头文件的作用：

1.#include<iostream>: cout cin

2.#include<cmath>:pow sqrt exp

3.#include<time.h>: time(0) time(NULL) timeDelay(int sec)

4.#include<string>: create string type

5.#include<cstdlib>: rand()

6.using namespace std;

everything on this list is in the std namespace.

e.g. std::cout std::string

算术运算符：

y+=x	y=y+x
x-=5	x=x-5
x/=y	x=x/y
price*=unit+1	price=price* (unit+1)
(price*=unit) ++	

% modulo 不可以用于计算浮点数

Example: 8%5 = 3

INCREMENT AND DECREMENT OPERATOR

x=3			
	x	y	x changes first
y=++x	4	4	y=(x=x+1)
y=--x	2	2	
x=3			
	x	y	y changes first
y=x++	4	3	x=(y=x)+1
y=x--	2	3	

Ex: x=3;

cout<<++x; 4 5 6 7

cout<<x++; 3 4 5 6

DATA TYPE

1. void: empty function type, no return
2. bool: true or false
3.char: a letter or text number
4. int: integral
5.float: single precision
5.double: double precision (8 bytes)

<u>char<int<bool<float<double</u>

```
int a = 4;
float b = 4.653;
double c = 1283.3829832;
char d = 'L';
char e = '3';
bool f = TRUE;
bool g = 0;
const int special = 147;
```

1.const: constant

2. unsigned: positive

3. signed: positive or negative

4.[#]: a pointer to an array address e.g. a(3)={3 2 1}; a[1]=3

5.string: text message

6. vector<>: special array class

```
string h = "Hello";                    Need #include <string>
string i = "This is also a string.";
```

<u>string 储存的字节数比实际储存的多一位" \0"</u>

※变量命名规则

不能以数字开头，可以是数字或_开头；

不能包含 %, ?, +,-,},#等运算符

不能包含 if, while, double, int, etc.

int x = 0; [initialized at definition] **变量初始化**

int x(0); [initialized at definition] **构造函数初始化**

int x{0}; [initialized at definition] **统一初始化**

※***rand()***

- Need **#include <cstdlib>** and **#include <ctime>**

Int x=rand() %3//在 0 到 2 之间的随机数

Switch **if-else**

Switch-Case

```
switch (myVar) {
case 1:
    //do one
    break;
case 2:
    //do two
    break;
default:
    //do other
    break;
}
```

If-Else if-Else

```
if (conditionOne) {
    //do condition 1
}
else if (conditionTwo) {
    //do condition 2
}
else {
    //do other condition
}
```

Loop

For Loop

```
for (int i = 0; i < 10; i++) {
  //repeat something
}
```

While Loop

```
while (condition) {
  //repeat something
}
```

Recursion Loop

```
void recursive() {
  if (condition) { return; }
  recursive();
}
```

Do While Loop

```
do {
  //repeat something
} while (condition);
```

1. break; : stop the loop and exit the loop
2. continue; : go back to the begining at the loop
3. return; :in int main this end the program; end the function
4. exit(0); :in int main this end the program; end the program

Create a function that uses all three of these:

1. prototype declaration（原型）

定义写在 int main 后面

2. call statement

3. definition

```
double spiderman (int web)
int main(){
    double petter=spiderman(3)
}
double spiderman(int web){
    return web*100;
}
```

Also,1. Give the function an output return

2. Allow the input variable to be changed in the function

```
double spiderman (int& web)
// &取得的是web的地址
int main(){
    int web=3;
    double petter=spiderman(3)
}
double spiderman(int& web){
    web* =100;
    //*表示取得web地址中的值
    return web*100;
}
//output petter=10000
```

Also, give the same name and different input types

```
double spiderman (int& web)
void spiderman (double& petter,int& web);
// &取得的是web的地址
int main(){
    int web=3;
    double petter=spiderman(3);
    spiderman(petter,web);
}
double spiderman(int& web){
    web* =100;
    //*表示取得web地址中的值
    return web*100;
}
void spiderman(double& petter,int& web){
    // do some thing
}
```

give the input variables default values

```
double spiderman(int web=3){
    web* =100;
    //*表示取得web地址中的值
    return web;
}
int main(){
    double petter=spiderman();
    //在输入为空的时候会默认输入web=3
    cout << petter;
}
```

Arrays

Create: int A [5]; (type+name+size)

Add elements: delate and recreate

int A [5] = {1,2,3,4,5};

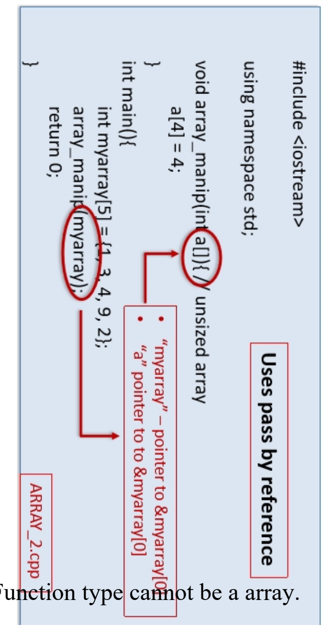
int A [5] = {0};

Access elements: A [#] number

int A [0] = 1; //编号从 0 开始

Fine length: sizeof(A)/sizeof(double)

Passing an Array into a Function:

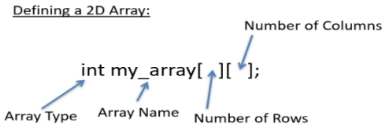


```
#include <iostream>
using namespace std;
void array_manip(int a[]){ // unsized array
    a[4] = 4;
}
int main(){
    int myarray[5] = {1, 3, 4, 9, 2};
    array_manip(myarray);
    return 0;
}
```

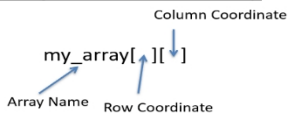
Uses pass by reference
"myarray" – pointer to &myarray[0]
"a" pointer to to &myarray[0]
ARRAY_2.cpp

Function type cannot be a array.

<u>Defining a 2D Array:</u>



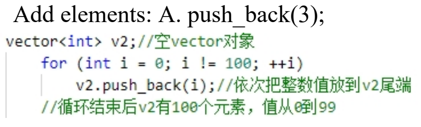
<u>Referencing a Position within the 2D Array:</u>



Vectors

Create: vector<int>A(5,0);

Add elements: A. push_back(3);



```
vector<int> v2;//空vector对象
    for (int i = 0; i != 100; ++i)
        v2.push_back(i);//依次把整数值放到v2尾端
    //循环结束后v2有100个元素，值从0到99
```

//编号从 0 开始

Access elements: A[1];

Fine length: A.size()

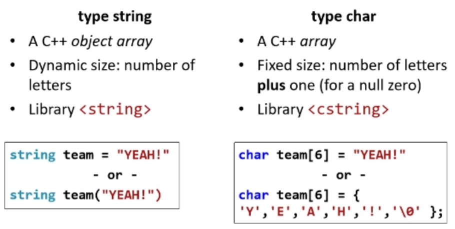
Passing an Array into a Function:

double QWQ (vector<int>A);

vector can be a type of function:

vector<int>QWQ(…){//*   *//}

String



type string	type char
• A C++ *object array*	• A C++ *array*
• Dynamic size: number of letters	• Fixed size: number of letters **plus** one (for a null zero)
• Library `<string>`	• Library `<cstring>`

```
string team = "YEAH!"
     - or -
string team("YEAH!")
```

```
char team[6] = "YEAH!"
     - or -
char team[6] = {
'Y','E','A','H','!','\0' };
```

Pointers (concept only)

指针 int*amw = &alex;

Pointer(use *)is a second variable name that stores as its value the address of another variable.

Address(use &) is the location in computer memory where the variable is stored

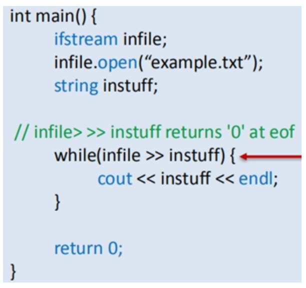
File input and output

#include <fstream> // input, output, input & output

• ifstream–taking input from a file.

• ofstream–giving output to a file.

• fstream–both input and output.



```
int main() {
        ifstream infile;
        infile.open("example.txt");
        string instuff;

  // infile>>instuff returns '0' at eof
        while(infile >> instuff) {
                cout << instuff << endl;
        }

        return 0;
}
```

ifstream inFile;

string myString;

getline(inFile, myString)

inFile>>myString;

This

The **<u>this</u>** pointer is an implicit parameter to all member functions. Therefore, inside a member function, this may be used to refer to the invoking object.

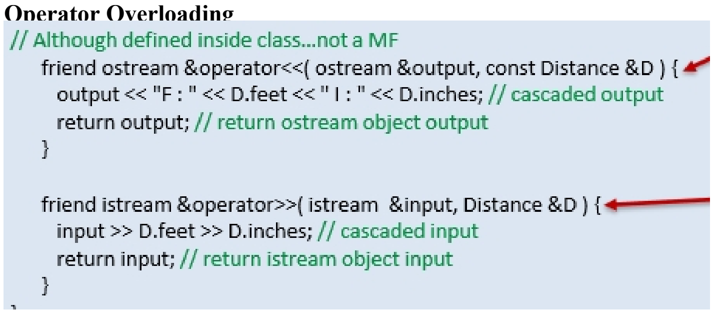
Friend

A **<u>friend</u>** function of a class is defined outside that class' scope but it has the right to access all private and protected members of the class.

Even though the prototypes for **<u>friend</u>** functions appear in the class definition, **<u>friends</u>** are not member functions.

A friend can be a function, function template, or member function, or a class or class template, in which case the entire class and all of its members are friends.

Operator Overloading



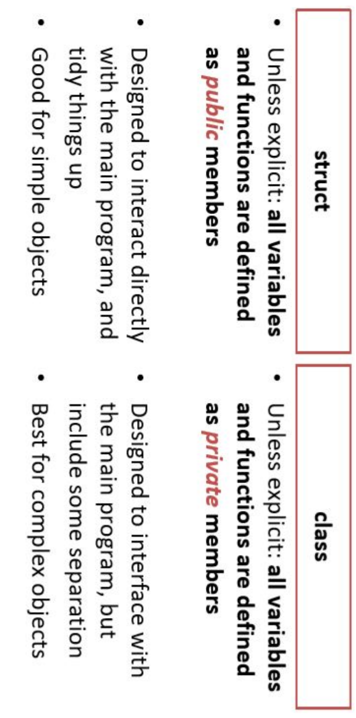
```
// Although defined inside class...not a MF
    friend ostream &operator<<( ostream &output, const Distance &D ) {
      output << "F : " << D.feet << " I : " << D.inches; // cascaded output
      return output; // return ostream object output
    }

    friend istream &operator>>( istream  &input, Distance &D ) {
      input >> D.feet >> D.inches; // cascaded input
      return input; // return istream object input
    }
```

Nested Class and Structs

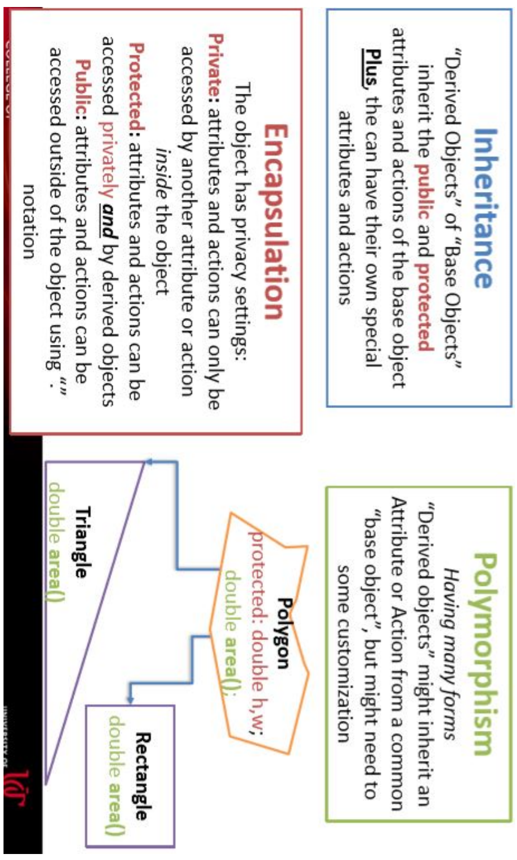
```
void bigClass::setBig(int u_Big)
{ bigVar = u_Big; };

void bigClass::littleClass::setLittle(int u_Little)
{ littleVar = u_Little; };
```

When creating [Nested] Class object [say, in main()]… use full scope resolution syntax:

```
bigClass::littleClass n;
```



struct	class
• Unless explicit: **all variables and functions are defined as *public* members**	• Unless explicit: **all variables and functions are defined as *private* members**
• Designed to interact directly with the main program, and tidy things up	• Designed to interface with the main program, but include some separation
• Good for simple objects	• Best for complex objects



Encapsulation

The object has privacy settings:

Private: attributes and actions can only be accessed by another attribute or action *inside* the object

Protected: attributes and actions can be accessed privately ***and*** by derived objects

Public: attributes and actions can be accessed outside of the object using "." notation

Inheritance

"Derived Objects" of "Base Objects" inherit the **public** and **protected** attributes and actions of the base object

Plus, the can have their own special attributes and actions

Polymorphism

Having many forms

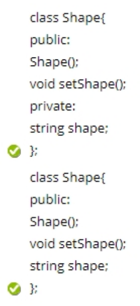
"Derived objects" might inherit an Attribute or Action from a common "base object", but might need to some customization

Namespace

Namespaces are simply an <u>abstract way of grouping items together</u>. A namespace more of a naming convention. They can contain variables and functions.

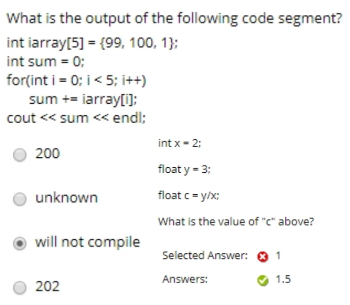
选择：

1. If a function is called more than once in a program, the values stored in the functions local do not persist between function call
2. Rational operators allow you to compare numbers
3. function is a collection of statements that performs a specific task.
4. A function can have zero to many parameters, and it can return these many values. (zero or only one)
5. mary.age
6. What two kind of statements must be followed for recursion to work? 1. Base case 2. Recursive call
7. Recursion is defined as: A function which calls itself.
8. When defining a recursive algorithm, what single property must hold? The 'problem' must get smaller with each call.
9. A recursive algorithm is analogous to what C++ construct which we've already done? Looping constructs - while, for, do-whil
10. An Integrated Development Environment (IDE) typically consists of: A Text Editor A Compiler A debugger
11. When passing a struct variable by value, what happens? A copy of the data(structure) is made.
12. To write to a file define an object of this type. ofstream
13. What is the return type of a class constructor? It doesn't have a return type.
14. This is true! That is all folks!
15. When defining a class implementation outside the class interface, what is the correct syntax? void Shape::setShape(string thing){ ...}
16. Data types that are created by a programmer are known as abstract data types (ADT)
17. Which of the following is not contained inside a program? Program name
18. What is NOT an abstraction? The joining of simple systems into a complex system.
19. The value of this type of local variable persists between function calls static
20. If Circle is a structure tag, the following statement: Circle doSomething(Circle c2)
can be header line for a function that takes a Circle structure as a parameter, does something, and returns a Circle structure.
21. What is the first legal subscript that can be used with the following array?
double array1[10]; 0
22. This is a collection of statements that performs a specific task function
23. Which is valid way to pass the following array to a function?
int array1[9] = {0}; ProcessArray(array1)



```
class Shape{
public:
Shape();
void setShape();
private:
string shape;
};
```
✓

```
class Shape{
public:
Shape();
void setShape();
string shape;
};
```
✓



What is the output of the following code segment?

```
int iarray[5] = {99, 100, 1};
int sum = 0;
for(int i = 0; i < 5; i++)
    sum += iarray[i];
cout << sum << endl;
```

200

unknown

[✓ will not compile]

202

```
int x = 2;
float y = 3;
float c = y/x;
```

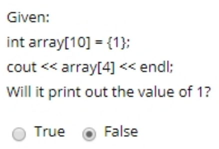
What is the value of "c" above?

Selected Answer: ✗ 1

Answers: ✓ 1.5

判断：

1. The code within a "do while" loop is only executed if the loop termination condition is true. (F)
2. You have to declare every variable in C++? (T)
3. In C++, public and private keywords must be given for every class method and variable. (F)
4. C++ is not a strongly typed language. (F)
5. An expression that has any value other than zero (0) is considered false when evaluating logical or conditional expressions. (F)
6. All loops can be written as a recursive algorithm, and all recursive algorithms can be written as a loop. (T)
7. In C++, public and private keywords must be given for every class method and variable. (F)
8. The default section is required in a switch statement. (F)
9. It can be problematic to use the equality operator to compare floating point values because of the imprecision of how floating point values are stored. (T)
10. Inheritance allows us to define a class in terms of another class, which makes it easier to create and maintain an application. (T)
11. The code within a "do while" loop is only executed if the loop termination condition is true. (F)
12. Encapsulation is a process of wrapping of data and methods in a single unit. It is achieved in C++ language by class concept. (T)
13. A C++ array is automatically initialized to the value of zero? (F)
14. Polymorphism means that a call to a function will cause a different function to be executed depending on the data type of parameters that invokes the function. (T)



Given:

```
int array[10] = {1};
cout << array[4] << endl;
```

Will it print out the value of 1?

True [✓ False]

```cpp
//infile,outfile,namespace

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

namespace OSW {
    double const g = 9.80665;
    double V0,y0,y;
    double Height(int t){
        y = y0 + V0 * t - 0.5 * g * pow(t, 2);
        return y;
    };
     int t;// time
}
using namespace OSW;


int main() {
    using namespace OSW;

    ifstream inFile;
    inFile.open("Lab9A.txt");

    if (!inFile.is_open()) {
        cout << "Fail." << endl;
        return 1;
        }

    //begin date input
    double inputFile;
    inFile >> OSW::V0;
    inFile >> OSW::y0;
    inFile.close();


    ofstream outFile;
    outFile.open("Lab9Aoutput.txt");
    //header information
    outFile << "Initial Velocity of Object: " << V0 << " m/s " << endl;
    outFile << "      Time                          Height" << endl;
    int n = 0;//count number
    double a = 0;//max
    double b = 0;//max time
    double Y;
    Y = OSW::y0;
    cout << "Initial Velocity of Object: " << V0 << " m/s " << endl;
    cout<< " Time                          Height" << endl;
    for (int t = 0; Y > 0; t++) {
        if (t != 0) {
            n++;
        }
        if (t < 10) {
            cout << "      " << t << "      ";
        }
        else {
            cout << "      " << t << "      ";
        }
        Y=Height(t);
        if (Y > 0) {
            cout << Y << endl;
        }
        else {
            Y = 0;
            cout << Y << endl;
        }
        // find the max
        if (Y > a) {
            a = Y;
            b++;
        }
              outFile << t << "\t";
          outFile << Y << endl;

    }
    cout << " Total Time: " << n << "-seconds " << endl;
    cout << "Maximum Height: " << a << " @" << b << " -seconds";

    outFile.close();
    return 0;
}
```

<u>**Pushback，设 vector，string，constant value，引用函数**</u>

```cpp
#include<iostream>
#include<cmath>
#include<vector>
using namespace std;
const double TOLERANCE = .0001;
double polynomial(double x) {
            double y = pow(x, 4) + 2 * pow(x, 3) - 31 * pow(x, 2) - 32 * x + 60;
            return y;
}
double derivative(double x) {
            double y1 = 4 * pow(x, 3) + 6 * pow(x, 2) - 62 * x - 32;
            return y1;
}
double Newton(double x) {
            double x1 = x - polynomial(x) / derivative(x);
            return x1;
}
int main() {
            double x1;
            vector<double>x(1);
            string type = "y";
            int n = 0;
            while (type == "y") {
                        cout << "Enter Guess: ";
                        cin >> x[n];
                        x1 = Newton(x[n]);
                        while (abs(x1 - x[n]) > TOLERANCE) {
                                    if (derivative(x[n]) == 0) {
                                                cout << "error!" << endl;
                                                break;
                                    }
                                    x.push_back(x1);
                                    n++;
                                    x1 = Newton(x[n]);
                        }
                        cout << "Root: " << x1 << endl;
                        cout << "Enter Another Guess: y/n? ";
                        cin >> type;
                        }
            return 0;
}
```

<u>**Struct 用法 struct 套函数**</u>

```cpp
#include<iostream>
using namespace std;
struct Time
{
            // Initial value
            int hour, minute, second;
            Time()
            {
                        hour = 0;
                        minute = 0;
                        second = 0;
            }
            // User
            Time(int user)
            {
                        hour = user / 3600;
                        minute = (user - 3600 * hour) / 60;
                        second = user - 3600 * hour - 60 * minute;
            }
            // To calculate
            int QAQ() {
                        return hour * 3600 + minute * 60 + second;
            }
            // Output statement
            void output() {
                        cout <<    hour << " : " << minute << " : " << second << endl;
            }

};
// Calculate the "half" tiime
Time Orz (int user){
             user=user/2;
             return Time(user);
}
            int main() {
                        int enter;
                        cout << "Time (sec): ";
                        cin >> enter;
                        Time A;
                        Time B(enter);
                        Time C=Orz(enter);
                        cout << "Using the default constructor: ";
                        A.output();
                        cout << "Using the user constructor: ";
                        B.output();
                        cout << "Using the half-time function: ";
                        C.output();
                        cout << "The half-time total seconds: " << C.QAQ() << endl;
                        return 0;
            }
```

get 用法，struct 中函数

```
#include<iostream>
#include<vector>
#include<string>
using namespace std;
class eat
{
private:
    // Initial value
    int rating;
    string name;
public:
    eat() {};
    void restaurant() {
        cout << "restaurant: ";
        cin >> name;
    }
    void rate() {
        cout << "rate: ";
        cin >> rating;
    }
    void print() {
        cout << name << " : " << rating << endl;
    }
    void setrate(const int& ratee) {
        rating = ratee;
    }
    int getrate() {
        return rating;
    }
};
int main()
{
    vector<eat>QAQ(5);
    int rating[3];
    for (int i = 0; i < 5; i++) {
        QAQ[i].restaurant(); QAQ[i].rate();
    }
    cout << "Show which one?" << endl;
    for (int i = 0; i < 3; i++) {
        cin >> rating[i];
    }
    cout << endl;
    cout << "These are the three restaurants that you want to show: " << endl;
    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 5; j++) {
            if (QAQ[j].getrate() == rating[i]) {
                QAQ[j].print();
            }
        }
    }
    return 0;
}
```

namespace 用法 IOf 用法

```
#include<iostream>
#include <fstream>
#include <string>
using namespace std;
namespace biu {
    double g = 9.80665;

    double v0, t, Y;
    double Y0;
    double Height() {
        Y = Y0 + v0 * t - 0.5 * g * t * t;
        return Y;
    }
}
int main() {
    using namespace biu;
    ifstream infile;
    infile.open("input.txt");
    string instuff;
    double max = 0, b = 0;
    infile >> instuff;
    cout << instuff << endl;
    infile >> v0;
    cout << v0 << endl;
    infile >> instuff;
    cout << instuff << endl;
    infile >> Y0;
    cout << Y0 << endl;
    infile.close();
    cout << endl;
    cout << "Time";
    cout << "                              ";
    cout << "Height" << endl;
    ofstream outfile;
    outfile.open("output.txt");
    Y = 1;//Let it run
    for (int i = 1; Y > 0; i++) {
        t = i;
        Height();
        biu::Height();
        if (Y < 0) {
            Y = 0;
        }
        if (Y > max) {
            max = Y;
            b++;

        }
        if (t < 10) {
            outfile << t << "                              " << Y << endl;
            cout << t << "                              " << Y << endl;
        }
        else {
            outfile << t << "                              " << Y << endl;
            cout << t << "                              " << Y << endl;
        }
    }
    outfile.close();
    cout << "                              Total Time: " << t << endl;
    cout << "Maximum Height: " << max << " @ " << b << "-seconds" << endl;
    return 0;
}
```

Operator 用法

```
Class   pika {
public:
    int y, mon, d, h, min, s;
    string temp;
    friend ifstream& operator>>(ifstream& input, pika& T)
    {
        input >> T.y >> T.temp >> T.mon >> T.temp >> T.d >> T.temp >> T.h >>
T.temp >> T.min >> T.temp >> T.s;
        return input;
    }
    friend ostream& operator<<(ostream& output, const pika& T)
    {
        output << "[" << T.y << T.temp << T.mon << T.temp << T.d << T.temp << T.h
<< T.temp << T.min << T.temp << T.s << "]";
        return output;
    }
};
struct haha {
    double y, mon, d, h, min, s;
    haha(pika Time) {
        y = Time.y; mon = Time.mon; d = Time.d; h = Time.h; min = Time.mon; s =
Time.s;
    }
    bool operator ==(const haha& T) {
        if (y == T.y && mon == T.mon && d == T.d && h == T.h && min == T.min
&& s == T.s)
            return 1;
        else
            return 0;
    }
    bool operator <(const haha & T) {
        double avey = 365.25; double avem = 30.4375;
        double s1 = y * avey * 24 * 60 * 60 + mon * avem * 24 * 60 * 60 + d * 24 * 60
* 60 + h * 60 * 60 + min * 60 + s;
        double s2 = T.y * avey * 24 * 60 * 60 + T.mon * avem * 24 * 60 * 60 + T.d *
24 * 60 * 60 + T.h * 60 * 60 + T.min * 60 + T.s;
        if (s1 < s2) { return 1; }
        else { return 0; }
    }
    bool operator >(const haha & T) {
        double avey = 365.25; double avem = 30.4375;
        double s1 = y * avey * 24 * 60 * 60 + mon * avem * 24 * 60 * 60 + d * 24 * 60
* 60 + h * 60 * 60 + min * 60 + s;
        double s2 = T.y * avey * 24 * 60 * 60 + T.mon * avem * 24 * 60 * 60 + T.d *
24 * 60 * 60 + T.h * 60 * 60 + T.min * 60 + T.s;
        if (s1 > s2) { return 1; }
        else { return 0; }
    }
    double operator-(const haha & T) {
        double avey = 365.25; double avem = 30.4375;
        double s1 = y * avey * 24 * 60 * 60 + mon * avem * 24 * 60 * 60 + d * 24 * 60
* 60 + h * 60 * 60 + min * 60 + s;
        double s2 = T.y * avey * 24 * 60 * 60 + T.mon * avem * 24 * 60 * 60 + T.d *
24 * 60 * 60 + T.h * 60 * 60 + T.min * 60 + T.s;

        if (s1 == s2) {
            double deltas = 0;
            return deltas;
        }
        else if (s1 < s2) {
            double deltas = s2 - s1;
            return deltas;
        }
        else if (s1 > s2) {
            double deltas = s1 - s2;
            return deltas;
        }
    }
};
int A(int deltas) {
    int year, month, day, hour, minute, second;
    double avey = 365.25; double avem = 30.4375;
    return year = deltas / 60 / 60 / 24 / avey;
}
int B(int deltas, int year) {
    int month;
    double avey = 365.25; double avem = 30.4375;
    return month = deltas / 60 / 60 / 24 / avem - year * 12;
}
int C(int deltas, int year, int month) {
    int day, hour, minute, second;
    double avey = 365.25; double avem = 30.4375;
    return day = deltas / 60. / 60. / 24. - month * avem - year * avey;
}
int D(int deltas, int year, int month, int day) {
    int hour, minute, second;
    double avey = 365.25; double avem = 30.4375;
    return hour = deltas / 60. / 60. - 24 * (day + month * avem + year * avey);
}
int E(int deltas) {
    int year, month, day, hour, minute, second;
    double avey = 365.25; double avem = 30.4375;
    return minute = deltas / 60 % 60;
}
int F(int deltas) {
    int year, month, day, hour, minute, second;
    double avey = 365.25; double avem = 30.4375;
    return second = deltas % 60;
}
class d {
public:
        int deltas;
        int year, month, day, hour, minute, second;
        void QAQ()
        {
                year = A(deltas);
                month = B(deltas, year);
                day = C(deltas, year, month);
                hour = D(deltas, year, month, day);
                minute = E(deltas);
                second = F(deltas);
        }
        friend ostream& operator<<(ostream& output, const d c)
        {
                cout << endl;
                output << "Answer: Years: " << c.year << " - Months: " << c.month << " -
Days: " << c.day << " -Hours: " << c.hour << " -Minutes: " << c.minute << " - seconds: " <<
c.second << endl;
                return output;
        }
};
int main() {
        pika T1, T2; string temp;
        ifstream infile;
        infile.open("input.txt");
        ofstream outfile;
        outfile.open("output.txt");
        cout << "You have entered T1: ";
        infile >> T1;
        outfile << "You have entered T1: ";
        cout << "[ " << T1.y << " / " << T1.mon << " / " << T1.d << " - " << T1.h << " : " <<
T1.min << " : " << T1.s << " ]"<<" and T2: ";
        outfile<< "[ " << T1.y << " / " << T1.mon << " / " << T1.d << " - " << T1.h << " : " <<
T1.min << " : " << T1.s << " ]" << " and T2: ";
        infile >> T2;
        cout << "[ " << T2.y << " / " << T2.mon << " / " << T2.d << " - " << T2.h << " : " <<
T2.min << " : " << T2.s << " ]" << endl;;
        outfile << "[ " << T2.y << " / " << T2.mon << " / " << T2.d << " - " << T2.h << " : " <<
T2.min << " : " << T2.s << " ]" << endl;;

        cout << "Comparison"<<endl;
        outfile << "Comparison" << endl;
        haha a(T1);
        haha b(T2);
        d       result;
        if (a == b) {
                cout << "Date T1 is the same as Date T2"<<    endl;
                cout << "Date subtraction"<< endl;
                outfile << "Date T1 is the same as Date T2" << endl;
                outfile<< "Date subtraction" << endl;
                result.deltas = a - b;
                result.QAQ();
                cout << T1 << "-" << T2 << endl;
                cout<<result;
                outfile << T1 << "-" << T2 << endl << endl;
                outfile<<result;
        }
        else if (a < b) {
                cout << "Date T1 is smaller than Date T2\n";
                cout << "Date subtraction\n";
                outfile << "Date T1 is the smaller than Date T2" << endl;
                outfile << "Date subtraction" << endl;
                cout << T2 << "-" << T1;
                result.deltas = a - b;
                result.QAQ();
                cout << result;
                outfile << T2 << "-" << T1 << endl;;
                outfile << result;
        }
        else if (a > b) {
                cout << "Date T1 is greater than Date T2"<<endl;
                cout << "Date subtraction"<<endl;
                outfile << "Date T1 is greater than Date T2" << endl;
                outfile << "Date subtraction" << endl;
                cout << T1 << "-" << T2;
                result.deltas = a - b;
                result.QAQ();
                cout << result;
                outfile << T1 << "-" << T2 << endl;
                outfile << result;
        }
        infile.close();
        outfile.close();
}
```

1. \t = Tab
Can use it to print out 5 spaces
2. Compiling(编 译): Done by a program called a “Compiler”. Translates Code into Understandable Language for the Computer. Takes this translation and puts it into an executable file.
3. 命名：数字不能开头，只有_符号，
名称在 1 到 255 个字符之间
4. Casting – change variable type
double x = 4.23987;
cout << static_cast<int>(x);
5. Unsigned int does not have sign bit. 永不为负，手动定义
6. Overflow： If these values are exceeded, odd values might appear (might loop back around to 0, etc.)
7. rand 函数
#include <iostream>
#include <cstdlib>
#include <ctime>
...
1)srand((unsigned)time(NULL));
2)srand(time(0));
3)srand(seed);
x = rand() % 6 + 1;(1 到 6 随机整数）
8. 二进制转换
10—>2: 除 2 取余法
2—>10: abcd=$d*2^0+c*2^1+b*2^2+a*2^3$
9. Prototypes are a way of identifying function beforehand. 当函数被定义在 int main 之后时