

Miscellaneous C++ Review topics

- การอ่านและเขียนไฟล์
- อาร์เรย์ (Array) และสายอักขระ (string)
- โครงสร้าง struct
- String and struct
- "this" pointer

คพ.213 โครงสร้างข้อมูล
Lecture 3

ปรับปรุงจากสไลด์ อ.เยาวดี เต็มธนาภัทร์

การอ่านและเขียนไฟล์

การอ่านและเขียนไฟล์

- **fstream** เป็นไลบรารีมาตรฐานที่ช่วยให้เราสามารถดำเนินการบางอย่างกับไฟล์ โดยจะมี 3 stream classes
 - ofstream => Stream class เพื่อเขียนข้อมูลลงไฟล์
 - ifstream => Stream class เพื่ออ่านข้อมูลจากไฟล์
 - fstream => Stream class เพื่ออ่านข้อมูลและเขียนข้อมูลกับไฟล์

ในการใช้คลาสที่อยู่ใน fstream library เราต้อง include เข้ามาในโปรแกรมด้วย

#include <fstream>

การเปิดไฟล์ (1)

■ ใช้ฟังก์ชัน open()

```
open(path, mode);
```

โหมด	คำอธิบาย
ios::in	เปิดไฟล์สำหรับการอ่านข้อมูลจากไฟล์
ios::out	เปิดไฟล์สำหรับการเขียนข้อมูลลงไปในไฟล์
ios::binary	โหมดไฟล์สำหรับการทำงานในโหมด binary
ios::ate	เปิดไฟล์และกำหนดให้ไฟล์ pointer ชี้ไปยังส่วนท้ายของไฟล์ ถ้าหากเราไม่กำหนด flag นี้จะชี้ไปที่จุดเริ่มต้นแทน
ios::app	เปิดไฟล์สำหรับการเขียนข้อมูลลงไปที่ท้ายของไฟล์
ios::trunc	ถ้าหากไฟล์ถูกเปิดสำหรับใหม่เขียนลงไปในไฟล์และมีไฟล์อยู่ โปรแกรมจะทำการลบไฟล์และสร้างใหม่

การเปิดไฟล์ (2)

- สำหรับการเปิดไฟล์ด้วย Stream object แต่ละแบบในภาษา C++ นั้นมีโหมดพื้นฐาน ดังนี้

คลาส	Default โหมดเมื่อสร้างออบเจกต์
ofstream	ios::out
ifstream	ios::in
fstream	ios::in ios::out

สามารถใช้ 2 modes พร้อมกันได้ ด้วย
การเชื่อมด้วย | (OR)

ตัวอย่างการเปิดไฟล์ในแบบต่างๆ

```
// open for write  
ofstream myfile1;  
myfile1.open("example1.txt");
```

```
// open file for read  
ifstream myfile2;  
myfile2.open("example2.txt");
```

```
// open file for read and write  
fstream myfile3;  
myfile3.open("example3.txt");
```

```
// open file for write and appending  
ofstream myfile4;  
myfile4.open("example4.txt", std::ios::out | std::ios::app);
```

การปิดไฟล์

- ใช้ฟังก์ชัน `close()`

```
myfile1.close();
```

```
myfile2.close();
```

```
myfile3.close();
```

การเขียนข้อมูลลงไฟล์ (text files)

- สามารถเขียนตัวอักษร ข้อความหรือตัวเลข ลงไปในไฟล์โดยใช้เครื่องหมาย << กับไฟล์ที่เราต้องการจะเขียน

```
ofstream myfile;
```

```
myfile.open ("example.txt");  
myfile << "This is the first line." << endl;  
myfile << "The second line." << endl;  
myfile << 10.5;  
myfile.close();
```

example.txt

```
This is the first line.  
The second line.  
10.5
```

```
return 0;
```


การอ่านข้อมูลจากไฟล์ (text files)

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

int main()
{
    ifstream myfile;
    string line;
    myfile.open ("example.txt");
    while (getline(myfile,line))
    {
        cout << line << '\n';
    }
    myfile.close();
    return 0;
}
```

**This is the first line.
The second line.
10.5**

การเขียนไฟล์ต่อท้าย (File appending)

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

int main()
{
    fstream myfile;
    myfile.open("example.txt",
               std::ios::in | std::ios::out | std::ios::app);

    myfile << endl << "append new text to the file." << endl;
    myfile << "This is the last line." << endl;

    myfile.close();

    return 0;
}
```

Basic Write/Read File

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

```
int main() {
    string filename = "fruit.txt";
    // Write to File
    ofstream fout(filename); // default mode is ios::out | ios::trunc
    if (!fout) {
        cerr << "Error: open file for output failed!" << endl;
        abort(); // in <cstdlib> header
    }
    fout << "apple" << endl; fout << "kiwi" << endl; fout << "banana" << endl;
    fout.close();

    // Read from file
    ifstream fin(filename); // default mode ios::in
    if (!fin) {
        cerr << "Error: open file for input failed!" << endl;
        abort();
    }
    char ch;
    while (fin.get(ch)) { // read a character at a time until the end-of-file
        cout << ch;
    }
    fin.close();
    return 0;
}
```

```
apple
kiwi
banana
```

```
Or
string str;
while(getline(fin, str))
    cout << str << endl;
```

อาร์เรย์ (ARRAY)

อาร์เรย์

- อาร์เรย์เป็นพื้นที่ต่อเนื่องในหน่วยความจำ
- การประกาศอาร์เรย์

`type variable[size];`

เช่น `int x[10];` จองพื้นที่สำหรับ `int` 10 จำนวน

- การเข้าถึงอาร์เรย์: ระบุตำแหน่ง โดย []

- ดัชนีของอาร์เรย์ใน C เริ่มต้นที่ 0

เช่น `x[0]` อ้างถึงอาร์เรย์ในตำแหน่งแรก

- การประกาศพร้อมให้ค่า

`int x[] = {9, 3, 4, 5};` ประกาศและให้ค่ากับ
อาร์เรย์ขนาด 4 ภายในมีค่าเป็น 9, 3, 4, 5 ตามลำดับ

อาร์เรย์หลายมิติ (1)

```
char x[3][2] = { {'a', 'b'}, {'c', 'd'}, {'e', 'f'} };
```

- x เป็นอาร์เรย์ขนาด 3x2
- โดย convention, subscript ตัวแรกแสดงแถว (row) และ subscript ตัวที่สองแสดงหลัก (column)
- โดย x จะถูกให้ค่าเริ่มต้นเป็น:

	col 0	col 1
row 0	a	b
row 1	c	d
row 2	e	f

อาร์เรย์หลายมิติ (2)

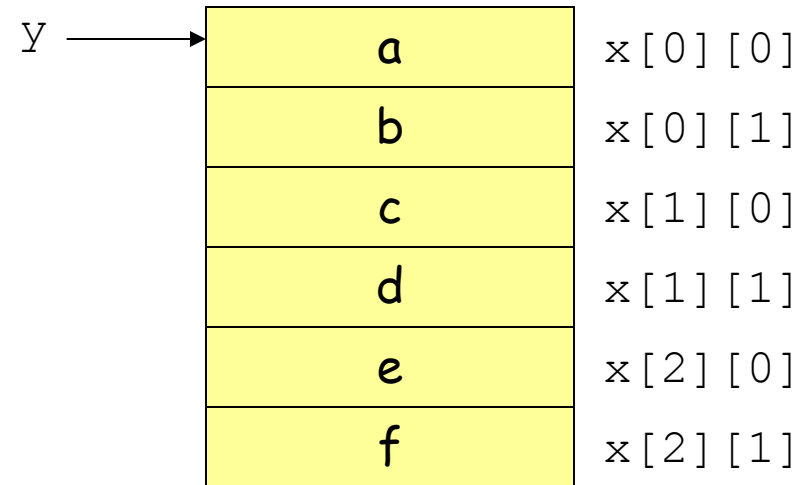
- แต่หน่วยความจำเป็น linear
 - อาร์เรย์ถูกเก็บจริงในหน่วยความจำอย่างไร?

```
char x[3][2] = {{ 'a', 'b' },  
                { 'c', 'd' },  
                { 'e', 'f' }};
```

```
char* y = &(x[0][0]);
```

```
for (int i = 0; i < 6; i++) {  
    cout << *y;  
    y++;  
}
```

```
// prints "abcdef"
```



การส่งผ่านอาร์เรย์หลายมิติไปฟังก์ชัน

- caller สามารถส่งโดยอ้างชื่อของอาร์เรย์ได้โดยตรง
- ฟังก์ชัน จำเป็นต้องรู้โครงสร้างของอาร์เรย์ ดังนั้นจำเป็นต้องระบุ *ขนาด* ของ subscripts ตัวที่ 2, 3, etc.

- ตัวเรียก เรียกโดย:
`func(x) ;`

- One-subscripted array parameter:
`void func (int x[]) {
 ...
}`

- Two-subscripted array parameter:
`void func (int x[][2]) {
 ...
}`

ต้องระบุขนาดของมิติอื่น ๆ

สายอักขระ (STRING)

สายอักขระ

- สายอักขระ สามารถกำหนดในรูปแบบ
 - อาร์เรย์ของตัวอักษร `char str[]`
 - การจองหน่วยความจำเป็นแบบ static
 - pointer ไปยังตัวอักษร `char* str` หรือ
 - การจองพื้นที่เก็บตัวชี้ ต้องจองหน่วยความจำเพื่อเก็บค่าก่อนใช้
 - ใช้คลาส `string` ใน `<string>`: `string str`
 - การจองหน่วยความจำเป็นแบบพลวัต (Dynamic Allocation) ยืดขยายหรือหดตามความจำเป็นได้อัตโนมัติ
 - ในแง่ประสิทธิภาพการทำงานจะช้ากว่าการใช้อาร์เรย์

การประกาศและให้ค่าสายอักขระ

- ประกาศสายอักขระและให้ค่าเป็น Hello
`char str[6] = {'H', 'e', 'l', 'l', 'o', '\0'};`
`char str[] = "Hello";`
`string str("Hello");`
- การหาความยาวสายอักขระ
 - ❑ อาร์เรย์ของอักขระ: `strlen(str)`
 - ❑ string: `str.length()`
- การต่อสายอักขระ (concatenate)
 - ❑ อาร์เรย์ของอักขระ: `strcat`
 - ถ้าขนาดของอาร์เรย์ใหญ่ไม่พอเพียง ผลลัพธ์ขาดเดาไม่ได้
 - ❑ string: `+` หรือใช้เมทอด `append`

C++ ฟังก์ชันที่ทำงานกับ string

Function Name		Function's Description
string&	append(const string& str1)	ต่อสายอักขระเข้าด้านท้าย
char&	at(int pos)	คืนตัวอักษรที่อยู่ตำแหน่ง pos
const char*	c_str()	คืน pointer ชีอาร์เรย์ของอักขระที่มีค่าเหมือน string ต้นแบบ
int	compare(const string& str1)	เปรียบเทียบค่า string
int	find(string& str1, int pos, int len)	ค้นค่าตำแหน่งของ str1 ตั้งแต่ตำแหน่ง pos ยาว len ตัว
int	length()	คืนความยาวของ string
string&	replace(int pos, int n, string& str1)	แทนตั้งแต่ตำแหน่งที่ pos จำนวน n ตัวด้วยค่าของ str1
void	resize(int length)	เปลี่ยนขนาดความยาว string ตามขนาดที่ให้
int	size()	บอกขนาด (bytes) ของ string
string	substr(int pos, int len)	สร้าง string ย่อยจาก string เดิมเริ่มที่ตำแหน่ง pos จำนวน len
void	swap(string& str1)	สลับค่าของ string กับพารามิเตอร์ที่ส่งให้

C ฟังก์ชันที่ทำงานกับ string

Function Name		Function's Description
char *	strcat(char *dest, const char *src)	ต่อสายอักขระที่ชี้โดย src ด้านท้ายของสายอักขระที่ชี้โดย dest
int	strcmp(const char *str1, const char *str2)	เปรียบเทียบสายอักขระ src1 กับ src2 (คืนบวก ลบ หรือ 0)
size_t	strlen(const char *str)	คืนค่าความยาวสายอักขระ str (ไม่นับตัวสิ้นสุด null character)
char*	strncpy(char* dest, char* src, int n)	สำเนาสายอักขระ src ไปไว้ที่ dest จำนวน n ตัว
char*	strtok(char *str, const char *delim)	แบ่งสายอักขระ str เป็นชุดของคำย่อย (tokens) โดยใช้ตัวแบ่ง delim

```
#include <iostream>
#include <string.h>
using namespace std;
int main() {
```

ตัวแปรสายอักขระ str1, str2 ประกาศแบบอาร์เรย์และให้ค่า Hello

```
char str1[] = "Hello";
char str2[6] = {'H', 'e', 'l', 'l', 'o', '\0'}; // "Hello";
```

```
char *str3 = new char[6];
strcpy(str3, "Hello");
```

ตัวแปรสายอักขระ str3 ประกาศแบบ pointer และให้ค่า Hello

```
string str4 = "Hello";
string str5("Hello");
```

ตัวแปรสายอักขระ str4, str5 ประกาศแบบ string และให้ค่า Hello

```
cout << "str1 " << str1 << " " << strlen(str1) << endl;
cout << "str2 " << str2 << " " << strlen(str2) << endl;
cout << "str3 " << str3 << " " << strlen(str3) << endl;
cout << "str4 " << str4 << " " << str4.length() << endl;
cout << "str5 " << str5 << " " << str5.length() << endl;
```

อาร์เรย์ char ขนาดตายตัว
string ขนาดเป็น dynamic

```
char concatStr[sizeof(str1) + sizeof(str2)];
strcpy(concatStr, str1);   strcat(concatStr, str2); // copy then concat
str4 += str5; //or str4.append(str5);
cout << "concatStr " << concatStr << " " << strlen(concatStr) << endl;
cout << "str4 " << str4 << " " << str4.length() << endl;
return 0;
```

}

STRUCT

Structure

■ struct

- ❑ ใช้ในการกำหนด container เพื่อเก็บข้อมูลเกี่ยวกับสิ่งของ
- ❑ จัดองค์กร (organize) ข้อมูลที่เกี่ยวข้องกันไว้ในก้อนเดียว
- ❑ สร้างเป็นชนิด (type) อย่างที่ต้องการ

เราอาจกำหนด struct ใน
header file เพื่อความ
สะดวกในการนำไปใช้

■ Student Record

- ❑ **name** – String
- ❑ **HW score** – อาร์เรย์ของ double
ขนาด 4
- ❑ **exam score** – อาร์เรย์ของ
double ขนาด 2
- ❑ **total score** – double

```
struct StudentRecord {  
    char *name;  
        // student name  
    double hw[4];  
        // homework scores  
    double exam[2];  
        // exam scores  
    double total;  
        // total score  
};
```


อีกตัวอย่างของการใช้ Header File

```
#ifndef STUSTRUCT_H_
#define STUSTRUCT_H_
struct StudentRecord {
    char *name;
    // student name
    double hw[4];
    // homework scores
    double exam[2];
    // exam scores
    double total;
    // total score
};

#endif /* STUSTRUCT_H_ */
```

การใช้งาน struct

- ประกาศตัวแปรของชนิด struct ใหม่ โดย

```
StudentRecord stu;
```

- เข้าถึงสมาชิกใน struct ผ่าน *operator* '.' ("dot"):

```
cout << stu.name << endl;
```

```
stu.hw[0] = 2.4;
```

```
percent = stu.total/100;
```

- การให้ค่า (assignment) :

```
StudentRecord s1, s2;
```

```
s1.name = new char[ (sizeof("John Smith")) ];
```

```
strcpy(s1.name, "John Smith");
```

```
...
```

```
s2 = s1;
```

สำเนา **structure** ทั้งหมด

สิ่งที่ควรระวัง

- ถ้าสมาชิกเป็น pointer, การสำเนาหมายถึงการ *สำเนา pointer* (ไม่ใช่สิ่งที่ถูก pointed)

```
StudentRecord s1, s2;
```

```
...
```

```
strcpy(s1.name, "John Smith);
```

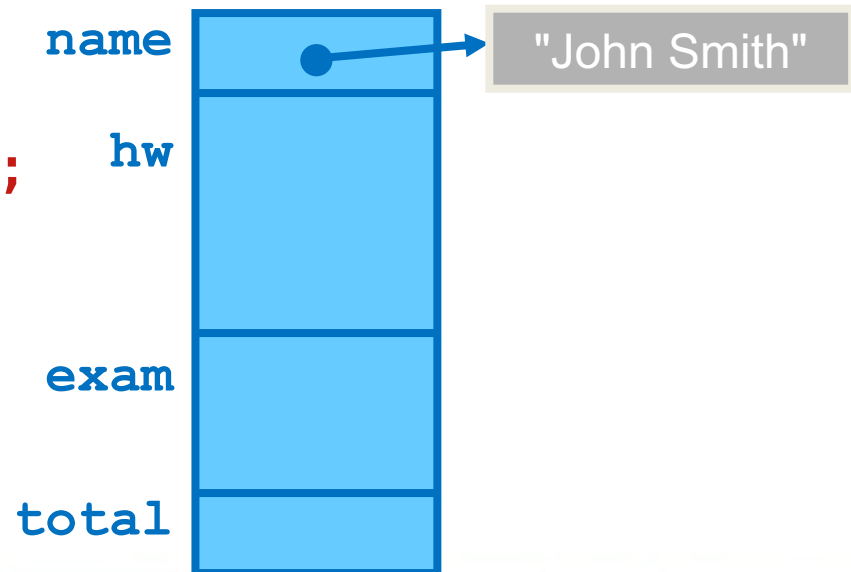
```
...
```

```
s2 = s1;
```

```
strcpy(s2.name, "Jane Doe");
```

```
// s1.name และ s2.name
```

```
// ชื่อมีค่า "Jane Doe" ทั้งคู่
```



```

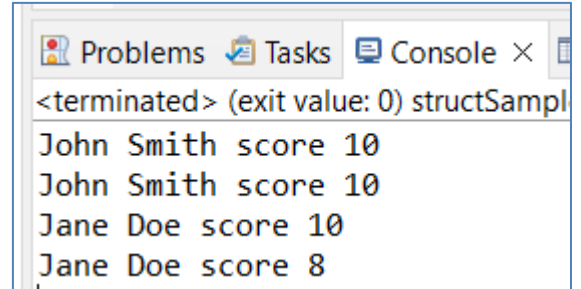
#include <iostream>
#include <string.h>
#include "stuStruct.h"
using namespace std;

int main() {
    StudentRecord s1, s2;
    s1.name = new char[sizeof("John Smith")];
    strcpy(s1.name, "John Smith");
    s1.hw[0] = 10;

    s2 = s1;
    cout << s1.name << " score " << s1.hw[0] << endl;
    cout << s2.name << " score " << s2.hw[0] << endl;

    s2.hw[0] = 8;
    strcpy(s2.name, "Jane Doe");
    cout << s1.name << " score " << s1.hw[0] << endl;
    cout << s2.name << " score " << s2.hw[0] << endl;
    return 0;
}

```



```

Problems Tasks Console ×
<terminated> (exit value: 0) structSampl
John Smith score 10
John Smith score 10
Jane Doe score 10
Jane Doe score 8

```

Pointers ชี้ไปยัง Structures

- การเข้าถึงสมาชิกใน structure เมื่อตัวแปรชนิดเป็น pointer โดย ->

```
StudentRecord *stuPtr = new StudentRecord();  
stuPtr->total = 20;  
...  
cout << "Student name is" << stuPtr->name;  
cout << "Total is " << stuPtr->total;
```

ตัวอย่างฟังก์ชัน (ทำงานไม่ได้ตามต้องการ)

```
void updateTotal(StudentRecord stu) {  
    stu.total = 0;  
  
    for (int i=0; i<4; i++)  
        stu.total += stu.hw[i];  
    for (int i=0; i<2; i++)  
        stu.total += stu.exam[i];  
}
```

ตัวอย่างฟังก์ชัน (ทำงานได้ตามต้องการ)

```
void updateTotal(StudentRecord *stu) {  
    stu->total = 0;  
  
    for (int i=0; i<4; i++)  
        stu->total += stu->hw[i];  
    for (int i=0; i<2; i++)  
        stu->total += stu->exam[i];  
}
```

หรือใช้ reference parameter

```
void updateTotal(StudentRecord &stu) {  
    stu.total = 0;  
  
    for (int i=0; i<4; i++)  
        stu.total += stu.hw[i];  
    for (int i=0; i<2; i++)  
        stu.total += stu.exam[i];  
}
```


ฟังก์ชันใน struct

- สามารถมีฟังก์ชันใน structure (เรียกว่า *member functions*) ได้
 - แต่ทุก members เป็น public (เห็นได้จากภายนอก)
- *class* ใน C++ มีลักษณะคล้ายกับ Struct โดยที่
 - Classes สามารถมี (data) members
 - Classes สามารถมี member functions
 - Classes สามารถ *ซ่อน (hide)* ส่วนของ members (functions และ data) ได้

ตัวอย่างอย่างง่าย

```
struct StudentRecord {  
    char *name;           // student name  
    double hw[4];         // homework scores  
    double exam[2];       // exam scores  
    double total;         // total score  
  
    void printTotal () {  
        cout << "Name: " << name << endl;  
        cout << "Total: " << total << endl;  
    }  
};
```

การเรียกใช้

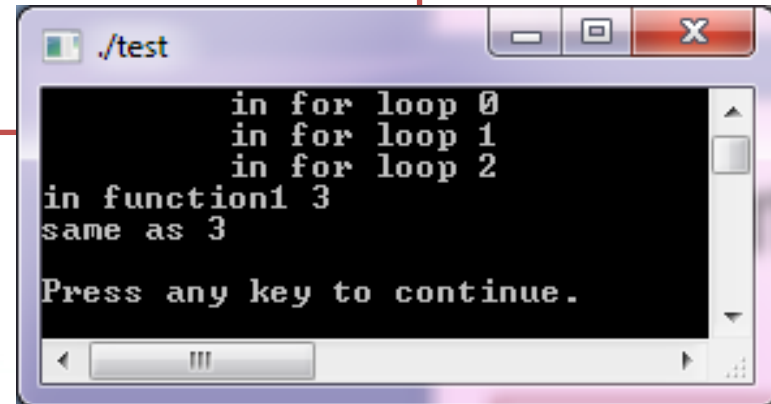
```
StudentRecord stu;  
:  
stu.printTotal();
```

this

- แต่ละวัตถุใน C++ maintains จะมี pointer ที่ชี้ไปที่ตัวเอง เรียกว่า "this" เป็นสมาชิกอยู่แบบอัตโนมัติ
- ในวัตถุ สามารถใช้ this เพื่อ
 - ระบุ address ของวัตถุนั้น
 - อ้างถึงวัตถุนั้น
 - เข้าถึงสมาชิกทุกตัวของวัตถุนั้น โดยเฉพาะในกรณีที่เกิด shadowed variable

Example

```
void Class1::function1(int param1){
    int var1 = 8;
    for (int i = 0; i < 3; i ++){
        int var1 = i;
        cout << "\t in for loop " << var1 << endl;
    }
    cout << "in function1 " << this->var1 << endl;
    cout << "same as " << (*this).var1 << endl;
}
int main (){
    Class1 c;
    c.function1(3);
}
```



```
./test
        in for loop 0
        in for loop 1
        in for loop 2
in function1 3
same as 3

Press any key to continue.
```