Yuan Ze University

C++ 程式初探 IV

2015暑期

lobal Logistics Lab.

ver. 1.0.2

C++ 程式 IV – 大綱

1. 時間函式
2. 格式化輸出
3. 遞迴函式(recursion)
4. 字串
5. 字串轉型

補充語法 時間計算

引入標頭檔

```
#include <ctime>
```

時間變數

```
clock_t a, b, c;
```

取得現在時間

```
clock();
```

計算時間

```
double(end-start)/CLK_TCK;
```

```
#include <iostream>
#include <Windows.h>
#include <ctime>
using namespace std;

int main(void) {
    clock_t start = clock(); //開始
    Sleep(500);
    clock_t end = clock();   //結束
    cout<<"elapsed time: "
    cout<<double(end-start)/CLK_TCK<<endl; //計算時間差
    system("PAUSE");
    return 0;
}
```

注意clock()函式精度(precision)不佳，精度約15毫秒 (毫秒為10^{-3} 秒)，而準度 (accuracy)則依系統而異，常有誤差。

補充語法　時間計算

引入標頭檔

```
#include <Windows.h>
```

```
#include <iostream>
#include <Windows.h>
#include <ctime>
using namespace std;

int main(void) {
    LARGE_INTEGER start, end, freq;
    QueryPerformanceFrequency(&freq); //取得CPU時脈
    QueryPerformanceCounter(&start);  //記錄開始時間
    Sleep(500);
    QueryPerformanceCounter(&end);    //記錄結束時間
    cout<<"elapsed time: ";
    //計算時間差
    cout<<(end.QuadPart-start.QuadPart)/(double)(freq.QuadPart)<<endl;
    system("PAUSE");
    return 0;
}
```

此計時方法精度極高，精度達微秒 (10^{-6} 秒)。準度依系統而異，而經實測發現準度超過精度(至少微秒)。

參考資料:
https://msdn.microsoft.com/zh-tw/library/windows/desktop/dn553408(v=vs.85).aspx

格式化輸出

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

int main(void)
{
    ofstream fout("result.txt");

    for(int i=1; i<=3; i++) {
        for(int j=1; j<4; j++)
            fout<<left<<setw(5)<<(i*j);
        fout<<endl;
    }

    system("PAUSE");
    return 0;
}
```

固定格式輸出

setw(n)

向右對齊(default)

right

向左對齊

left

格式化輸出

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

int main(void)
{
    double a = 4560/17.0;
    cout<< a <<endl;
    cout<<fixed<<setprecision(10)<< a <<endl; //設定小數點10位
    cout<<scientific<< a <<endl;              //科學記號表示
    cout<<fixed<< a <<endl;                   //一般數學表示

    system("PAUSE");
    return 0;
}
```

精確度的設定，只要使用一次，之後都會採用這個設定。

格式化標頭檔

```
#include <iomanip>
```

精確度

```
setprecision(n)
```

一般數字表示

```
fixed
```

科學記號表示

```
scientific
```

檔案輸出

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

int main(void)
{
    int number = 6;
    ofstream fout("result.txt",ios::app);

    fout<<"number = ";
    fout<<number;
    fout<<endl;

    system("PAUSE");
    return 0;
}
```

輸出類別 變數名稱 (檔名字串);

```
ofstream fout("fname.txt");
ofstream fout("fname.txt",ios::app);
```

不覆蓋，接續在檔案後

檔案輸出

```
#include<fstream>
#include<iostream>
using namespace std;
int main(void)
{
    int number;
    ifstream fin;
    fin.open("data.txt");
    if(!fin.is_open()) {
        cout<<"file cannot be found..."<<endl;
        system("PAUSE");
        exit(1);
    }
    fin>>number;
    fin.close();
    system("PAUSE");
    return 0;
}
```

開啟檔案

```
fout.open("fname.txt");
```

關閉檔案

```
fout.close();
```

檢查是否開啟

```
fout.is_open();
```

開啟檔案後，務必檢查

函式

```
int pow(int n, int p) {
     int r = 1;
     for(int i=0; i<p; i++)
         r *= n;
     return r;
}

int max(int n, int a[]) {
    int max = a[0];
    for(int i=1; i<n; i++)
        if(a[i]>max)
            max = a[i];
    return max;
}
```

函數

回傳型別 函式名稱 (變數型態 變數名稱)

```
int  fun  (int  input)
```

函數 - 陣列引數

回傳型別 函式名稱 (變數型態 變數名稱)

```
int  fun  (int  ary[])
int  fun  (int  ary[][3])
```

補充語法 函式多載(overload)

```
int max(int n, int a[]) {
    int max = a[0];
    for(int i=1; i<n; i++)
        if(a[i]>max)
            max = a[i];
    return max;
}

int max(int a, int b) {
    if(a>b)
        return a;
    else
        return b;
}
```

函數多載

```
int fun (int a);
int fun (int a, int b);
int fun (double a, double b);
int fun (int a, int b, int c);
```

回傳值不一樣不可多載

```
int fun (int a, int b);
double fun (int a, int b);
```

補充語法 函式預設參數值

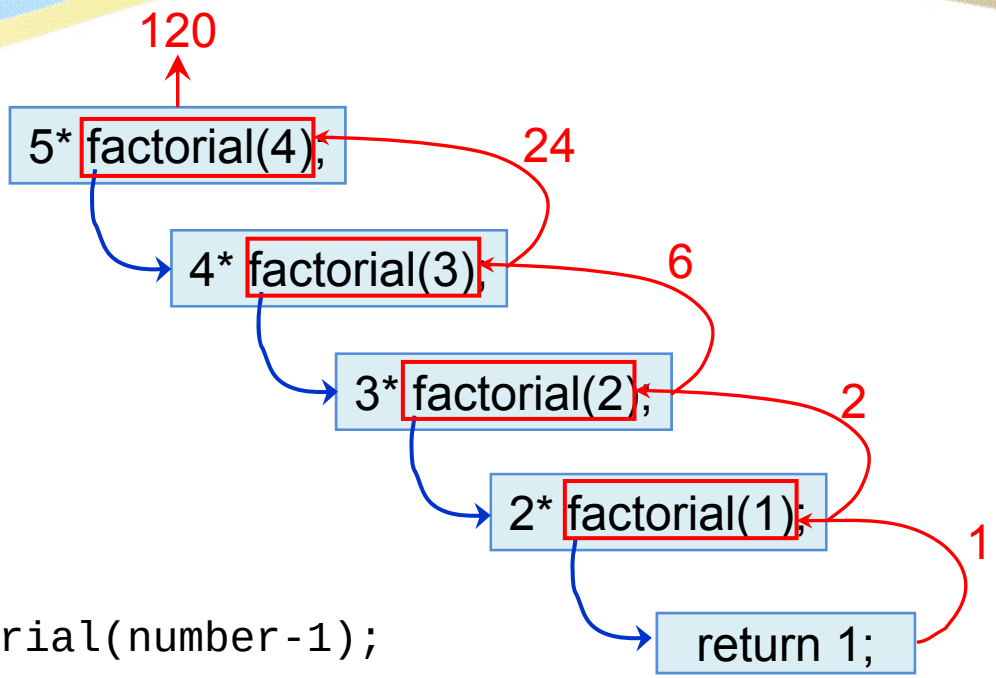
```
int max(int a, int b = 0, int c = 0) {
    if(a>b)
        if(a>c) return a;
        else    return c;
    else
        if(b>c) return b;
        else    return c;
}

int main(void)
{
    cout << max(5) << endl;
    cout << max(5,6) << endl;
    system("PAUSE");
    return 0;
}
```

預設參數不得在中間

```
int max(int a, int b=0, int c)
```

函式 – 遞迴 (Recursion)

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

// 遞迴階層函數
int factorial(int number )
{
    if(number <= 1)
        return 1;
    else
        return number * factorial(number-1);
}

main()
{
    factorial(5);//=5*4*3*2*1=120
    system("PAUSE");
    return 0;
}
```

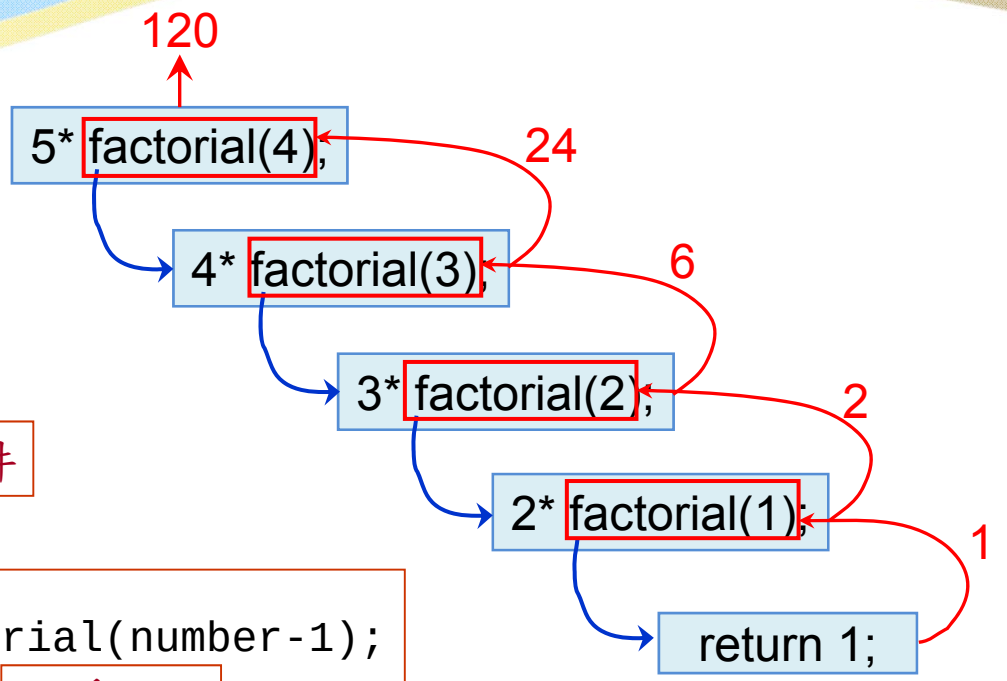


函式 – 遞迴

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

// 遞迴階層函數
int factorial(int number )
{
    if(number <= 1)
        return 1;
    else
        return number * factorial(number-1);
}

main()
{
    factorial(5);//=5*4*3*2*1=120
    system("PAUSE");
    return 0;
}
```

終止條件

遞迴



練習 1

Fibonacci numbers： 1, 1, 2, 3, 5, 8, 13, 21, 34, 55,...

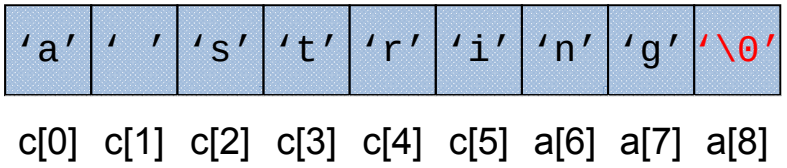
Fibonacci 數列是由{1,1}數字開始，之後每個數字為前兩個數字之合，嘗試以程式計算Fibonacci 數列。

1.利用for迴圈計算第12個Fibonacci數列的數字。

2.利用遞迴函式計算第12個Fibonacci數列的數字。

3.思考一下，兩個方法的計算效率。

C-style字串



C-style：

```
int main(void) {
    char* a   = "a string";
    char  b[] = "a string";
    char  c[] = {'a',' ','s','t','r','i','n','g','\0'};

    int length = strlen(c);
    int compare = strcmp(a, b);

    system("PAUSE");
    return 0;
}
```

長度

比較

compare 數值所代表的意義：
compare>0： lexicographically greater → a > b
compare<0： lexicographically less → a < b
compare==0： a == b

C-style字串

C-style：

```
int main(void) {
    char* str1 = "a str";
    char* str2 = "ing";
    char* str3 = strcat(str1, str2); //str3 is "a string"

    char* str1 = "a string";
    char* str2 = "copy";
    strcpy(str2, str1); //str2 become "a string"

    system("PAUSE");
    return 0;
}
```

結合

複製

C++字串

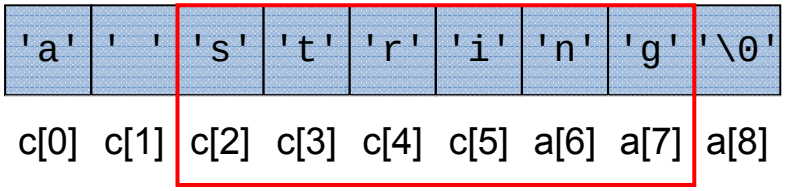
VS已經將string包含至外部相依的文件庫中，只要有使用std即可直接使用，不再需要 #include <string>。

C++ style：

```
#include <string>

stint main(void) {
    string  str1 = "a string";
    string  str2("a string");
    string  str3(str2, 2, 8); //str3 is "string"

    int length = str3.length();
    str1.compare(str2);
    str1.compare("word");

    system("PAUSE");
    return 0;
}
```

長度

比較



C++字串

C++ style：

```
#include <string>

stint main(void) {
    char* str1 = "a";
    char* str2 = "string";
    string  str3 = str1 + str2; //"a string"
    string  str4 = "a " + str2; //"a string"

    char* str5 = str4.c_str();
    cout << str4.c_str() << endl;

    system("PAUSE");
    return 0;
}
```

結合

轉換成c-style

轉換不同型態至字串

VS 2010 後之版本

引入標頭檔

```
#include <string>
```

string 轉型函數

```
string = to_string( _val)
double = stod( _Str)
int = stoi( _Str)
```

注意，此方法沒有安全檢查，請確保字串為正確的"數字字串"。

```
#include <iostream>
#include <sstream>
using namespace std;
int main(void) {
    double d = 23.7;
    string str = to_string(long double(d)); // VS 2010: 浮點數至字串
    string str = to_string(d); //VS2013
    double dd = stod(str);                  // 字串至浮點數

    int i = 13;
    str = to_string(long long(i));          // VS 2010: 整數至字串
    str = to_string(i); //VS2013
    int ii = stoi(str);                     // 字串至整數

    system("PAUSE");
}
```

補充語法 判斷是否為數字

適用任何版本

```
#include <iostream>
#include <sstream>
using namespace std;
int main(void) {
    //string s = "is 12 loh";
    //string s = "12 loh";
    string s = "12";
    char * p ;
    int x = strtol(s.c_str(), &p, 10) ;
    if(*p == 0) {
        cout << " 是數字 " <<endl;
    }
    else {
        cout << " 不是數字 " <<endl;
    }
    cout << " x = " << x <<endl;
    system("PAUSE");
}
```

string 轉型函數 (含判斷)

```
int = strtol(s.c_str(), &p, 10)
```

```
double = strtod(s.c_str(), &p)
```

轉換不同型態至字串

適用任何版本

int 與 string 互轉

```
#include <iostream>
#include <sstream>
using namespace std;
int main(void) {
    string str;
    int i = 100;
    stringstream  i2str;
    //將int轉成string
    i2str << i;      // 將數字100 輸入i2str
    i2str >> str;    // 將數字100 輸出至str
    cout << str << endl ;

    int j = 0;
    //將string轉成int
    stringstream  str2i;
    str2i << str; // 將字串"100" 輸入str2i
    str2i >> j;   // 將字串"100" 輸出至 j
    cout << j << endl ;

    system("PAUSE");
}
```

引入標頭檔

```
#include <sstream>
```

使用sstream物件

```
stringstream  sstr;
```

轉換不同型態至字串

適用任何版本

double 與 string 互轉

```
#include <iostream>
#include <sstream>
using namespace std;
int main(void) {
    string str;
    double d = 23.7;
    stringstream  d2str;
    //將int轉成string
    d2str << d;      // 將數字23.7 輸入i2str
    d2str >> str;    // 將數字23.7 輸出至str
    cout << str << endl ;

    double f = 0;
    //將string轉成int
    stringstream  str2f;
    str2f << str; // 將字串"23.7" 輸入str2i
    str2f >> f;   // 將字串"23.7" 輸出至 f
    cout << f << endl ;

    system("PAUSE");
}
```

引入標頭檔

```
#include <sstream>
```

使用sstream物件

```
stringstream  sstr;
```

轉換不同型態至字串

重複使用stringstream物件

```
// ......
int main(void) {
    string str;
    double d = 23.7;
    stringstream  coverter;
    //將int轉成string
    coverter << d;      // 將數字23.7 輸入
    coverter >> str;    // 將數字23.7 輸出
    cout << str << endl ;

    coverter.str("");
    coverter.clear();

    double f = 0;
    //將string轉成int
    coverter << str; // 將字串"23.7" 輸入
    coverter >> f;   // 將字串"23.7" 輸出
    cout << f << endl ;

    system("PAUSE");
}
```

清空stringstream物件

引入標頭檔

```
#include <sstream>
```

使用sstream物件

```
.str("input");
```

```
.clear();
```

串接多種型態至字串

引入標頭檔

```
#include <sstream>
```

使用sstream物件

```
stringstream  sstr;
```

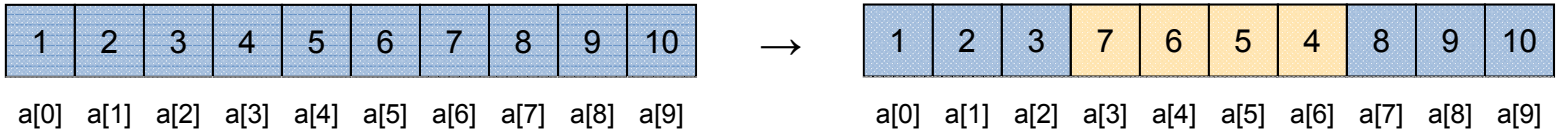
```
#include <iostream>
#include <sstream>
using namespace std;
int main(void) {
    //串接字串 & 整數 & 浮點數

    //方法1: 轉型成字串
    string filename = "test_data_" + to_string(long long(100))
                      + "_" +  to_string(long double(20.7));
    cout << filename <<endl;

    //方法2: 用sstream物件，以字串流形式串接
    stringstream sstr_fname;
    sstr_fname << "test_data_" << 100 << "_" << 20.7 ;
    cout << sstr_fname.str() << endl;

    system("PAUSE");
}
```

練習 2

1.假設有一個包含10個元素的陣列a[10]，嘗試在不需其他陣列的幫助下，將這個陣列元素反轉。

2.嘗試在不需其他陣列的幫助下，將陣列中a[2]至a[7]的元素反轉。

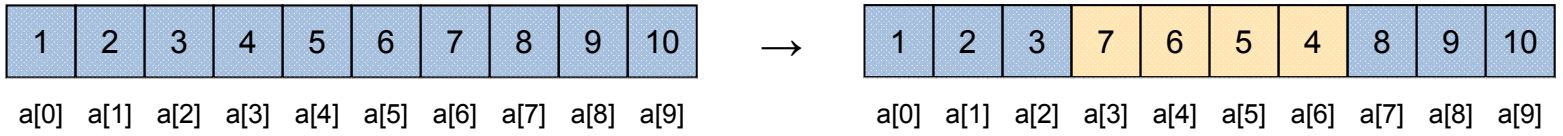
3.隨機產生兩個介於0-9的變數r1與r2，且此兩個變不可相同，嘗試在不需其他陣列的幫助下，將陣列中a[r1]至a[r2]的元素反轉。



作業 – TSP問題: 2-opt

延續練習3，隨機產生兩個介於0-9的變數r1與r2，且此兩個變不可相同，嘗試在不需其他陣列的幫助下，將陣列中a[r1]至a[r2]的元素反轉。但在作業中需選若選擇的r1與r2使得 | r1-r2 |>|10- (r1-r2) |，則反轉r1與r2外的陣列範圍，如圖所示。

case1: | r1-r2 |<=|10- (r1-r2) |



case2: | r1-r2 |>|10- (r1-r2) |