CIS 129
Advanced Computer Programming

Chapter 8: User Defined Datatypes: Header and Class

Mr. Horence Chan

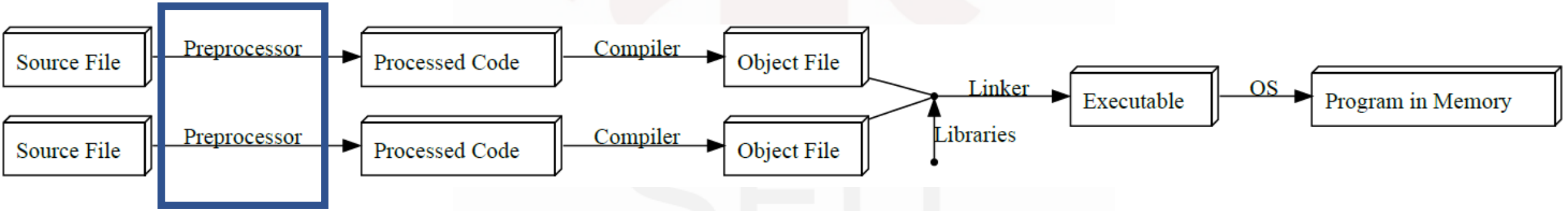
Header File

- The C++ Standard Library offers its users a variety of functions, one of which is header files.
- In C++, all the header files may or may not end with the **.h extension**.
- A header file in C++ contains:
  - Function definitions
  - Data type definitions
  - Macros

Header File

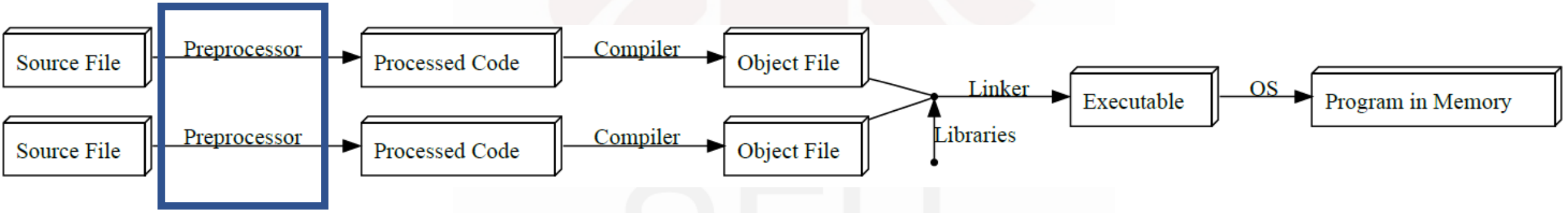
- Header files offer these features by importing them into your program with the help of a preprocessor directive called **`#include`**.

- These preprocessor directives are responsible for instructing the C/C++ compiler that these files need to be processed before compilation.



Header Files

- Instead of writing a large and complex code, you can create your own header files and include it in the C++ library to use it whenever you wish as frequently as you like. It enhances code functionality and readability.



Header File

- Basically, header files are of 2 types:

1. _____ **header files:** These are the pre-existing header files already available in the C++ compiler.

2. _____ **header files:** Header files designed by the user.

Standard library header files

- Examples of standard library header files:

1. `#inlcude<iostream>` (Input Output Stream) – Used as a stream of input and output.

   Examples: _____

2. `#include<iomanip>` (Input-Output Manipulation) – Used to manipulating the output

   Examples: _____

3. `#include<fstream>` (File stream) – Used to control the data to read from a file as an input and data to write into the file as an output.

   Examples: _____

Syntax of Header File

- **Standard library header files:**
  - `#include<filename>`
  - The name of the header file is enclosed within angular brackets <>

- **User-defined header files:**
  - `#include "_____"`
  - The name of the header file is enclosed within double quotes ""
  - `___`: Extension of header file

- Note: do not include the same header file in the same program twice

User-defined Header Files

```
int SumNNumbers (int number){
    int sum=0;
    for(int i=1; i<=number; i++){
        sum+=i;
    }
    return sum;
}
```

- For example, we want to find the sum of first n positive numbers.
- Since it not pre-defined in the standard C++ library, we can create a header file and named it "_____"

User-defined Header Files

```
#include <iostream>
#include "_____________"
using namespace std;
int main(){
int num;
cout << "Enter the number of positive"
      << " integers to be added: ";
cin >> num;
cout <<endl;
cout << "The sum of the first " << num
     << " positive integers is "
     <<_____________________(num) <<
endl;
}
```

- In the source file (e.g. main.cpp), find the sum of first n positive numbers using a self-created header file
- Sample Output:

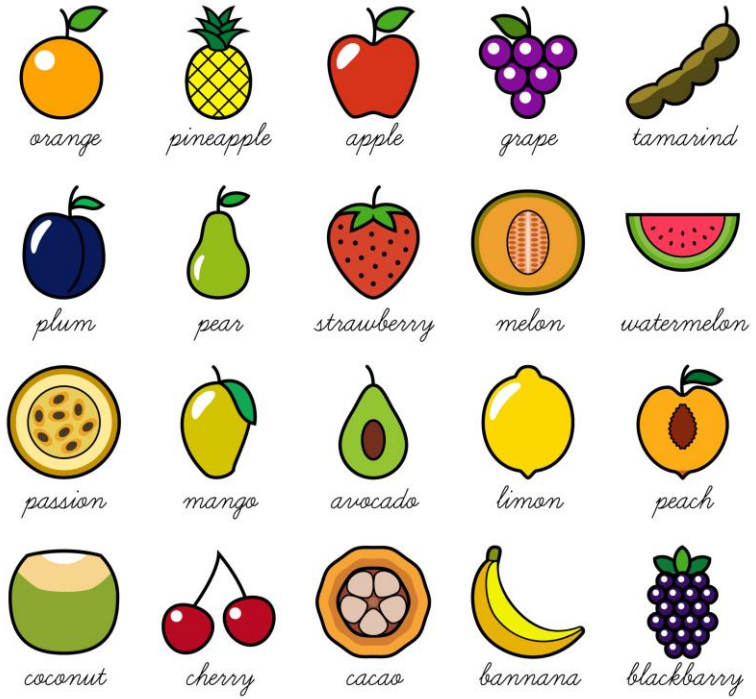
Enter the number of positive integers to be added: 99

The sum of the first 99 positive integers is 4950

Class and Object

- For example,

Class	Objects
Fruit	Apple, Banana, Mango
Vehicle	Car, Truck, Bus



- A class is a template for objects, and an object is an instance of a class.
- When the individual objects are created, they inherit all the variables and functions from the class.

Class

- Classes have **Visibility Modes** (Access Specifiers) for the data inside them.
- The three main visibility modes of members of a class are:
- _____: Members are **accessible** from outside the class.
- _____: Members **cannot** be accessed (or viewed) from outside the class
- _____: Members **cannot** be accessed from outside the class, however, they can be accessed in **inherited** classes.

- Note: Members of a class are _____ by default

Class: Public

```
class Student{
   public:
        string Name;
        char Grade;
        int Mark;
};
int main(){
Student sudent1;
sudent1.Name = "Peter Parker";
sudent1.Grade = 'B';
sudent1.Mark = 80;
Student sudent2 ={"Mary Jane", 'A', 90};
cout<<"Student information: "<<endl;
cout<< sudent1.Name <<" "<< sudent1.Grade <<" "<< sudent1.Mark <<endl;
cout<< sudent2.Name <<" "<< sudent2.Grade <<" "<< sudent2.Mark <<endl;
return 0;
}
```

- `Name`, `Grade` and `Mark` (data inside the class) are known as **data members/ attributes** of the class.
- `sudent1`, and `sudent2` (which were earlier structure type variables) are termed as **objects**.

Output

Student information:

Peter Parker B 80

Mary Jane A 90

Class: Public

```
class Student{
   public:
        string Name;
        char Grade;
        int Mark;
};
int main(){
Student sudent1;
sudent1.Name = "Peter Parker";
sudent1.Grade = 'B';
sudent1.Mark = 80;
Student sudent2 ={"Mary Jane", 'A', 90};
cout<<"Student information: "<<endl;
cout<< sudent1.Name <<" "<< sudent1.Grade <<" "<< sudent1.Mark <<endl;
cout<< sudent2.Name <<" "<< sudent2.Grade <<" "<< sudent2.Mark <<endl;
return 0;
}
```

Class: Student	student1	student2
Name	Peter Parker	Mary Jane
Grade		
Mark		

Class: Private

```
class Student{

        string Name;
        char Grade;
        int Mark;
};
int main(){
Student sudent1;
sudent1.Name = "Peter Parker";
sudent1.Grade = 'B';
sudent1.Mark = 80;
Student sudent2 ={"Mary Jane", 'A', 90};
cout<<"Student information: "<<endl;
cout<< sudent1.Name <<" "<< sudent1.Grade <<" "<< sudent1.Mark <<endl;
cout<< sudent2.Name <<" "<< sudent2.Grade <<" "<< sudent2.Mark <<endl;
return 0;
}
```

- Members of a class are _____ by default
- Members _____ be accessed (or viewed) from outside the class

Class: Private

```
class Student{
        string Name;
        char Grade;
        int Mark;
    public:
        void Input(){
        getline(cin, Name);
        cin >> Grade >> Mark;
        }
        void Output(){
        cout<< Name <<" "<< Grade<<" " << Mark;}
        };
int main(){
        Student sudent1;
        sudent1.Input();
        sudent1.Output();
        return 0;}
```

- To access private members outside the class, we may define some function in public and access the function in `main()`
- Classes have the ability to have functions inside them to manipulate the data.
- These are known as _____.
- These can be defined in public mode so that objects can directly access them.
- Note: When we say 'members' of a class, we are referring to both data members and member functions.

Sample Output:

Peter Parker

B 80

Peter Parker B 80

Class: Function

```
class Student{
        string Name; char Grade; int Mark;
        public:
        void Input();
        void Output();
};
int main(){
        Student sudent1;
        sudent1.Input();
        sudent1.Output();
        return 0;
}
void Student::Input(){
        getline(cin, Name);
        cin >> Grade >> Mark;
}
void Student::Output(){
         cout<< Name <<" "<< Grade<<" " << Mark;
}
```

- Member functions is usually preferable to define them outside the class, especially if they have control structures such as conditional constructs and loops.
- This is because some compilers tend to give errors when functions containing these are defined within the classes.
- To define functions outside the classes, we have the function prototypes within the classes, and the definition outside.

Sample Output:

Peter Parker

B 80

Peter Parker B 80

Class: Function

```
class Student{
        string Name; char Grade; int Mark;
        public:
        void Input();
        void Output();
};
int main(){
        Student sudent1;
        sudent1.Input();
        sudent1.Output();
        return 0;
}
void Student::Input(){
        getline(cin, Name);
        cin >> Grade >> Mark;
}
void Student::Output(){
         cout<< Name <<" "<< Grade<<" " << Mark;
}
```

- '_____' is called 'scope resolution operator', and is used to indicate that the functions belong to a particular class.
- For instance, `Student::Input()` means function `Input()` belongs to class _____.
- Had we only written `void Input()` and defined it, it would have meant that this function is not related to the class.
- Member functions in a class are identical to regular functions- they are defined normally, have a return type, name and arguments.

Passing Class Objects as Arguments to Functions

```
class Student{
        public:
        int Marks;
        void adder(Student studentX){
                Marks = Marks + studentX.Marks;}};
int main(){
        Student student1, student2, average;
        student1.Marks = 80;
        student2.Marks = 90;
        average.Marks = 0;
        cout << average.Marks << endl;
        average.adder(student1);
        cout << average.Marks << endl;
        average.adder(student2);
        cout << average.Marks << endl;
        average.Marks /= 2;
        cout << average.Marks << endl;
        return 0;}
```

- Similar to other variables, class objects can also be passed as arguments to functions.
- When running `Marks = Marks + studentX.Marks;`, the expression in the function `adder` is changed to

_____

Output:

Having a class as return-type of a function

```
class Student{
        public:
        int Marks;
        Student adder(Student studentX, Student studentY){
            Student studentA;
         studentA.Marks = studentX.Marks + studentY.Marks;
            return studentA;
        }
};
int main(){
        Student student1, student2, average;
        student1.Marks = 80;
        student2.Marks = 90;
        average = average.adder(student1 , student2);
        cout << average.Marks <<endl;
        average.Marks /= 2;
        cout << average.Marks <<endl;
        return 0;}
```

- We can return an object of the class `Student` when function `adder` is called.
- When running `studentA.Marks = studentX.Marks + studentY.Marks;`, the expression in the function `adder` is changed to

_____

Output: