2d-array usually to store rectangeular table of data

base type **name** [row cap][colum cap]

↑ Model declaration

eg: int x [100][20]

string names [2][2] = {{"Freddy", "kelly"}, {"Jack", "Arthur"}}

You specify values row by row.

both rows and cols are **French counted**

2d-arrays and functions

Q1 How do we tell C++ to use a whole 2 array as an input argument for a function?

A Just supply the array name, no [][] at all.

Advice you should always also plan to supply #rows & #cols, too.

```
int main ( ) {
    int x[3][4];
    readarray(x, 3, 4);
    printarray(x, 3, 4);
    return 0;
}
```

Q2 When we make the function. How do we specify in the title line that one of the paras is a 2d array?

A You set up that para with an array declaration that omits the row count but must supply colour count

↑ this limits the use of your function

Goal Make Two Functions

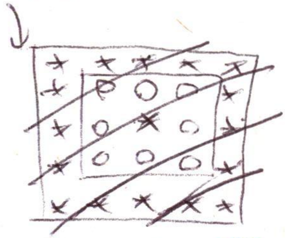
```
void printarray ( int x[][4], int rows, int cols)
    for ( int i = 0 ; i < rows ; i++) {
        for (int j = 0 ; j < cols ; j++)
            Cout << x[i][j] << " ";
        cout << endl;
    }
}
```

```
void readarray ( int x[][4], int rows, int cols ) {
    for ( int r = 0 ; r < rows , r++ ) {
        for ( int c = 0 ; c < cols; c++ ) {
            cout << "Give # in row " << r+1 << " and col " << c+1;
            cin >> x[r][c];
        }
    }
```

wave pattern example.

char pic [41][41];

画图对~



print 没有问题

主要是 makeWave

```
void makeWave (char picture [][41], int cap, int m,
               int w )
    char waveChar;
    if (w%2 == 0)   waveChar = 'x';
    else   waveChar = ' ';
    for (int x = m - w ; x <= m+w ; x++) {
        picture[m-w][x] = picture[m+w][x]
        = picture[x][m-w] = picture[x][m+w] = waveChar;
    }
```

虽然 make wave 方法里未用
但因为是 2D-array
所以必须有.

```
int main() {
    char picture [41][41];
    for ( int w=0 ; w<=20 ; w++ )
        makeWave( picture, 41, 20, w );
    print ( picture , 41 , 41 );
    return 0;
}
```

Final Tues 12-23-14 8:30 am - 10:30 am. Ky170.

Strings & characters

- Type char. stores a single character, always go in single ''

  char x = 'y'; ~~char x = "x"~~; 写了就不行了.

- Type string stores a line of characters (so human text) which in C++ is written in " "

  "word", "x", "Queens College", "" ← empty string

Rules for char data

C++ stores every single character as an integer using ASCII code.

	'0'	'1'	'2'	'3'	...	'9'	'A'	'B'	...	'Z'	'a'	'b'	...	'z'
ASCII	48	49	50			57	65	66		90	97	98		122

upper case letters form a continous sequence so do the lower case ones.

You can do arithmetic with char variables.

```
char x = 'A';
cout << x << x+1 << (char) x+1 << endl;
```

output: A 66 B

```
int main() {
    char word[5] = {'H','e','l','l','o'};
    for (int i=0; i<=4; i++) cout << upper(word[i]) << endl;
    return 0;
}
```

H E L L O

bad code:

```
char upper (char c) {
    if (c < 97 || c > 122) return c;
    return c - 32;
}
```

better code:

```
char upper( char c ) {
    if (c < 'a' || c > 'z') return c;
    return c - ('a' - 'A');
}
```

string 自带 function toupper()

```
int main() {
    for (int i=0; i<=127; i++)
        cout << i << "codes" <<
            (char) i << endl;
    return 0;
}
```

ASCII code.

A string is a line of characters — important.

C++ provides 2 ways to handle strings.

- modern C++ string: type string — convenient, clear, and it's a class. <string>

  C++ types

  fundamental type → int, char, double, bool

  class → string

  other classes can be made by programmers to represent any imaginable type of data

- old style c string: store a string as char[] — Don't use.

If a variable has a class as its type it allows special functions known as methods that do important tasks for the class. These functions are called on a little differently

[var name] . [method name] ([read data])

```
queensCollegeStudent freddy;
cout << freddy.getGPA();
freddy.changeGrade("cs111", "A");
```

to use string data.
we need to know. what. methods are available.

```
string state = "NY";
cout << state.length() << endl;          2
cout << state.substring(1) << endl;      Y
   state.insert(1, "ew␣"); 
   state.insert(5, "ork");
```

在 position 1 插入 "ew␣"

New␣York
1 5

either. class string (C++ string)

or. char[] C-strings need #include <cstring>

The class string has a lot of useful convienent methods:

including .length(), .find(target), .rfind(target)
.substr(index), .insert(index, addition) +

```
string s = "Hello";
cout << s.length(); → 5
cout << s.find("ll"); → 2.
cout << s.find("l"); → 2.
cout << s.rfind("l"); → 3.
cout << s.rfind("ll"); → 2.
cout << s.find("x"); → -1
```

```
cout << s.substr(2); llo
cout << s.substr(2, 2); ll
cout << s.substr(0, 4); Hell
cout << s.substr(s.rfind("l")); lo
```

firstDigit()
first digit()

都可以 尽量用 lowercase.

```
string t = "NY";
t.insert(1, "ew ");
Cout << t; → New Y
t = t + "ork";
Cout << t;   New York.
string t = "New Y";
t.insert(5, "ork");
   or   t.length().
t = t + "ork";
```

C-string has type char[]

```
int main() {
    char s[] = "Hello";   →
    cout << s; → Hello
    cout << s[2]; → l
    s[2] = 'L';        HeLlo
    string s = "hello";
    cout << s[2] → l
    s[0] = 'H';
    cout << s; Hello
```

Q: What is stored in computer?

| 'H' | 'e' | 'l' | 'l' | 'o' | '\0' |

must mark the end of a c-string

```
    char s[3] = "Hello"; //is a bug buffer overflow.
    cout << strlen(s); 5
    strcat(s, "Hello");        HelloHello
```

standard functions. do things to c-string
strlen gives the length
strcat adds to the end

```
    char s[1000] = "Hello";
    strcat(s, "Hello");
    strcpy(t, s);
    strstr(s, t); // finds where t is in s.
```

#include <cstring>

```
    cout << strstr
    char s[] = "Hello";
    char t[] = "llo";
    cout << strstr(s, t) - s; → 2.
```

uses ~~pointer~~ arithmetic
llo
llo 在 s 里的位置.

to use string data.
we need to know what methods are available.

```
string state = "NY";
cout << state.length() << endl;        2
cout << state.substr(1) << endl;       Y
  state.insert(1, "ew "); 在 position 1 插入 "ew "
  state.insert(5, "ork");              New York
                                       1   5
```

either class string (C++ string)
or char[] C-strings need #include <cstring>
The class string has a lot of useful convienent methods:
including .length(), .find(target), .rfind(target)
.substr(index), .insert(index, addition) +

```
string s = "Hello";     (01234)
cout << s.length(); → 5          cout << s.substr(2); llo
cout << s.find("ll"); → 2.       cout << s.substr(2,2); → ll
cout << s.find("l"); → 2.        cout << s.substr(0,4); → Hell
cout << s.rfind("l"); → 3.       cout << s.substr(s.rfind("l")); lo
cout << s.rfind("ll"); → 2.          firstDigit()
cout << s.find("x"); → -1            first digit()   都3:尽量用 lowercase.
```

```
string t = "NY";
t.insert(1, "ew ");
Cout << t; → New Y
t = t + "ork";
Cout << t;   New York.
string t = "New Y";    (01234)
t.insert(5, "ork");
         t.length().
or
t = t + "ork";
```

C-string has type char[]

```
int main() {
    char s[] = "Hello";  →  Q: What is stored in computer?
                            | 'H' | 'e' | 'l' | 'l' | 'o' | '\0' |
                            must mark the end of a c-string
    cout << s; → Hello
    cout << s[2]; → l
    s[2] = 'L';          HeLlo
    string s = "hello";   (01234)
    cout << s[2] → l
    s[0] = 'H';
    cout << s;  Hello

    char s[5] = "Hello";  // is a bug buffer overflow.
    cout << strlen(s); 5     standard functions. do things to
HelloHello                   c-string
  ↓ strcat(s, "Hello");      strlen gives the length
                             strcat adds to the end

    char s[1000] = "Hello";
    strcat(s, "Hello");
    strcpy(t, s);
#include  strstr(s, t);  // finds where t is in s.
<cstring> cout << strstr
      char s[] = "Hello";
      char t[] = "llo";
      cout << strstr(s, t) - s; → 2.
            uses pointer arithmetic
            ↓
            llo        llo在s里的位置.
```

Files

for real life input & output we often prefer to use files instead of to screen.

C++ provides two classes:

ofstream & ifstream that look after either an output file or an input file.

goal C++ should write Hello to a file output.txt.

```
#include <fstream>
using namespace std;
int main() {
    ofstream f;
    f.open("output.txt");
    f << "Hello" << endl;
    f.close();
    return 0;
}
```

setup a variable for any file you plan to use.

connect your variable to a real life file.

variable f now be used in same way we use cout.

goal write Hello to a file out1.txt and Goodbye to 2nd file c:\out2.txt.

```
#include <fstream>
using namespace std;
int main() {
    ofstream f1, f2;
    f1.open("out1.txt");
    f2.open("c:\\out2.txt");
    f1 << "Hello" << endl;
    f2 << "Goodbye" << endl;
    f1.close();
    f2.close();
    return 0;
}
```

stupid trick \\ in a string means \

goal input files work like cin

read first word in file "input.txt"

```
#include <iostream>
#include <fstream>
using namespace std;
int main() {
    ifstream f;
    f.open("input.txt");
    if (f==0) {
        cout << "No such a file!\n";
        return 0;
    }
    string word;
    f >> word;
    cout << "Your file starts with " << word;
    cout << endl;
    f.close();
    return 0;
}
```

goal ask user for names of input & output and it copies input to output.

Two problems — How do we know?

while (there's more) input, read it in & print it out.

f.eof() is a method says true when reach end of file.

f.open("....")

open method works with a string.

this is a c-string unfortunately.

```
# include <iostream>
# include <fstream>
using namespace std;
main ( ) {
    string inName, outName;
    cout << "What are the input & output files?";
    cin >> inName >> outName;
    ifstream fIn;
    ofstream fOut;
    fIn.open( inName.c_str( ) );
    fOut.open( outName.c_str( ));
    if ( fIn==0 || fOut==0 ) return 0;
    while (!fIn.eof( )) {
        char x;  <--- Don't want to miss space
        x = fIn.get( );
        fOut << x;
    }
    fIn.close( );
    fOut.close( );
    return 0;
}
```

下划线 → method returns a c-string copy of a C++ string

venus> cp fileA fileB

Goal) modify our copying ~~probl~~ program so we can work as follows

venus> g++ copy.cpp

array of c-strings: venus> ./a.out (#0) fileA (#1) fileB (#2)

argc: | ./a.out | fileA | fileB |

argc = 3

To do this, our main task need input.

The main task can have a second possible ~~the~~ title to allow for input.

```
int main( int argc, char* argv[ ] )
```

↑ argc: Counts the # of pieces of input

```
int main ( int argc, char* argv[ ] ) {
    ...
    fIn.open( argv[1] );
    fOut.open( argv[2] );
```

~~... a Illegal Exception class for Date12~~

作业里有个问题要注意:

```
int gapSum( int x[], int cap) {
    int sum = 0;
    for( int i=1; i<cap; i++) {
        if (x[i] > x[i-1] )
            sum += x[i] - x[i-1];
        else sum += x[i-1] - x[i];
    }
    return sum;
}
```

☆ 如果是 [i+1]>[i] 记得 array index out of bound issue