計算機程式語言

物件導向程式設計

Object-Oriented Programming

Joseph Chuang-Chieh Lin
Dept. CSIE, Tamkang University

Platform

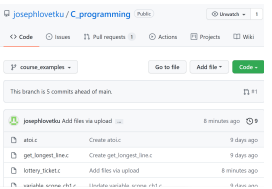
- Dev-C++

  Click here to download.

  Note: Please use this version otherwise you can't compile your programs/projects in Win10.

- OnlineGDB (https://www.onlinegdb.com/)

My GitHub page:
click the link here to visit.



- Other resources:
  - MIT OpenCourseWare - Introduction to C++ [link].
  - Learning C++ Programming [Programiz].

Useful Resources

- Tutorialspoint
  - https://www.tutorialspoint.com/cprogramming/index.htm
  - Online GCC Compiler
- Programiz
  - https://www.programiz.com/c-programming

Course TA

- 陳淯鑫
- Email address: 611410431@o365.tku.edu.tw

Grading Policy

- 出席率： 5%
- 平時作業 + 小考：35%
- 期中上機考 (4/16)：30%
- 期末上機考 (6/11)：30%
- CPE 加分：
  - 通過 1 題： +2%
  - 通過 2 題： +5%
  - 通過 3+ 題：+10%

作業繳交注意事項

- **課堂**作業需下課前繳交完畢
  - 補交期限為**當日**晚上 23:59:59，成績以 60% 計算
- 課後作業依公布的繳交期限為準
  - 補交期限為**當週週五**晚上 23:59:59，成績以 60% 計算

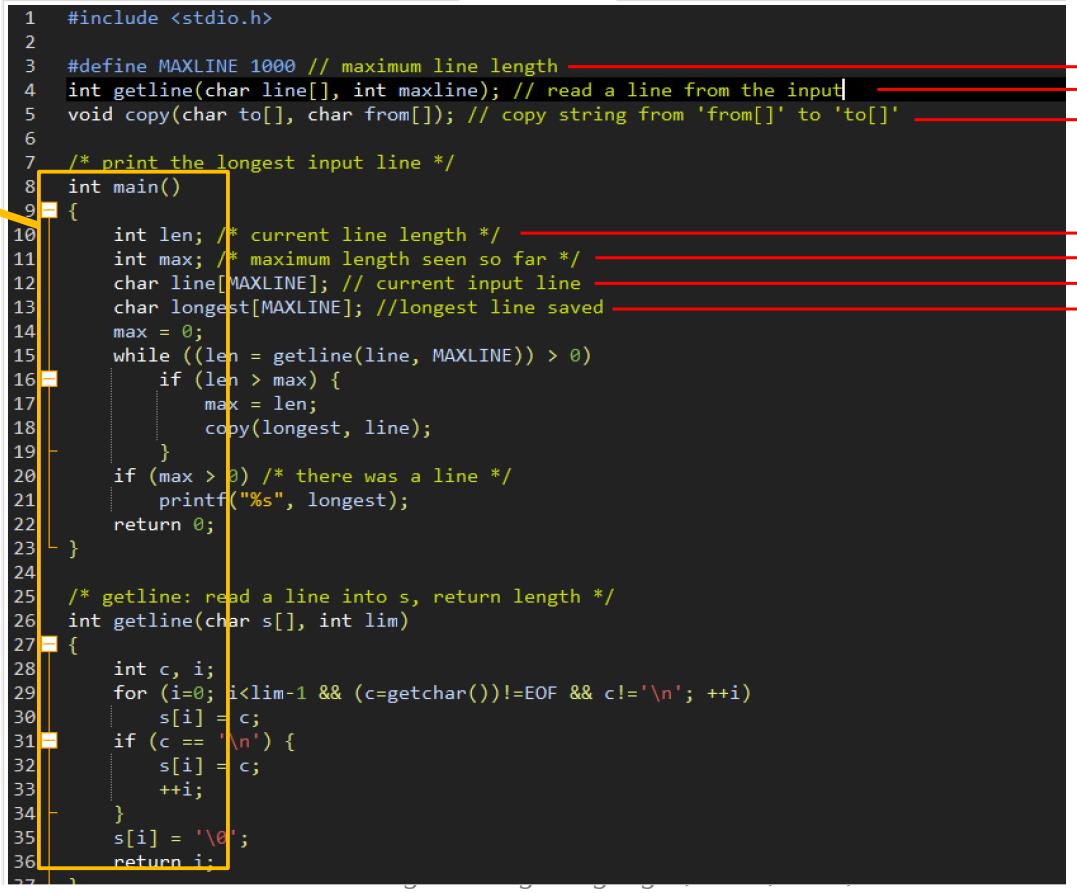
須上傳的檔案

- 程式原始碼 (.c 或 .cpp)
  - 請包含**註解**並請適當**縮排**
- 說明文件 (**若有要求繳交才需要**)
  - 純文字檔格式(例如 .txt 格式)
  - 請扼要說明解題邏輯(以 100 字以內為原則, 最多不要超過 200 字)
  - 解題邏輯:著重在"想法",而非說明程式執行流程。

程式碼範例

適當縮排

註解



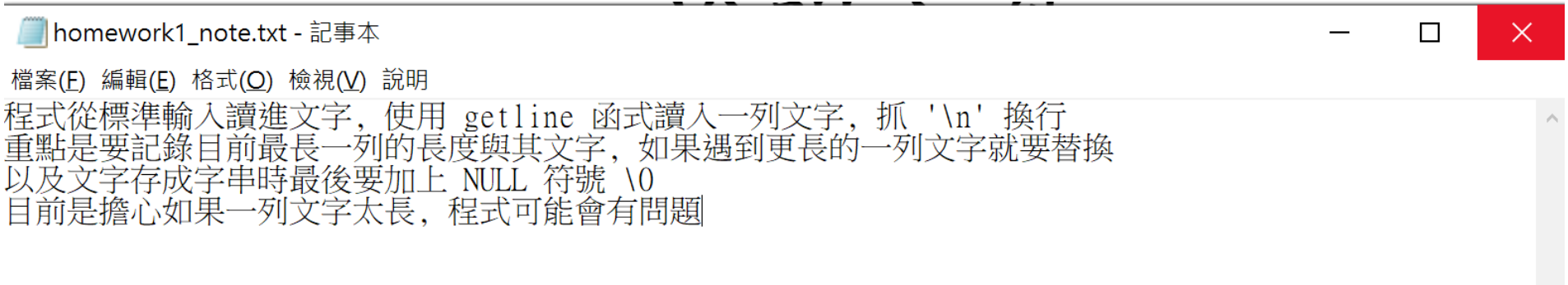
```c
#include <stdio.h>

#define MAXLINE 1000 // maximum line length
int getline(char line[], int maxline); // read a line from the input
void copy(char to[], char from[]); // copy string from 'from[]' to 'to[]'

/* print the longest input line */
int main()
{
    int len; /* current line length */
    int max; /* maximum length seen so far */
    char line[MAXLINE]; // current input line
    char longest[MAXLINE]; //longest line saved
    max = 0;
    while ((len = getline(line, MAXLINE)) > 0)
        if (len > max) {
            max = len;
            copy(longest, line);
        }
    if (max > 0) /* there was a line */
        printf("%s", longest);
    return 0;
}

/* getline: read a line into s, return length */
int getline(char s[], int lim)
{
    int c, i;
    for (i=0; i<lim-1 && (c=getchar())!=EOF && c!='\n'; ++i)
        s[i] = c;
    if (c == '\n') {
        s[i] = c;
        ++i;
    }
    s[i] = '\0';
    return i;
}
```

說明文件範例


homework1_note.txt - 記事本
檔案(F) 編輯(E) 格式(O) 檢視(V) 說明
程式從標準輸入讀進文字，使用 getline 函式讀入一列文字，抓 '\n' 換行
重點是要記錄目前最長一列的長度與其文字，如果遇到更長的一列文字就要替換
以及文字存成字串時最後要加上 NULL 符號 \0
目前是擔心如果一列文字太長，程式可能會有問題

（課後作業）評分標準補充

作業評分標準

- 程式原始碼檔案可成功編譯 +50%
- 通過作業公布測資+10%
- 通過所有測資+40%
  - 未通過所有測資則按通過比例給分。
    - 例如助教設計測資兩筆，僅過一筆 => + 20%

➢ 執行程式跑測資時當掉或是執行時間超過一分鐘以上，皆為未通過。

額外扣分準則

程式碼沒有註解：	-5%
程式碼沒有縮排：	-5%
說明文件未繳交：	-10%
有說明文件但未說明解題想法：	-10%

Introduction to OOP

Object-Oriented Programming Languages

- [Wikipedia] A programming paradigm based on the concept of "objects", which can contain data and code.
  - Data: in the form of fields (i.e., *attributes* or *properties*)
  - Codes: procedures (i.e., *methods*).
- Examples:
  - C++, JAVA, Python, Perl, Lisp.

- Most popular: *Class*-based.
  - Objects are instances of classes.

C++

- Created by Bjarne Stroustrup (starting in 1979 and has become generally available since 1985).
- Originally C with classes and renamed as C++.
- Efficiency of C is maintained.
- Applications in
  - System software
  - Application software
  - Device drivers
  - Embedded software
  - Games
  - High-performance computing server.

Why C++?

- Roughly a superset of C.
- Maintainability and Portability.
- International standard.
- General purpose.
- Powerful yet efficient.
- Low-level access to hardware.
- Easy to move to other OOP languages
  - But NOT in other direction.

The four basic concepts of OOP

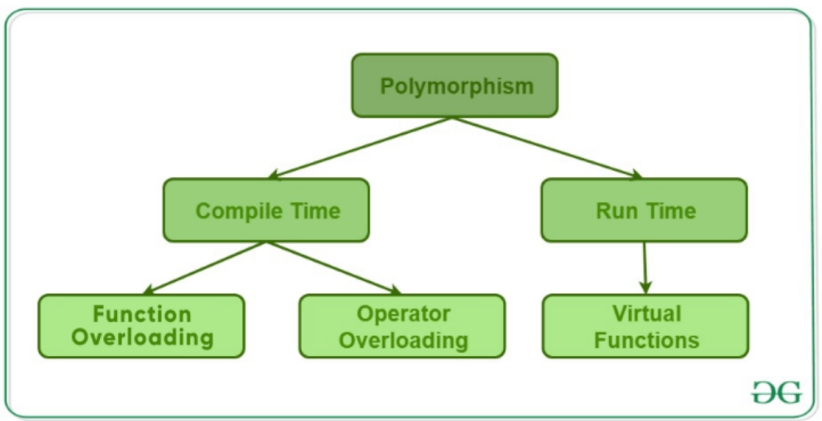
- **Encapsulation (封裝)**
  - Wrapping up of data and information under a single unit.
  - Binding together the data and the functions that manipulates them.
  - Emphasize on the “interface”.
  - Using classes.

- **Abstraction (抽象化)**
  - Displaying only essential information and hiding the details.
  - Providing only essential information about the data to the outside world.
  - Hiding the details and implementation.
  - Using classes or header files.

The four basic concepts of OOP

- **Inheritance (繼承)**
  - The ability of deriving properties and characteristics from another class.

- **Polymorphism (多型)**
  - The ability of a message to be displayed in more than one form.



Reference:
https://www.geeksforgeeks.org/polymorphism-in-c/?ref=lbp

GNU Compiler Collection (GCC)
C++ Compiler

```
g++ [options] input_file.cpp
```

To object code file:

```
g++ -c file_1.cpp // get file_1.o
g++ -c file_2.cpp // get file_2.o
```

To link object code files and produce a executable file:

```
g++ -o executable.exe file_1.o file_2.o
```

To directly produce a executable file from a C++ source code:

```
g++ -o executable.exe source.cpp
```

Your first C++ program

```
#include <iostream>

int main() {
    std::cout << "Welcome to C++!\n"; // :: scope resolution operator
    return 0;
}
```

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

int main() {
    cout << "Welcome to C++!\n";
    //cout << "Welcome to C++!" << endl;
    return 0;
}
```

A simple I/O example

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

int main() {
    int n1 = 0, n2 = 0;
    cout << "Enter two numbers: " << endl;
    cin >> n1 >> n2;
    cout << "The sum is " << n1+n2 << endl;

    return 0;
}
```

`main()` with no return type

```
#include <iostream>

using namespace std;

main()
{
     cout<<"Hello World";
}
```

```
#include <iostream>

using namespace std;

void main()
{
     cout<<"Hello World";
}
```

Continually getting input numbers

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

int main() {
    int sum = 0, value = 0;
    // continually getting numbers and sum over them until
    // reaching end-of-file
    while (cin >> value)
        sum += value;
    cout << "The sum is " << sum << endl;

    return 0;
}
```

Note:
In Windows OS, use `Ctrl+z` to represent end-of-file.
In Mac OS or Unix/Linux OS, use `Ctrl+d` to represent end-of-file.

String I/O

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

int main()
{
    char str[100];

    cout << "Enter a string: ";
    cin >> str;
    cout << "You entered: " << str << endl;

    return 0;
}
```

String I/O: read a line of text

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

int main()
{
    char str[100];
    //string str;

    cout << "Enter a string: ";
    cin.get(str, 100);
    //getline(cin, str);
    cout << "You entered: " << str << endl;

    return 0;
}
```

Structure

Define a customized data type: **struct**

```
struct Person
{
    char name[50];
    int age;
    float salary;
};
```

Three members of Person:

- name
- age
- Salary

No memory is allocated so far.
Just a blueprint for creating variables of such a datatype.

Define a customized data type: **struct**

```
struct Person
{
    char name[50];
    int age;
    float salary;
};
```

```
Person bill;
bill.age = 50;
bill.salary = 10000;

cout << bill.age;
cout << bill.salary;
```

Three members of Person:

- name
- age
- Salary

No memory is allocated so far.
Just a blueprint for creating variables of such a datatype.

Passing a structure to a function

```
struct Person {
    char name[50];
    int age;
    float salary;
};

void displayData(Person);

int main() {
    Person p;

    cout << "Enter Full name: ";
    cin.get(p.name, 50);
    cout << "Enter age: ";
    cin >> p.age;
    cout << "Enter salary: ";
    cin >> p.salary;

    // Function call with structure variable as an argument
    displayData(p);

    return 0;
}
```

```
void displayData(Person p) {
    cout << "\nDisplaying Information." << endl;
    cout << "Name: " << p.name << endl;
    cout << "Age: " << p.age << endl;
    cout << "Salary: " << p.salary;
}
```

Passing a structure to a function

```
struct Person {
    char name[50];
    int age;
    float salary;
};

void displayData(Person);

int main() {
    Person p;

    cout << "Enter Full name: ";
    cin.get(p.name, 50);
    cout << "Enter age: ";
    cin >> p.age;
    cout << "Enter salary: ";
    cin >> p.salary;

    // Function call with structure variable as an argument
    displayData(p);

    return 0;
}
```

```
void displayData(Person p) {
    cout << "\nDisplaying Information." << endl;
    cout << "Name: " << p.name << endl;
    cout << "Age: " << p.age << endl;
    cout << "Salary: " << p.salary;
}
```

Exercise (2%)
How to make it run successfully?

Return a structure from a function

```
struct Person {
    char name[50];
    int age;
    float salary;
};

void displayData(Person);
Person getData(Person);

int main() {
    Person p, temp;

    temp = getData(p);
    p = temp;
    displayData(p);

    return 0;
}
```

```
void displayData(Person p) {
    cout << "\nDisplaying Information." << endl;
    cout << "Name: " << p.name << endl;
    Cout << "Age: " << p.age << endl;
    cout << "Salary: " << p.salary;
}
```

```
Person getData(Person p) {

    cout << "Enter Full name: ";
    cin.get(p.name, 50);

    cout << "Enter age: ";
    cin >> p.age;

    cout << "Enter salary: ";
    cin >> p.salary;

    return p;
}
```

Benefit of using `struct`

leftUpCorner_x, leftUpCorner_y,

rightDownCorner_x, rightDownCorner_y,

```
double area(double leftUpCorner_x,
            double leftUpCorner_y,
            double rightDownCorner_x,
            double rightDownCorner_y) {
    ...
}
```

Benefit of using `struct`

leftUpCorner_x, leftUpCorner_y,

rightDownCorner_x, rightDownCorner_y,

```
double area(double leftUpCorner_x,
            double leftUpCorner_y,
            double rightDownCorner_x,
            double rightDownCorner_y) {
    ...
}
```

```
struct Rect {
    double leftUpCorner_x;
    double leftUpCorner_y;
    double rightDownCorner_x;
    double rightDownCorner_y;
};
```

Exercise (5%)

- Using `struct` to compute the area of the input axis-parallel rectangle

```
struct Rect {
    double leftUpCorner_x;
    double leftUpCorner_y;
    double rightDownCorner_x;
    double rightDownCorner_y;
};
```

SAMPLE INPUT:
```
10 20 30 -10
```

SAMPLE OUTPUT:
```
600
```

Pointer to a structure

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

struct Distance {
    int feet;
    float inch;
};

int main() {
    Distance *ptr, d;

    ptr = &d;

    cout << "Enter feet: ";
    cin >> (*ptr).feet;
    cout << "Enter inch: ";
    cin >> (*ptr).inch;

    cout << "Displaying information." << endl;
    cout << "Distance = " << (*ptr).feet << " feet " << (*ptr).inch << " inches";

    return 0;
}
```

Defining the Sales_data type

```
struct Sales_data {
    std::string bookNo;
    int units_sold = 0;
    double revenue = 0.0;
};
```

Class

Class

- A blueprint/prototype/sketch for the **object**.
- To design a "car", you need:
  - Wheels
  - Engines
  - A steering wheel
  - Windows
  - Lights
  - …
  -

Class

- A blueprint/prototype/sketch for the **object**.
- To design a "school", you need:
  - Buildings
    - Windows
    - Chairs
    - A Blackboard
    - ...
  - Teachers
  - Students
  - Walls
  - Staffs

Class

- We define our own data structures by defining a **class**.
- A class defines a type along with a collection of operations that are related to that type.
- A primary focus of the design of C++ codes is to make it possible to define **class types** that behave as naturally as the built-in types.

Create a class (by an example)

```
class Room {
    public:
        double length;
        double breadth;
        double height;

        double calculateArea(){
            return length * breadth;
        }

        double calculateVolume(){
            return length * breadth * height;
        }

};
```

data members
資料成員

member functions
成員函式

Create a class (by an example)

```
class Room {
    public:
        double length;
        double breadth;
        double height;

        double calculateArea(){
            return length * breadth;
        }

        double calculateVolume(){
            return length * breadth * height;
        }

};
```

```
int main() {
    // create objects of Room
    Room r1, r2;

    //assign values to data members
    r1.length = 42.5;
    r1.breadth = 30.8;
    r1.height = 19.2;

    //calculate the area
    cout << "area: ";
    cout << r1.calculateArea();
    cout << endl;
}
```

The keyword “public” and “private”

- **public**: the members in the class can be assessed anywhere from the program.
- **private**: the members can only be assessed from within the class (i.e., member functions).

Example of using **private**

```
class Room {
    private:
        double length;
        double breadth;
        double height;
    public:
        void initData(double len, double brth, double hgt) {
            length = len;
            breadth = brth;
            height = hgt;
        }
        double calculateArea(){
            return length * breadth;
        }
        double calculateVolume(){
            return length * breadth * height;
        }
};
```

Example of using **private**

```
class Room {
    private:
        double length;
        double breadth;
        double height;
    public:
        void initData(double len, double brth,
            length = len;
            breadth = brth;
            height = hgt;
        }
        double calculateArea(){
            return length * breadth;
        }
        double calculateVolume(){
            return length * breadth * height;
        }
};
```

```
int main() {
    // create objects of Room
    Room r1;

    //initial the Room object
    r1.initData(42.5, 30.8, 19.2);

    //calculate the area
    cout << "area: ";
    cout << r1.calculateArea();
    cout << endl;

    cout << "volume: ";
    cout << r1.calculateVolume();
    cout << endl;
}
```

Learn from a well-designed class in a header file.

The Bookstore transactions

- We shall learn how to “use” a class.
- `Sales_item.h` [link to download]

```
#include <iostream>
#include "Sales_item.h"
int main()
{
    Sales_item book;
    // read ISBN, number of copies sold, and sales price
    std::cin >> book;
    // write ISBN, number of copies sold, total revenue, and average price
    std::cout << book << std::endl;
    return 0;
}
```

The Bookstore transactions

- Sample input of previous code:

```
0-201-70353-X 4 24.99
```

- The sample output:

```
0-201-70353-X 4 99.96 24.99
```

The Bookstore transactions - Adding `Sales_items`

```
#include <iostream>
#include "Sales_item.h"
int main()
{
    Sales_item item1, item2;
    std::cin >> item1 >> item2;
    // read a pair of    transactions
    std::cout << item1 + item2 << std::endl;
    // print their sum
    return 0;
}
```

Sample input of previous code:

```
0-201-78345-X 3 20.00
0-201-78345-X 2 25.00
```

The sample output:

```
0-201-70353-X 5 110 22
```

The Bookstore transactions
- Example of Member functions/methods

```
#include <iostream>
#include "Sales_item.h"
int main()
{
    Sales_item item1, item2;
    std::cin >> item1 >> item2;
    // first check that item1 and item2 represent the same book
    if (item1.isbn() == item2.isbn()) {
        std::cout << item1 + item2 << std::endl;
        return 0; // indicate success
    } else {
        std::cerr << "Data must refer to same ISBN"
                  << std::endl;
        return -1; // indicate failure
    }
}
```

The Bookstore Program

```
#include <iostream>
#include "Sales_item.h"
int main()
{
     Sales_item total; // variable to hold data for the next transaction
     // read the first transaction and ensure that there are data to process
     if (std::cin >> total) {
          Sales_item trans; // variable to hold the running sum
          // read and process the remaining transactions
          while (std::cin >> trans) {
               // if we're still processing the same book
               if (total.isbn() == trans.isbn())
                    total += trans; // update the running total
               else {
                    // print results for the previous book
                    std::cout << total << std::endl;
                    total = trans; // total now refers to the next book
               }
          }
          std::cout << total << std::endl; // print the last transaction
     } else {
          // no input! warn the user
          std::cerr << "No data?!" << std::endl;
          return -1; // indicate failure
     }
     return 0;
}
```

Course Assignment (2%)

- Please compile the BookStore program and run an example.
- Get a snapshot of the successful execution of the program and then upload it with the source codes to iClass.

Supplementary

- Difference between

  ```
  cout << "blah blah ..." << endl;
  ```

  and

  ```
  cout << "blah blah ..." << "\n";
  ```

- `cout << "blah blah ..." << endl;`

  is equivalent to

  `cout << "blah blah ..." << "\n"` **and then flush the buffer.**