Class and Object

Module 2

OOP (IT 2005)

4th Semester ECSc

Introduction

- Class forms the building blocks of the object-oriented programming paradigm.
- A class is the basic mechanism to provide data encapsulation.
- Data encapsulation is an important feature of object-oriented programming paradigm.
- It binds data and member functions in a single entity in such a way that data can be manipulated only through the function defined in that class.

Class Declaration

```
class class_name
{
   private:
     data declarations;
     function declarations;
   public:
     data declarations;
     function declarations
 };
```

Class Declaration

```
class class_name
{
  private:
    data declarations;
    function declarations;
  public:
    data declarations;
    function declarations
};
```

All data members and member functions declared private can be accessed only from within the class.

Class Declaration

```
class class_name
{
  private:
    data declarations;
    function declarations;
  public:
    data declarations;
    function declarations
};
```

All data members and member functions declared public can be accessed from outside the class.

Class Declaration

```
class class_name
{
  private:
    data declarations;
    function declarations;
  public:
    data declarations;
    function declarations
};
```

By default, members of the class, both data and function, are private.

Object

- To use a class, we must create variables of the class known as object.
- The process of creating objects of the class is called class instantiation.
- Memory is allocated only when we create object(s) of the class.
- Syntax of defining an object of a class:

```
class_name object_name;
```

Program Example 1

- Write a C++ program to calculate an area of a rectangle.

```
#include <iostream>
using namespace std;
class rectangle
{
    private:
        float length;
        float breadth;
    public:
        void get_data();
        float area();
        void show_data();
};
```

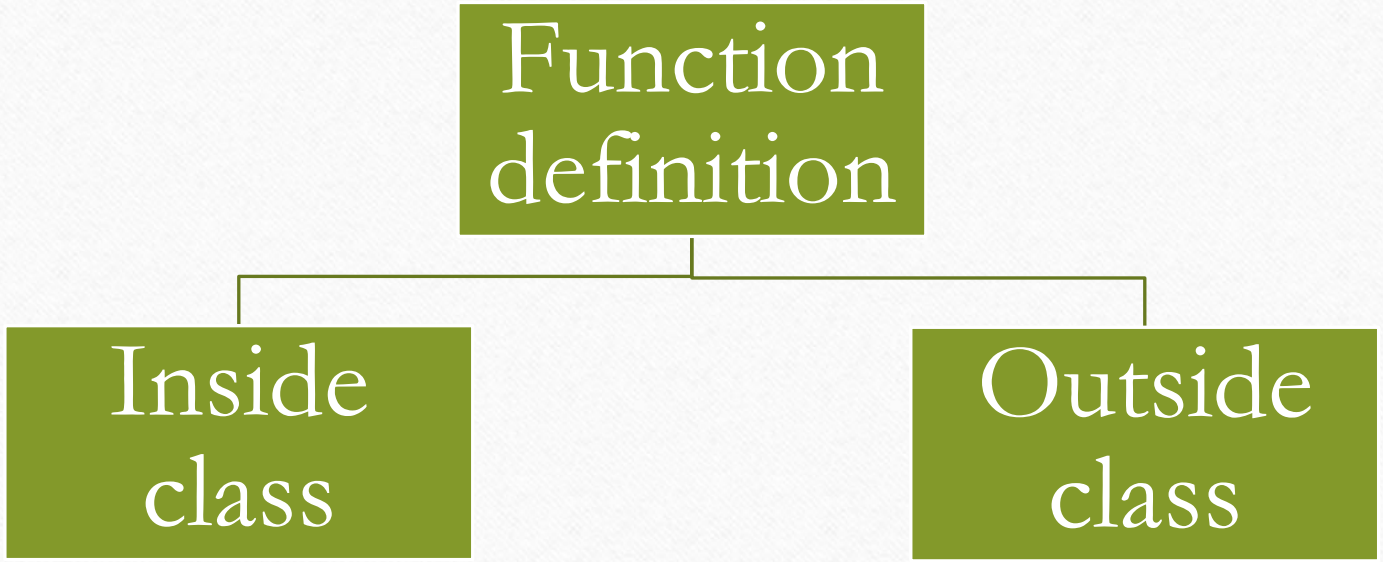
Function Definition Outside Class

```
void rectangle::get_data()
{
    cin >> length >> breadth;
}
float rectangle::area()
{
    return (length * breadth);
}
void rectangle::show_data()
{
    cout << "Length: " << length << endl;
    cout << "Breadth: " << breadth << endl;
}
```

Object Creation

```
int main()
{
    rectangle R; // object of class rectangle
    cout << "Enter the length and breadth: ";
    R.get_data();
    R.show_data();
    cout << "Area: " << R.area() << endl;
    return 0;
}
```

Function Definition Inside Class vs. Outside Class



- Fast code execution
- Poor code readability

- Better code readability and clarity
- Slow execution of code

Program Example 1

- Write a C++ program to calculate an area of a rectangle.

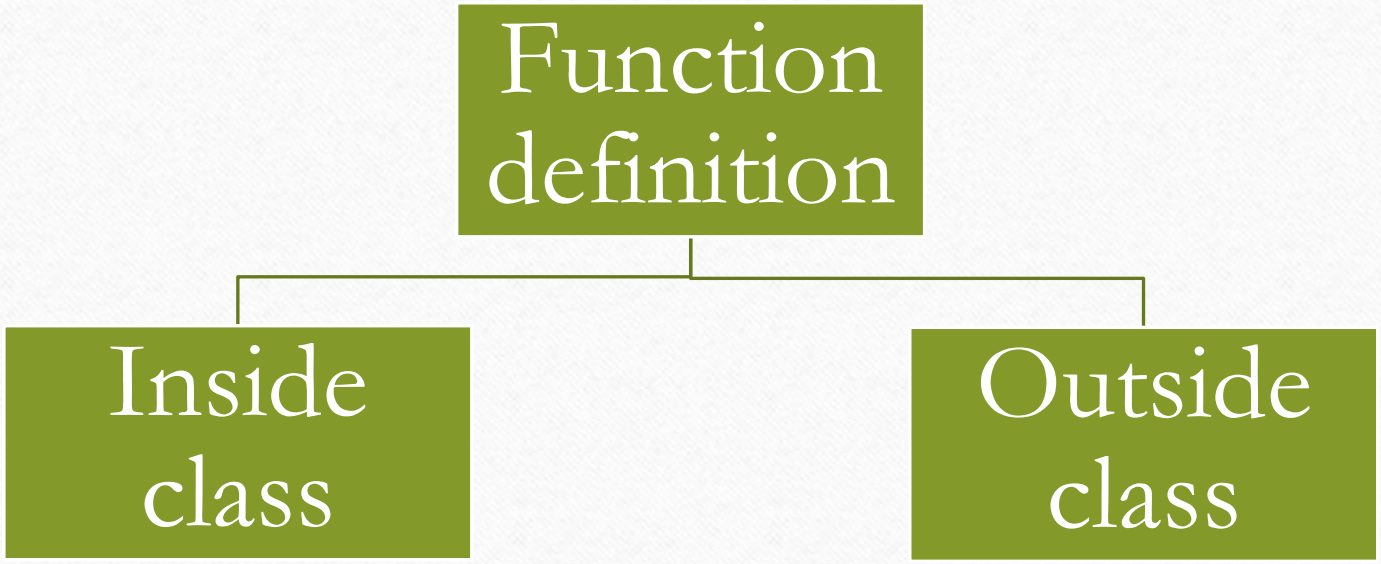
```
#include <iostream>
using namespace std;
class rectangle
{
    private:
        float length;
        float breadth;
    public:
        void get_data() { cin >> length >> breadth; }
        float area() { return (length * breadth); }
        void show_data() { cout << "Length: " << length << endl;
    cout << "Breadth: " << breadth << endl; } };
```

Function definition inside the class

Object Creation

```
int main()
{
    rectangle R; // object of class rectangle
    cout << "Enter the length and breadth: ";
    R.get_data();
    R.show_data();
    cout << "Area: " << R.area() << endl;
    return 0;
}
```

Function Definition Inside Class vs. Outside Class



- Fast code execution
- Poor code readability

- Better code readability and clarity
- Slow execution of code

How to make it fast?

Making a Member Function Inline

- For fast execution of functions, C++ enables programmers to make a member function defined outside the class an inline.
- This can be done by using the keyword **inline** in the header line of function definition.
- See Program Example 2

Program Example 2

- Write a C++ program to calculate an area of a rectangle.

```
#include <iostream>
using namespace std;
class rectangle
{
    private:
        float length;
        float breadth;
    public:
        void get_data();
        float area();
        void show_data();
};
```

Inline Function Definition Outside Class

```
inline void rectangle::get_data()
{
    cin >> length >> breadth;
}
inline float rectangle::area()
{
    return (length * breadth);
}
inline void rectangle::show_data()
{
    cout << "Length: " << length << endl;
    cout << "Breadth: " << breadth << endl;
}
```

Object Creation

```
int main()
{
    rectangle R; // object of class rectangle
    cout << "Enter the length and breadth: ";
    R.get_data();
    R.show_data();
    cout << "Area: " << R.area() << endl;
    return 0;
}
```

Program Example 3

- A class has 3 students. Each student must read 3 subjects. Write a C++ program to read the marks obtained by each student in each subject. Also calculate the total marks obtained by each student. Apply the concept of class and object.

Student ID	Subject 1 (100)	Subject 2 (100)	Subject 3 (100)
Student 1	75	80	55
Student 2	80	75	63
Student 3	70	64	42

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

class student
{
    private:
        char name[5];
        int roll;
        float sub1;
        float sub2;
        float sub3;
    public:
        void get_data();
        float total();
};
```

```
inline void student::get_data()
{
    cout << "Enter the name: ";
    cin >> name;
    cout << "Enter roll number: ";
    cin >> roll;
    cout << "Enter the marks of three subjects: ";
    cin >> sub1 >> sub2 >> sub3;
}
```

```
inline float student::total()
{
    return (sub1 + sub2 + sub3);
}
```

```
int main()
{
student S[3]; // array of objects of class student

for (int i = 0; i < 3; i++)
    S[i].get_data();

for (int i = 0; i < 3; i++)
    cout << "Total marks of student " << (i + 1) << ": " << S[i].total() <<
endl;

return 0;
}
```

Program Example 4

- A class has 3 students. Each student must read 3 subjects. Write a C++ program to read the marks obtained by each student in each subject. Also calculate the total marks obtained by each student and average marks obtained in each subject. Apply the concept of class and object.

Student ID	Subject 1 (100)	Subject 2 (100)	Subject 3 (100)
Student 1	75	80	55
Student 2	80	75	63
Student 3	70	64	42

```
#include <iostream>
using namespace std;
float avg(float x, float y, float z);
class student
{
    private:
        char name[5];
        int roll;
        float sub1;
        float sub2;
        float sub3;
    public:
        void get_data();
        float total();
};
```

Global function to calculate average in each subject

```
inline void student::get_data()
{
    cout << "Enter the name: ";
    cin >> name;
    cout << "Enter roll number: ";
    cin >> roll;
    cout << "Enter the marks of three subjects: ";
    cin >> sub1 >> sub2 >> sub3;
}

inline float student::total()
{
    return (sub1 + sub2 + sub3);
}
```

```
float avg(float x, float y, float z)
{
    return ((x + y + z) / 3.0);
}
int main()
{
student S[3]; // object of class student
for (int i = 0; i < 3; i++)
    S[i].get_data();
for (int i = 0; i < 3; i++)
    cout << "Total marks of student " << (i + 1) << ": " << S[i].total() <<
endl;
cout << "Average marks in subject 1: " << avg(S[0].sub1, S[1].sub1,
S[2].sub1) << endl;
cout << "Average marks in subject 2: " << avg(S[0].sub2, S[1].sub2,
S[2].sub2) << endl;
cout << "Average marks in subject 3: " << avg(S[0].sub3, S[1].sub3,
S[2].sub3) << endl;
return 0;
}
```

```
float avg(float x, float y, float z)
{
    return ((x + y + z) / 3.0);
}
int main()
{
student S[3]; // object of class student
for (int i = 0; i < 3; i++)
    S[i].get_data();
for (int i = 0; i < 3; i++)
    cout << "Total marks of student " << (i + 1) << ": " << S[i].total() <<
endl;
cout << "Average marks in subject 1: " << avg(S[0].sub1, S[1].sub1,
S[2].sub1) << endl;
cout << "Average marks in subject 2: " << avg(S[0].sub2, S[1].sub2,
S[2].sub2) << endl;
cout << "Average marks in subject 3: " << avg(S[0].sub3, S[1].sub3,
S[2].sub3) << endl;
return 0;
}
```

Illegal access and usage

Friend Function

- A non-member function cannot access an object's private data.
- A friend function of a class is a non-member function of the class that can access its private members.
- We use a keyword **friend** to declare a function friend of a class.

Important Notes on Friend Function

- Friend function is a normal external function that is given special access privileges.
- The declaration of the friend function is included in the class definition.
- The declaration can be done either in the private or in the public section.
- A friend function must be defined outside the class.
- The keyword **friend** is used during the function declaration.
- Do not use the keyword **friend** during the function definition.

Program Example 4

- A class has 3 students. Each student must read 3 subjects. Write a C++ program to read the marks obtained by each student in each subject. Also calculate the total marks obtained by each student and average marks obtained in each subject. Apply the concept of class and object.

Student ID	Subject 1 (100)	Subject 2 (100)	Subject 3 (100)
Student 1	75	80	55
Student 2	80	75	63
Student 3	70	64	42

```
#include <iostream>
using namespace std;
class student
{
    private:
        char name[5];
        int roll;
        float sub1;
        float sub2;
        float sub3;
        friend void avg(student &S1, student &S2, student &S3);
    public:
        void get_data();
        float total();
};
```

```
inline void student::get_data()
{
    cout << "Enter the name: ";
    cin >> name;
    cout << "Enter roll number: ";
    cin >> roll;
    cout << "Enter the marks of three subjects: ";
    cin >> sub1 >> sub2 >> sub3;
}
```

```
inline float student::total()
{
    return (sub1 + sub2 + sub3);
}
```

```
void avg(student& S1, student& S2, student& S3)
{
    cout << "Average marks in subject 1: " << ((S1.sub1 + S2.sub1 + S3.sub1) /
3.0) << endl;
    cout << "Average marks in subject 2: " << ((S1.sub2 + S2.sub2 + S3.sub2) /
3.0) << endl;
    cout << "Average marks in subject 3: " << ((S1.sub3 + S2.sub3 + S3.sub3) /
3.0) << endl;
}
```

```
int main()
{
student S[3]; // object of class student
for (int i = 0; i < 3; i++)
    S[i].get_data();

for (int i = 0; i < 3; i++)
    cout << "Total marks of student " << (i + 1) << ": " << S[i].total() <<
endl;

    avg(S[0], S[1], S[2]);
    return 0;
}
```

Notice the function call

Program Example 5

- Write a class called Complex that has data members real and imaginary. Using the class Complex, create two objects and then add the objects. Display both the complex numbers and the result of addition.

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

class complex
{
    private:
        int real;
        float imag;
    public:
        void get_data();
        void display_data();
        void add_num(complex &c1, complex &c2);
};
```

```
inline void complex::get_data()
{
    cout << "Enter the real and imaginary part: ";
    cin >> real >> imag;
}

inline void complex::add_num(complex& c1, complex& c2)
{
    real = c1.real + c2.real;
    imag = c1.imag + c2.imag;
    display_data();
}

inline void complex::display_data()
{
    cout << "The real part is: " << real << endl;
    cout << "The imaginary part is: " << imag << endl;
}
```

```
int main()
{
    complex num1, num2, num3;
    cout << "Enter the first number..." << endl;
    num1.get_data();
    cout << "Enter the second number..." << endl;
    num2.get_data();

    num3.add_num(num1, num2);
    return 0;
}
```