";

    s.data_ = new char[100]; // Exposed Init
    s.top_ = - 1;             // Exposed Init

    for(int i = 0; i < 5; ++i)
        s.push(str[i]);
    while (!s.empty()) {
        cout << s.top(); s.pop();
    } // Outputs: EDCBA -- Reversed string
    delete [] s.data_;      // Exposed De-Init
    return 0;
}
```

Using vector

```
#include <iostream> #include <vector>
using namespace std;
class Stack { public:
    vector<char> data_; int top_;
public:
    int empty() { return (top_ == -1); }
    void push(char x) { data_[++top_] = x; }
    void pop() { --top_; }
    char top() { return data_[top_]; }
};
int main() {
    Stack s; char str[10] = "ABCDE";

    s.data_.resize(100); // Exposed Init
    s.top_ = -1;         // Exposed Init

    for(int i = 0; i < 5; ++i)
        s.push(str[i]);
    while (!s.empty()) {
        cout << s.top(); s.pop();
    } // Outputs: EDCBA -- Reversed string

    return 0;
}
```

- public data reveals the *internals* of the stack (no information hiding)
- Spills data structure codes (Exposed Init / De-Init) into the application (main)
- To switch from array to vector or vice-versa the application needs to change



Program 12.03/04: Stack: Implementations using public data – Risks

Module 12

Partha Pratim
Das

Objectives &
Outline

Access
Specifiers

public and
private

Information
Hiding

Stack (public)
Stack (private)

Get-Set Idiom

Summary

Using dynamic array

```
#include <iostream> #include <cstdlib>
using namespace std;
class Stack { public:
    char *data_; int top_;
public:
    int empty() { return (top_ == -1); }
    void push(char x) {data_[++top_] = x; }
    void pop() { --top_; }
    char top() { return data_[top_]; }
};
int main() {
    Stack s; char str[10] = "ABCDE";

    s.data_ = new char[100]; // Exposed Init
    s.top_ = - 1;             // Exposed Init

    for(int i=0; i<5; ++i) s.push(str[i]);

    s.top_ = 2; // STACK GETS INCONSISTENT

    while (!s.empty()) {
        cout << s.top(); s.pop();
    } // Outputs: CBA -- WRONG!!!
    delete [] s.data_;      // Exposed De-Init
    return 0;
}
```

Using vector

```
#include <iostream> #include <vector>
using namespace std;
class Stack { public:
    vector<char> data_; int top_;
public:
    int empty() { return (top_ == -1); }
    void push(char x) { data_[++top_] = x; }
    void pop() { --top_; }
    char top() { return data_[top_]; }
};
int main() {
    Stack s; char str[10] = "ABCDE";

    s.data_.resize(100); // Exposed Init
    s.top_ = -1;         // Exposed Init

    for(int i=0; i<5; ++i) s.push(str[i]);

    s.top_ = 2; // STACK GETS INCONSISTENT

    while (!s.empty()) {
        cout << s.top(); s.pop();
    } // Outputs: CBA -- WRONG!!!

    return 0;
}
```

- Application may intentionally or inadvertently tamper the value of `top_` – this corrupts the stack!
- `s.top_ = 2`; destroys consistency of the stack and causes wrong output



Program 12.05/06: Stack: Implementations using private data – Safe

Module 12

Partha Pratim
Das

Objectives &
Outline

Access
Specifiers

public and
private

Information
Hiding

Stack (public)
Stack (private)

Get-Set Idiom

Summary

Using dynamic array

```
#include <iostream>

using namespace std;
class Stack { private:
    char *data_; int top_;
public:
    // Initialization
    Stack(): data_(new char[100]), top_(-1) {}
    // De-Initialization
    ~Stack() { delete[] data_; }
    int empty() { return (top_ == -1); }
    void push(char x) { data_[++top_] = x; }
    void pop() { --top_; }
    char top() { return data_[top_]; }
};

int main() {
    Stack s; char str[10] = "ABCDE";
    for (int i=0; i<5; ++i) s.push(str[i]);
    while (!s.empty()) {
        cout << s.top(); s.pop();
    }
    return 0;
}
```

Using vector

```
#include <iostream>
#include <vector>
using namespace std;
class Stack { private:
    vector<char> data_; int top_;
public:
    // Initialization
    Stack(): top_(-1) { data_.resize(100); }
    // De-Initialization
    ~Stack() {};
    int empty() { return (top_ == -1); }
    void push(char x) { data_[++top_] = x; }
    void pop() { --top_; }
    char top() { return data_[top_]; }
};

int main() {
    Stack s; char str[10] = "ABCDE";
    for (int i=0; i<5; ++i) s.push(str[i]);
    while (!s.empty()) {
        cout << s.top(); s.pop();
    }
    return 0;
}
```

- private data hides the *internals* of the stack (information hiding)
- Data structure codes contained within itself with initialization and de-initialization
- To switch from array to vector or vice-versa the application needs *no* change
- **Application cannot tamper stack – any direct access to `top_` or `data_` is compilation error!**



Interface and Implementation

Module 12

Partha Pratim Das

Objectives & Outline

Access Specifiers

public and private

Information Hiding

Stack (public)

Stack (private)

Get-Set Idiom

Summary

Interface

```
// File: Stack.h
class Stack { private: // Implementation
    char *data_; int top_;
public: // Interface
    Stack();
    ~Stack();
    int empty();
    void push(char x);
    void pop();
    char top();
};
```

Implementation

```
// File: Stack.h
class Stack { private: // Implementation
    char *data_; int top_;
public: // Interface
    Stack();
    ~Stack();
    int empty();
    void push(char x);
    void pop();
    char top();
};

// File: Stack.cpp // Implementation
Stack::Stack(): data_(new char[100]), top_(-1) {}
Stack::~~Stack() { delete[] data_; }
int Stack::empty() { return (top_ == -1); }
void Stack::push(char x) { data_[++top_] = x; }
void Stack::pop() { --top_; }
char Stack::top() { return data_[top_]; }
```

Application

```
#include "Stack.h"
int main() {
    Stack s; char str[10] = "ABCDE";
    for (int i = 0; i < 5; ++i) s.push(str[i]);
    while (!s.empty()) { cout << s.top(); s.pop(); }
    return 0;
}
```



Module 12: End of Lecture 21

Module 12

Partha Pratim
Das

Objectives &
Outline

Access
Specifiers

public and
private

Information
Hiding

Stack (public)
Stack (private)

Get-Set Idiom

Summary

- Access specifiers
 - public Access Specifier
 - private Access Specifier
- Information Hiding
 - Stack with public data
 - Stack with private data



Module 12: Lecture 22

Module 12

Partha Pratim
Das

Objectives &
Outline

Access
Specifiers

public and
private

Information
Hiding

Stack (public)
Stack (private)

Get-Set Idiom

Summary

- Get-Set Idiom
 - Idiom for fine-grained Access Control



Get-Set Methods: Idiom for fine-grained Access Control

Module 12

Partha Pratim
Das

Objectives &
Outline

Access
Specifiers

public and
private

Information
Hiding

Stack (public)
Stack (private)

Get-Set Idiom

Summary

- As noted, we put all *attributes* in private and the *methods* in public. This restricts the access to data completely
- To fine-grain the access to data we provide selective public member functions to *read* (get) and / or *write* (set) data

```
class MyClass { // private
    int readWrite_; // Like re_, im_ in Complex -- common aggregated members

    int readOnly_; // Like DateOfBirth, Emp_ID, RollNo -- should not need a change

    int writeOnly_; // Like Password -- reset if forgotten

    int invisible_; // Like top_, data_ in Stack -- keeps internal state

public:
    // get and set methods both to read as well as write readWrite_ member
    int getReadWrite() { return readWrite_; }
    void setReadWrite(int v) { readWrite_ = v; }

    // Only get method to read readOnly_ member - no way to write it
    int getReadOnly() { return readOnly_; }

    // Only set method to write writeOnly_ member - no way to read it
    void setWriteOnly(int v) { writeOnly_ = v; }

    // No method accessing invisible_ member directly - no way to read or write it
}
```



Module Summary

Module 12

Partha Pratim
Das

Objectives &
Outline

Access
Specifiers

public and
private

Information
Hiding

Stack (public)
Stack (private)

Get-Set Idiom

Summary

- Access Specifiers helps to control visibility of data members and methods of a class
- The private access specifier can be used to hide information about the implementation details of the data members and methods
- Get, Set methods are defined to provide an interface to use and access the data members



Instructor and TAs

Module 12

Partha Pratim
Das

Objectives &
Outline

Access
Specifiers

public and
private

Information
Hiding

Stack (public)
Stack (private)

Get-Set Idiom

Summary

Name	Mail	Mobile
Partha Pratim Das, <i>Instructor</i>	ppd@cse.iitkgp.ernet.in	9830030880
Tanwi Mallick, <i>TA</i>	tanwimallick@gmail.com	9674277774
Srijoni Majumdar, <i>TA</i>	majumdarsrijoni@gmail.com	9674474267
Himadri B G S Bhuyan, <i>TA</i>	himadribhuyan@gmail.com	9438911655