Nested Loop, Arrays

Tai, Wei Hsuan

week 5

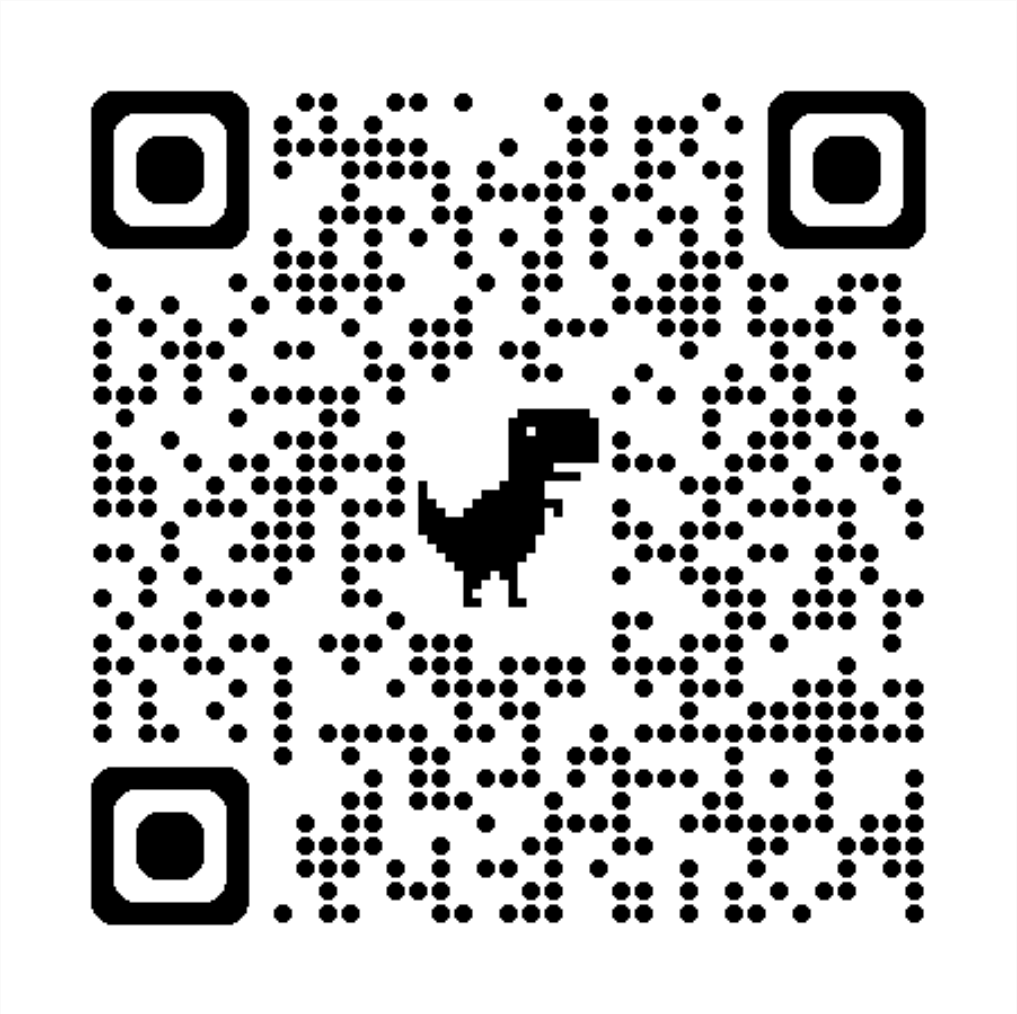
Announcement

Recording or photographing the slides or any content displayed on the screen is permitted for personal use only.

Redistribution, modification, or any commercial use of the materials is strictly prohibited.

© 2025 [Tai, Wei Hsuan]. All rights reserved. For personal use only.

Course Materials



進度安排

日期	主題	日期	主題
9/12	課程簡介、基礎輸入輸出、變數	11/28	函式
9/19	變數的運算	12/5	遞迴
10/17	選擇結構與邏輯運算子	12/12	Struct, Vector
10/31	重複結構	12/19	Stack, Queue
11/7	巢狀迴圈、陣列	12/26	TBD
11/21	期中考	1/2	TBD
		1/9	期末考

Outline

1. Recap

2. Nested Loop

3. Arrays

Recap

Basic Structure

```
1  #include<bits/stdc++.h>
2  using namespace std;
3
4  int main(){
5      // code here
6      return 0;
7  }
```

Variables and basic I/O

```
1  #include<bits/stdc++.h>
2  using namespace std;
3
4  int main(){
5      int a;
6      cin >>a;
7      cout<<"The value of a is: "<<a<<'\n';
8  }
```

Selection Statements

```
1            if(condition){
2                // code when condition is true
3            }else if(condition2){
4                // code when condition2 is true
5            }else{
6                // code when both conditions are false
7            }
```

for loop

How to utilize `for` loop to print odd numbers from 1 to n?

```
1               int n;
2               cin >>n;
3               for(int i=1;i<=n;i++){
4                   if(i%2==1){
5                       cout<<i<<'\n';
6                   }
7               }
```

for loop(cont.)

You can also:

```
1                int n;
2                cin >>n;
3                for(int i=1;i<=n;i+=2){
4                    cout<<i<<'\n';
5                }
```

Question: Can you use `continue` to achieve the same effect?

while loop

```
1               int i=1, n;
2               cin >>n;
3               while(i<=n){
4                   cout<<i<<'\n';
5                   i+=2;
6               }
```

Prime Number Detection

Given a positive integer n, how to determine whether it is a prime number?

It is well known that a prime number is a natural number greater than 1 that cannot be formed by multiplying two smaller natural numbers. In other words, a prime number has exactly two distinct positive divisors: 1 and itself.

Thus, we can just check whether n is divisible by any integer from 2 to n-1.

Implementation

```
1            int n;
2            cin >>n;
3            bool is_prime=true;
4            for(int i=2;i<n;i++){
5                if(n%i==0){
6                    is_prime=false;
7                    break;
8                }
9            }
10           if(is_prime)cout<<n<<" is a prime number.\n";
11           else cout<<n<<" is not a prime number.\n";
```

The above implementation has a small error. Can you spot it?

Little Error

What if the input n is 1? By definition, 1 is not a prime number. However, the above code will output that 1 is a prime number. To fix this, we can add a condition to check if n is less than or equal to 1.

```
int n;
cin >>n;
bool is_prime=true;
if(n<=1)is_prime=false;
for(int i=2;i<n;i++){
    if(n%i==0){
        is_prime=false;
        break;
    }
}
if(is_prime)cout<<n<<" is a prime number.\n";
else cout<<n<<" is not a prime number.\n";
```

Optimization

A computer can do about 10^8 operations per second. If n is very large (e.g., 10^12), the above algorithm may take too long to run.

Note that if n is divisible by a number larger than $\sqrt{n}$, the corresponding divisor must be smaller than $\sqrt{n}$. Therefore, we only need to check for divisibility up to $\sqrt{n}$.

With this optimization, we can efficiently determine the primality of large numbers.

Implementation

```
1           int n;
2           cin >>n;
3           bool is_prime=true;
4           if(n<=1)is_prime=false;
5           for(int i=2;i*i<=n;i++){
6               if(n%i==0){
7                   is_prime=false;
8                   break;
9               }
10          }
11          if(is_prime)cout<<n<<" is a prime number.\n";
12          else cout<<n<<" is not a prime number.\n";
```

Nested Loop

Nested Loop

The main idea is, anything can be inserted into a loop, including another loop. If we want to do some loop repeatedly, we can use nested loops.

For instance, how to print a 5×4 rectangle of asterisks(*)? Note that there are 5 rows and 4 columns.

Implementation

```
1            for(int i=1;i<=5;i++){
2                for(int j=1;j<=4;j++){
3                    cout<<"*";
4                }
5                cout<<'\n';
6            }
```

Another example

Note that in the inner loop, the veiables defined in the outer loop are also accessible. Thus, we can use nested loops to print a multiplication table.

```
1             for(int i=1;i<=9;i++){
2                 for(int j=1;j<=9;j++){
3                     cout<<i<<"x"<<j<<"="<<i*j<<'\t';
4                 }
5                 cout<<'\n';
6             }
```

Practice

ZeroJudge Problem c418: c418. Bert 的三角形 (1)

Arrays

Basic Concept

If you regard variables as boxes that can store a single value, then an array is like a row of boxes that can store multiple values of the same type. Each box in the array can be accessed using an index, which indicates its position in the array. In fact, arrays are contiguous memory locations that store multiple values of the same type.

Declare an Array

In the previous page, we know that an array is a collection of variables of the same type. To declare an array, we need to specify the type of the elements and the number of elements in the array. The syntax is similar to variables declaration, as follows:`TYPE IDENTIFIER[SIZE]`

For instance, `int a[10]` declares an array of 10 integers.

Usage of Arrays

You can consider each element in the array as a separate variable. You can access and modify each element using its index. Note that the index starts from 0. For example, to access the first element of the array `a`, you can use `a[0]`. To access the second element, you can use `a[1]`, and so on.

Some examples

```
1               int a[5];
2               for(int i=0;i<5;++i){
3                   cin>>a[i];
4               }
5               for(int i=0;i<5;++i){
6                   cout<<a[i]<<" ";
7               }
```

String

Sometimes, the format of the input is a string of characters. In C++, we can use a character array to store a string. However, a character array is inconvenient to use. Thus, C++ provides a built-in string type that is easier to use. Here is an example of using a string:

```
1           string s;
2           cin >>s;
3           cout<<"The input string is: "<<s<<'\n';
4           cout<<"The length of the string is: "<<s.size()<<'\n';
5           cout<<"The first character of the string is:
             ↪  "<<s[0]<<'\n';
```

String(cont.)

Note that the index of a string also starts from 0. Thus, the first character of the string is at index 0, the second character is at index 1, and so on. You can also use a loop to iterate through each character in the string.

You can consider a string as a flexible array of characters that can grow or shrink in size as needed. The usage of strings is similar to arrays, but strings have additional functionalities that make them easier to work with. If you truly interest in string, you can refer to C++ String Reference for more details.

Practice

- Given n numbers, list them in reverse order.
- Given n numbers, list the frequency of each number.
- The formula of standard deviation is:$SD = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)^2}$, where μ is the mean of the n numbers. Given n numbers, calculate the standard deviation.