Object Oriented Programming I

Topics today:

- File Input
- Functions

File Input

File Input

- File input needs **#include<fstream>**
- An input stream that reads from the file fileName is created by

```
ifstream streamName(fileName);
```

- After this, streamName can be used completely similar to **cin**
- fileName can be an absolute path, e.g. “**C:test.txt**”,
  or a path relative to the directory of the executable C++ program, e.g. “**input.txt**”

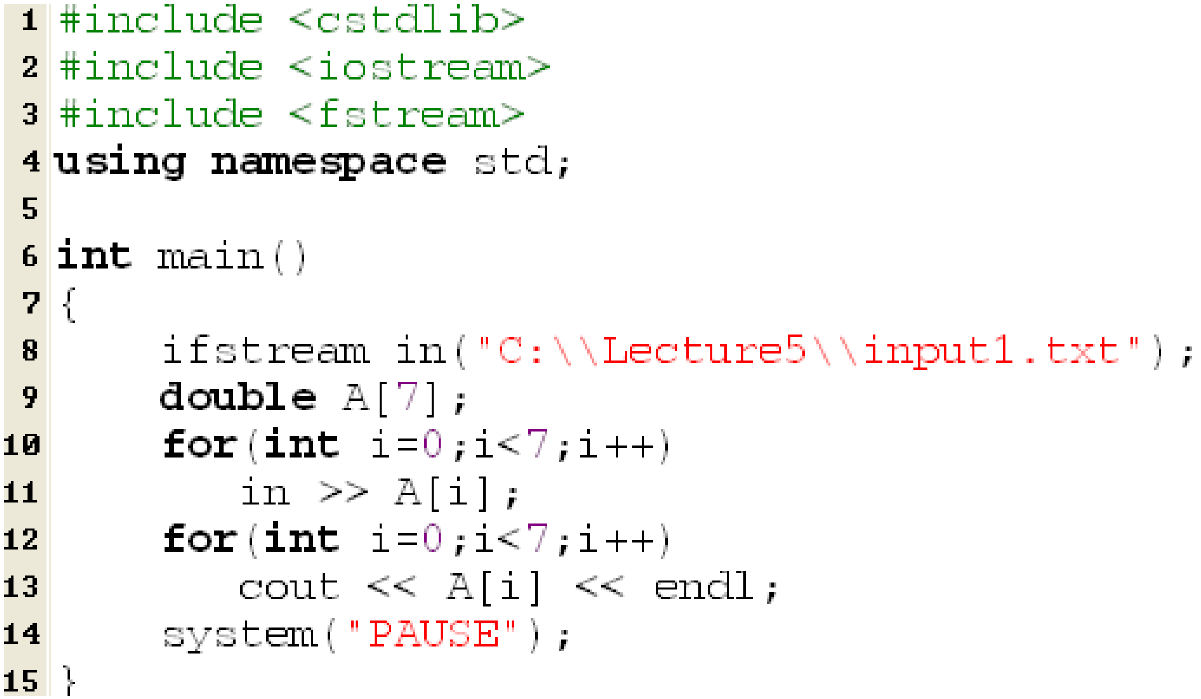
Example 1

- Create a textfile **input1.txt** with the content shown below
- Read the values contained in the file into an array with entries of type double
- Print the entries of the array to the screen

```
2342234.02384
3424.340
340349.29347
923482.23784
92347.347
20342.234234
820482.03284
```

Solution

Assume that the file **input1.txt** with the content on the last slide has been created already. Then the following program does the job.



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

int main()
{
    ifstream in("C:\\Lecture5\\input1.txt");
    double A[7];
    for(int i=0;i<7;i++)
       in >> A[i];
    for(int i=0;i<7;i++)
       cout << A[i] << endl;
    system("PAUSE");
}
```

Useful Functions for File Input

- Assume an input stream **in** has been declared, e.g. with **ifstream in(“input.txt”);**
- **in.eof()** returns true if the end of the file has been reached, otherwise false
- **in.fail()** returns true if the last input failed (e.g. string is attempted to be read into an integer), otherwise false
- **in.clear()** resets the input stream to its original state after an input failed

Using a while-loop for File Input

- Often we do not know how many data values a file contains
- Need to know when the end of the file is reached
- Solution:
  - create a temporary variable, say “**buffer**”
  - create an input stream, say “**in**”
  - use **while(in>>buffer)** or **while(!in.eof())**
  - in the while-loop, append the values to a vector
- Why it works: **(in>>buffer)** will be false if and only if the end of the input is reached or an input error is encountered

Problem 2

- Create a textfile **input1.txt** with the content shown below
- Read the values contained in the file into a vector using a loop **while(!in.eof())** and the **push_back** function for vectors
- Print the entries of the vector to the screen

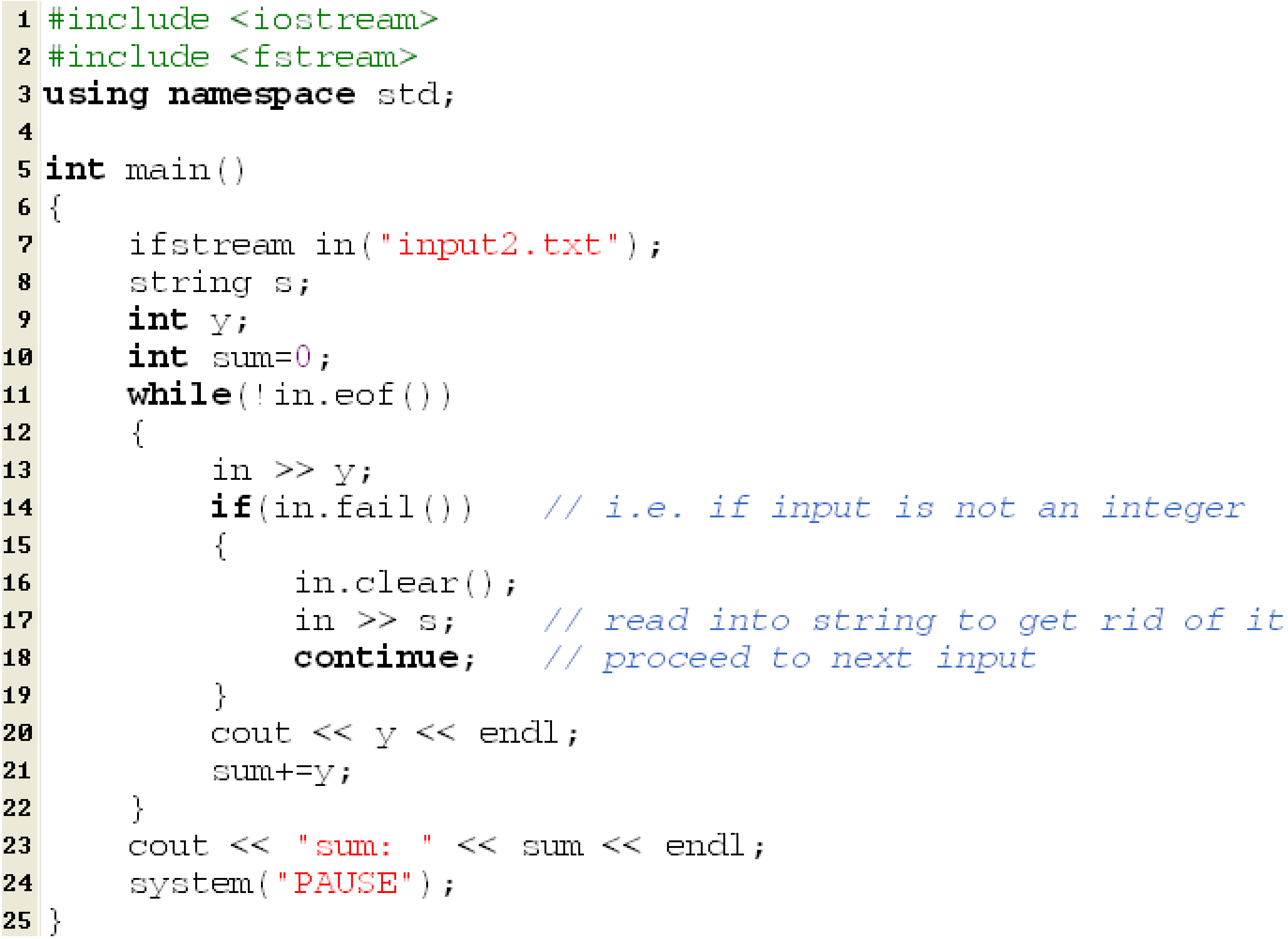
```
2342234.02384
3424.340
340349.29347
923482.23784
92347.347
20342.234234
820482.03284
```

Example 2 (extract numbers from a file)

- Create a textfile **input2.txt** with the content shown below
- Read the numbers from the file into a C++ program and print their sum to the screen.

```
jjsf
3444
&*^*&(
3234
abc
def
123
xyz
```

Solution



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

int main()
{
    ifstream in("input2.txt");
    string s;
    int y;
    int sum=0;
    while(!in.eof())
    {
        in >> y;
        if(in.fail())   // i.e. if input is not an integer
        {
            in.clear();
            in >> s;    // read into string to get rid of it
            continue;   // proceed to next input
        }
        cout << y << endl;
        sum+=y;
    }
    cout << "sum:  " << sum << endl;
    system("PAUSE");
}
```

“Messy” Input Files

- Often the data in input files are not arranged in a good way for C++ input
- Still we need to be able to extract the information we need
- The next problem is a simple example for this

Problem 3

- Download the file **NasdaqCompsiteIndex.txt** from edventure
- Write a program that reads the data from this file
- Note that only the numbers in the second column need to be read (corresponding year and month can be determined without reading it from the file)
- After the data are read, the program should ask the user to input a year and month on the keyboard
- The Nasdaq index for this year and month should be printed to the screen

Problem 4 (Reading Text from File)

- Create a textfile **input3.txt** with this content:

```
This is a textfile
containing text.
```

- Read the symbols in this file into a vector with entries of type char
- Print the symbols which were read (entries of the vector) on the screen
- Note that spaces are gone
- To read spaces, try **ifstream in(“input3.txt); string s; getline(in, s);** etc. as an alternative

Functions

Purpose of Functions

- We have learned about variables, loops, conditions, and vectors
- In principle, this is enough to do any kind of computation efficiently
- But what is missing?
- A good way to structure programs
- The most useful feature of C++ for structuring programs is functions

Why should we use functions?

History

- A programming project can involve millions of lines of source code
- Programming breakthrough in the 1950s:
  Use functions to divide a complex task into smaller, simpler tasks
- C and Fortran (1960s to 80s) massively use functions
- That’s why they are called procedural programming languages (procedure=function)
- In C++, functions are still fundamental

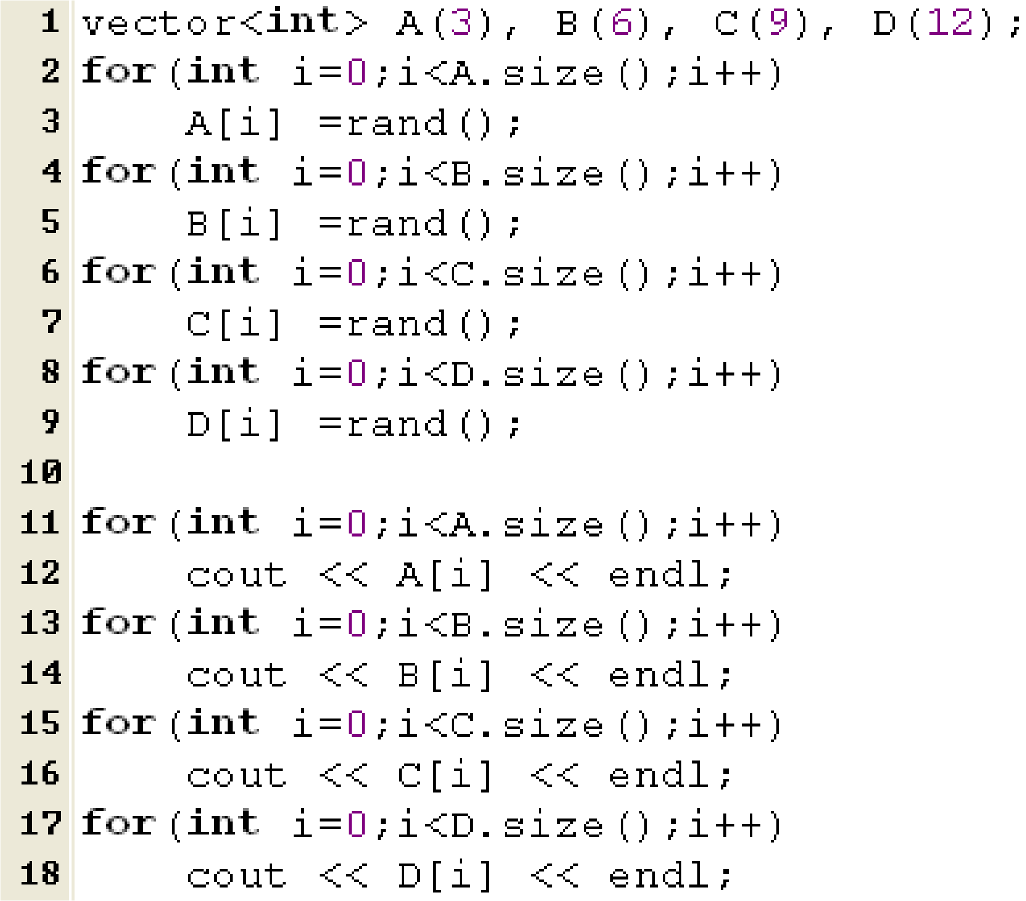
Main Ideas Behind Functions

- A function is a small, almost complete program inside a bigger program which is supposed to be independent from the rest of the program
- Each function is supposed to contribute one step to achieving the goal of the complete program
- It is much easier to solve a problem in small steps than by a single chunk of code inside the main function
- Separating a problem into small, easy steps naturally gives rise to application of functions
- A small step (e.g. printing all entries of a vector to the screen) which has to carried out repeatedly is best implemented by a function

Example Demonstrating Advantages of Functions

- Create four vectors with random entries of type **int** of length 3,6,9,12, respectively
- Print all entries of these vectors to the screen

Solution Without Functions

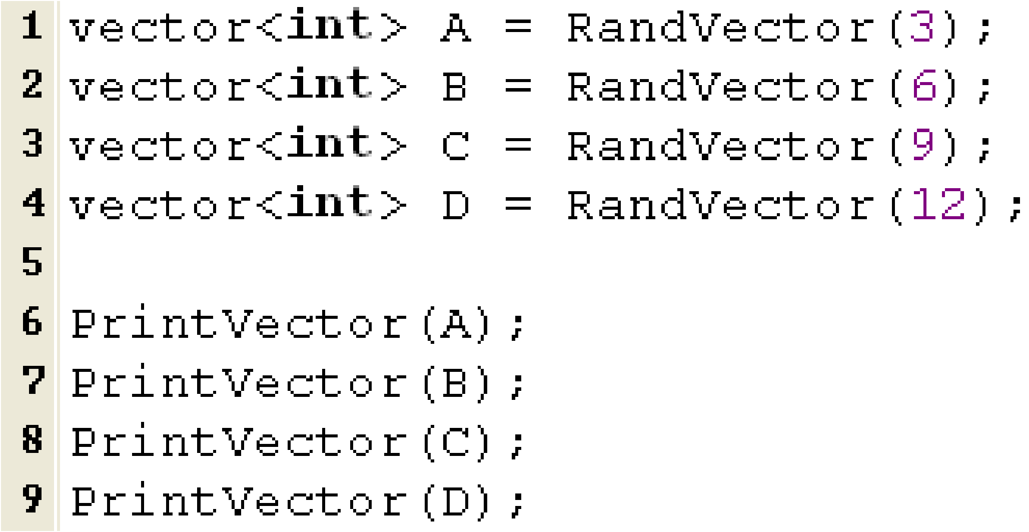


```
vector<int> A(3), B(6), C(9), D(12);
for(int i=0;i<A.size();i++)
    A[i] =rand();
for(int i=0;i<B.size();i++)
    B[i] =rand();
for(int i=0;i<C.size();i++)
    C[i] =rand();
for(int i=0;i<D.size();i++)
    D[i] =rand();

for(int i=0;i<A.size();i++)
    cout << A[i] << endl;
for(int i=0;i<B.size();i++)
    cout << B[i] << endl;
for(int i=0;i<C.size();i++)
    cout << C[i] << endl;
for(int i=0;i<D.size();i++)
    cout << D[i] << endl;
```

Solution With Functions

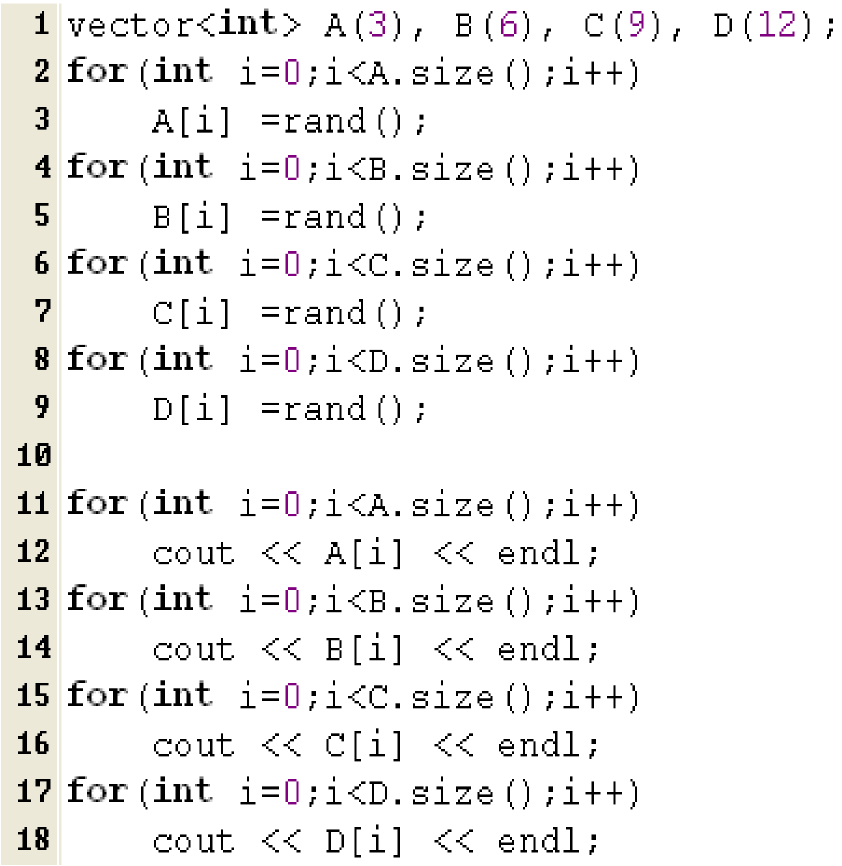
Suppose we have a function **RandVector(n)** which returns a random vector of length **n** and a function **PrintVector()** which prints all entries of a vector to the screen.

Solution:



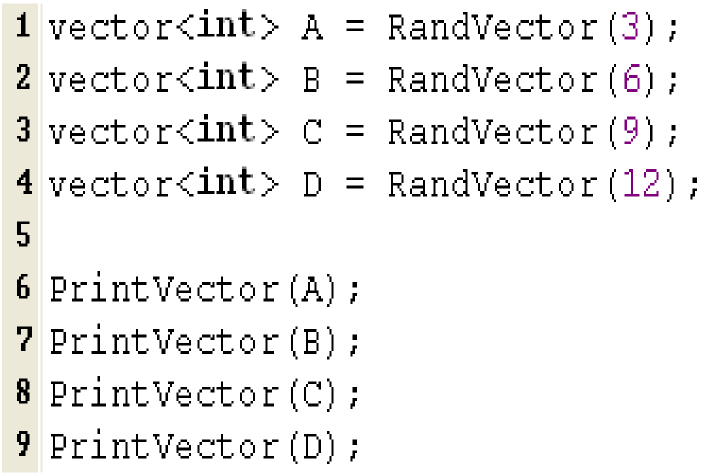
```
vector<int> A = RandVector(3);
vector<int> B = RandVector(6);
vector<int> C = RandVector(9);
vector<int> D = RandVector(12);

PrintVector(A);
PrintVector(B);
PrintVector(C);
PrintVector(D);
```

Without functions



```
vector<int> A(3), B(6), C(9), D(12);
for(int i=0;i<A.size();i++)
    A[i] =rand();
for(int i=0;i<B.size();i++)
    B[i] =rand();
for(int i=0;i<C.size();i++)
    C[i] =rand();
for(int i=0;i<D.size();i++)
    D[i] =rand();

for(int i=0;i<A.size();i++)
    cout << A[i] << endl;
for(int i=0;i<B.size();i++)
    cout << B[i] << endl;
for(int i=0;i<C.size();i++)
    cout << C[i] << endl;
for(int i=0;i<D.size();i++)
    cout << D[i] << endl;
```

With functions



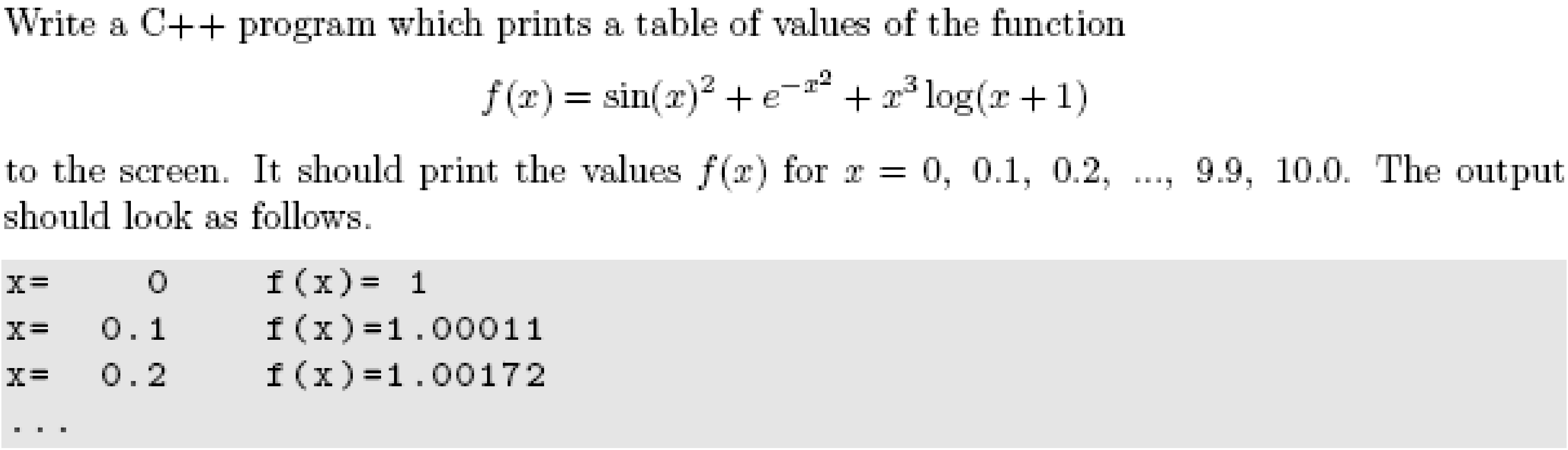
```
vector<int> A = RandVector(3);
vector<int> B = RandVector(6);
vector<int> C = RandVector(9);
vector<int> D = RandVector(12);

PrintVector(A);
PrintVector(B);
PrintVector(C);
PrintVector(D);
```

Which functions are available already?

Built-in Functions

- Some functions, like **rand()**, are already provided in C++
- These are called built-in functions
- See http://www.cppreference.com/wiki/ under “Standard C library” for the available built in functions, e.g. **sin, cos, exp, tan, time, assert, exit,…**
- To use them in C++, we need appropriate include statements, e.g. **#include<cmath>** for math functions

Problem 5 (built-in functions)

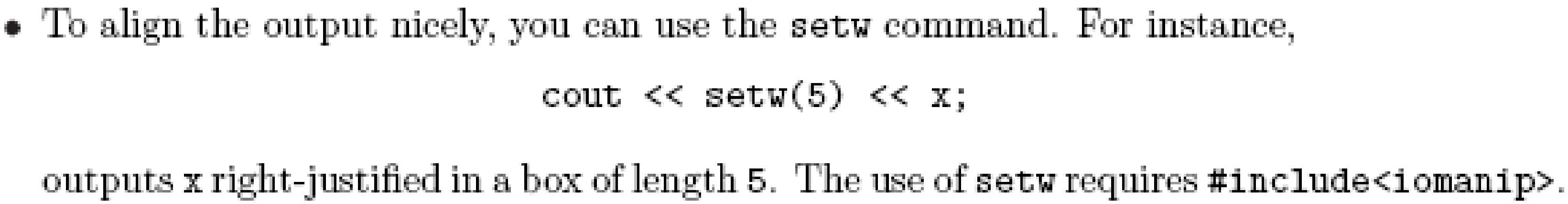


Write a C++ program which prints a table of values of the function

$$f(x) = \sin(x)^2 + e^{-x^2} + x^3 \log(x+1)$$

to the screen. It should print the values $f(x)$ for $x = 0,\ 0.1,\ 0.2,\ \ldots,\ 9.9,\ 10.0$. The output should look as follows.

```
x=    0    f(x)= 1
x=  0.1    f(x)=1.00011
x=  0.2    f(x)=1.00172
...
```



- To align the output nicely, you can use the setw command. For instance,

```
cout << setw(5) << x;
```

outputs x right-justified in a box of length 5. The use of setw requires #include<iomanip>.

User Defined Functions

- If we need functions which are not built-in, we need to write them ourselves
- Such functions are called user defined functions
- Most functions we use will be user defined (the set of built-in functions is minimal)

How does a C++ function compare with a mathematical function?

Mathematical Functions

Mathematical functions have a name,
input parameters, and return value.
Typical example:

$$
\begin{aligned}
p(n) \;=\; & \frac{1}{\pi\sqrt{8}} \sum_{k=1}^{\infty} \left\{ k^{1/2} \left(n - \frac{1}{24}\right)^{-3/2} \left(\sum_{\substack{h=1 \\ \mathrm{ggT}(h,k=1)}}^{k} \exp\left(-2\pi i h/k + \pi i \sum_{r=1}^{k-1} \frac{r}{k} \left(\frac{hr}{k} - \left[\frac{hr}{k}\right] - \frac{1}{2} \right) \right) \right) \right. \\
& \left. \left(\sqrt{\frac{2\pi^2}{3k^2}\left(n - \frac{1}{24}\right)} \cosh \sqrt{\frac{2\pi^2}{3k^2}\left(n - \frac{1}{24}\right)} - \sinh \sqrt{\frac{2\pi^2}{3k^2}\left(n - \frac{1}{24}\right)} \right) \right\}
\end{aligned}
$$

Name: p
Parameter: n
Return value: complicated (the complex number on the right side)

C++ Functions

- C++ functions are similar to mathematical functions (name, input parameters, return value). But:
- They can have a task aside from returning a value (destroying the operating system, for instance)
- They may have no input parameters
- They may have no return value

How to create a C++ function?

Function Definition

- To create a user defined function, we must provide the necessary C++ commands
- This is called the function definition
- Function definition consists of a function head and a function body
- Function head contains information on function name, input parameters, and return value
- Function body contains the C++ commands which fulfill the purpose of the function

Function Head

```
returnType FunctionName(parameters)
```

- This is the head of a function with name FunctionName
- The parameters consist of a comma separated list of variables declarations
- The returnType is the type of the value returned by the function
- If the function has no return value, the return type is **void**

Function Heads, Example 1

```
double pow(double a, double b)
```

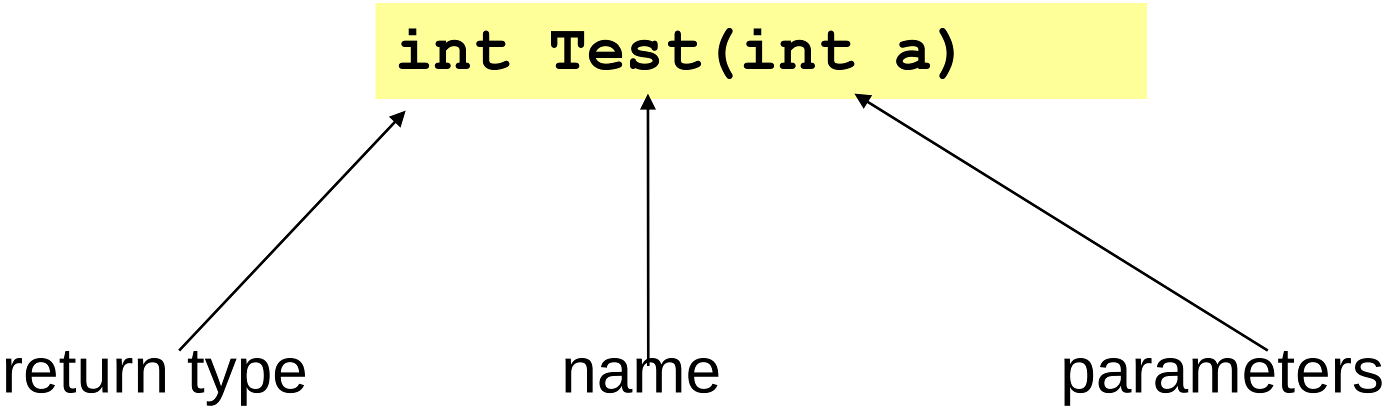
- Function head of the built-in function **pow**
- Has two input parameters of type double
- Returns a value of type double

Function Heads, Example 2

What is the head of a function with:

- Name: **Test**
- Input parameters: one variable **a** of type **int**
- Return value: of type **int**

Solution



```
int Test(int a)
```

Function Heads, Example 3

What is the head of a function with:

- Name: **DestroyWindows**
- No input parameters
- No return value

```
void DestroyWindows()
```

Function Heads, Example 4

What is the head of a function with:

- Name: **BoxVolume**
- 3 input parameter of type double
- Return type double

```
double BoxVolume(double a, double b, double c)
```

Function Heads, Example 5

What is the head of a function with:

- Name: **PrintVector2Screen**
- One input parameter: vector with entries of type **int**
- No return value

```
void PrintVector2Screen(vector<int> v)
```

Function Definition and Function Body

```
returnType FunctionName(parameters) // function head
{
     statements
}
```

- The function definition consists of the function head followed by the C++ statements which fulfill the purpose of the function
- These statements are enclosed in curly braces and are called the function body
- A function usually is defined at global scope, i.e. outside any other function (especially outside the `main` function)

Function Definition: Example 1

Write down the definition of a function that adds up three integers and returns the result.

Function Definition: Example 1

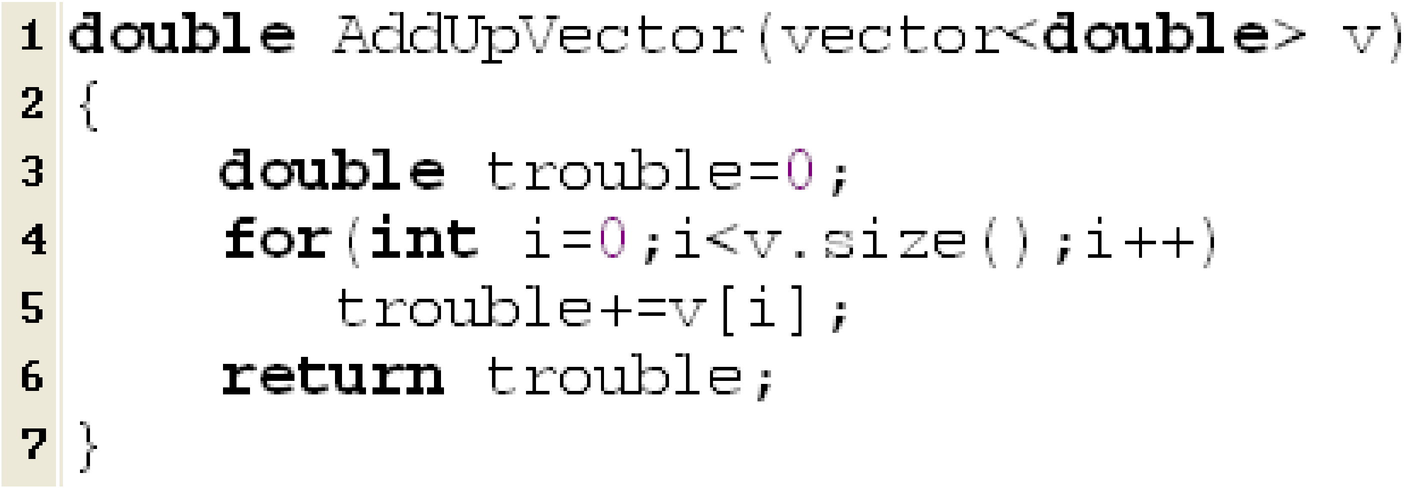
```
int Add(int a,int b,int c)
{
    return a+b+c;
}
```

Return type	**int**
Function name	**Add**
Parameters	**a,b,c,** all of type **int**
Function head	**int Add(int a,int b,int c)**
Function body	**{ return a+b+c; }**
Task	Add 3 integers and return result

Function Definition: Example 2

- Write down the definition of a function that adds up the entries of a vector with entries of type double and returns the result

Solution



```
double AddUpVector(vector<double> v)
{
    double trouble=0;
    for(int i=0;i<v.size();i++)
       trouble+=v[i];
    return trouble;
}
```

Return Values Must be Returned

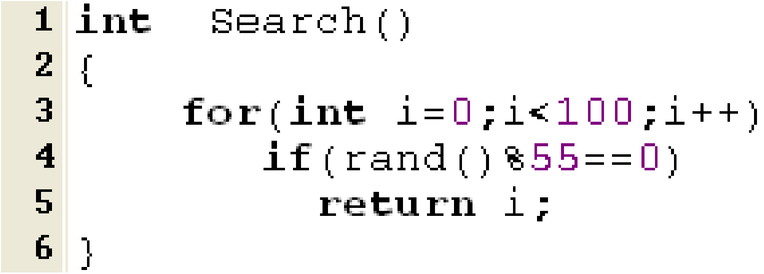
- If a function has return type **void**, it does not return a value
- In all other cases, we must make sure in the function body that a value of the correct type is returned under any circumstances
- For instance, **if(condition) return x;** usually will be wrong if we don’t make sure that a value is returned when the condition is false

Return Statements

```
return expression;
```

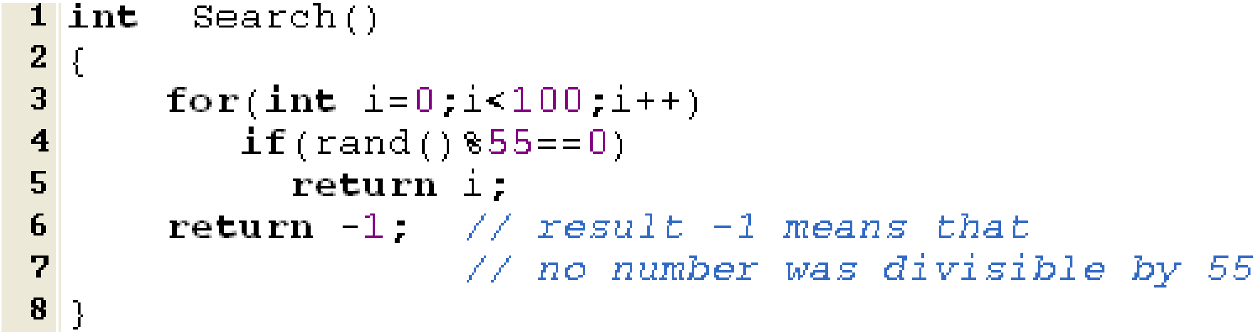
- Functions return values by such return statements
- From the function head, we know the return type of the function. The expression must have same type
- A return statement immediately stops the execution of the function
- **return;** stops the execution of the function without returning a value (only allowed if the return type is **void**)

Question: What is wrong with the following function?



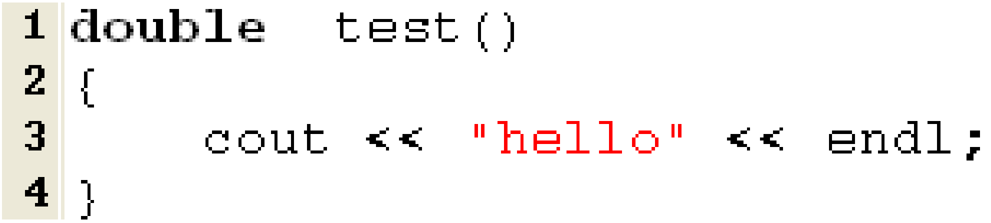
```
int   Search()
{
      for(int i=0;i<100;i++)
         if(rand()%55==0)
           return i;
}
```

If no random number is divisible by 55, no value is returned.
Correct version (for instance):



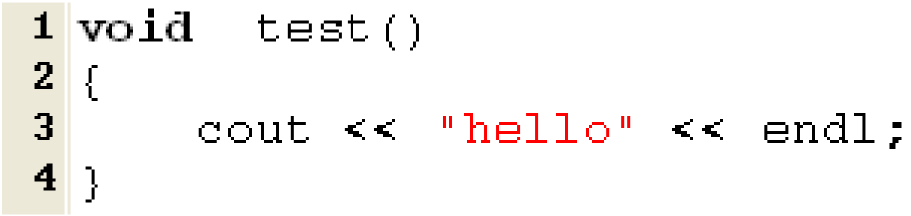
```
int   Search()
{
      for(int i=0;i<100;i++)
         if(rand()%55==0)
           return i;
      return -1;  // result -1 means that
                  // no number was divisible by 55
}
```

Question: What is wrong with the following function?



```
double  test()
{
      cout << "hello" << endl;
}
```

The return type is double, so there must be a return statement which returns a double value. Correct version:



```
void  test()
{
      cout << "hello" << endl;
}
```

Return type **void** means the functions does not return a value

Problem 6

- Write down the definition of a function that multiplies two **int**’s and returns the result
- Hint: to avoid integer overflow, you can use return type **long long** which has range

$$-2^{63},\ldots,2^{63}-1$$

Problem 7

- Write down the definition of a function that checks if two integers have the same parity (both even or both odd) and returns the answer as a bool value

How to use a C++ function: function call

Using a Function = Function Call

- Functions which are defined already can be used (repeatedly if necessary)
- To use a function, we call the function
- This is called a functions call
- Function definition and function call are two completely different things

Most Functions Need Input Information

Examples:

- **IsPrime(int n):** needs a number to test
- **PrintVector2Screen(vector<int> v):** needs a vector to print

Input information is passed to functions through the function parameters

How is Input Information Passed to a Function?

As a comma separated list in parenthesis after the function name.

Examples of function calls:

- **IsPrime(1001)**
- **AreEqual(v,w)**

Attention: if we use variables as parameters in a function call, we must declare and initialize them first!

Function Call

A function call consists of the function name followed by a comma separated list of values for the parameters enclosed in parenthesis

- Purpose:
- Passes the values of the parameters to the function
- Executes the function with this input

Function Call: Example 1

Function contained in **<cmath>**

pow(a,b): takes two parameters of type double and returns **a** to the power **b** in type double

Possible function call:

```
pow(2.0,31.0)
```

(function name followed by a comma separated list of values for the parameters)

What Happens in a Function Call?

The program reaches `pow(2.0,31.0)`
What happens?

- The parameter values 2.0 and 31.0 are passed to the function
- The function is executed with this input and computes the result 2147483648
- The function call is replaced by the result (!)
- Hence, **cout << pow(2.0,31.0);** has the same effect as **cout << 2147483648;**

Rules for Function Calls

- **Treat a function call like a value –** except it has no return value
- Functions with return type void:
  cannot treat function call as a value – functions call must be single command, not inside any expression
- The values of function parameters can be specified by any expressions of the correct type
- Unlike function definitions, function calls are done inside other functions (often inside the main function)

Function Call: Example 2

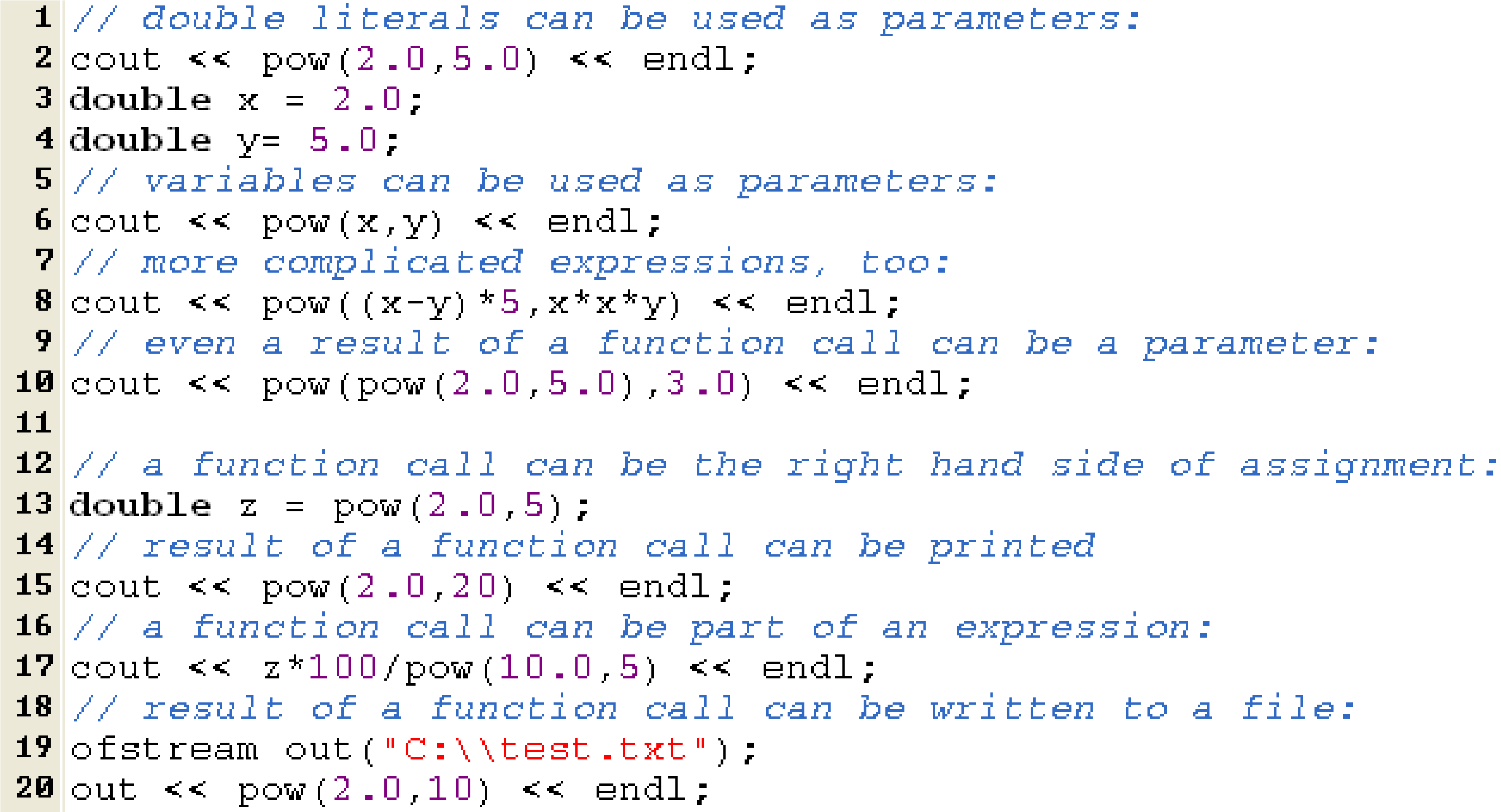
Task:

- Show that we can use any expression with return value of type double as parameters in a call of the function pow

- Show that with a function call of the function **pow**, we can do exactly the same things as with a value of type double:

  - print to the screen,
  - use in other expressions,
  - write to a file etc.

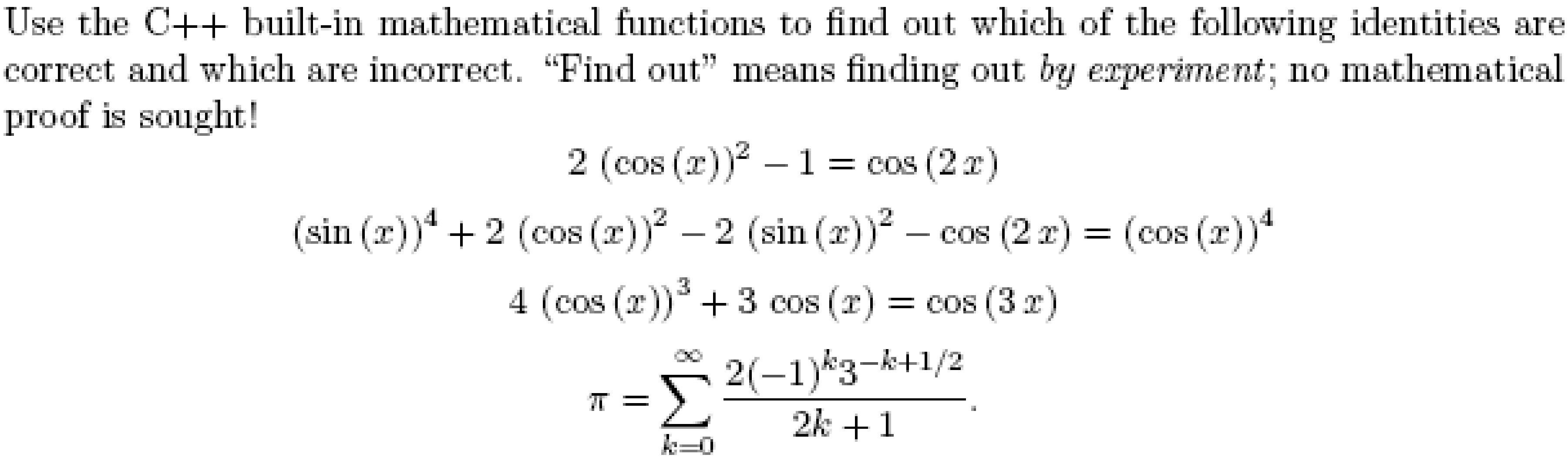
Solution



```
// double literals can be used as parameters:
cout << pow(2.0,5.0) << endl;
double x = 2.0;
double y= 5.0;
// variables can be used as parameters:
cout << pow(x,y) << endl;
// more complicated expressions, too:
cout << pow((x-y)*5,x*x*y) << endl;
// even a result of a function call can be a parameter:
cout << pow(pow(2.0,5.0),3.0) << endl;

// a function call can be the right hand side of assignment:
double z = pow(2.0,5);
// result of a function call can be printed
cout << pow(2.0,20) << endl;
// a function call can be part of an expression:
cout << z*100/pow(10.0,5) << endl;
// result of a function call can be written to a file:
ofstream out("C:\\test.txt");
out << pow(2.0,10) << endl;
```

Additional Problems

Problem 8



Use the C++ built-in mathematical functions to find out which of the following identities are correct and which are incorrect. “Find out” means finding out *by experiment*; no mathematical proof is sought!

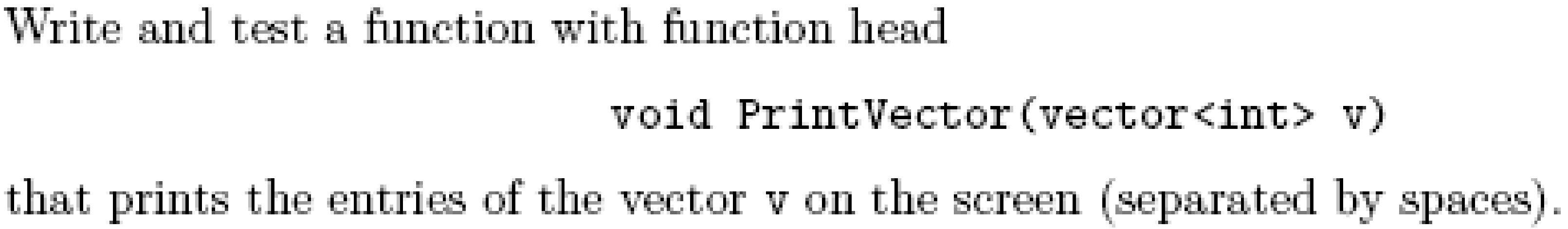
$$2\,(\cos(x))^2 - 1 = \cos(2\,x)$$

$$(\sin(x))^4 + 2\,(\cos(x))^2 - 2\,(\sin(x))^2 - \cos(2\,x) = (\cos(x))^4$$

$$4\,(\cos(x))^3 + 3\,\cos(x) = \cos(3\,x)$$

$$\pi = \sum_{k=0}^{\infty} \frac{2(-1)^k 3^{-k+1/2}}{2k+1}.$$

Problem 9

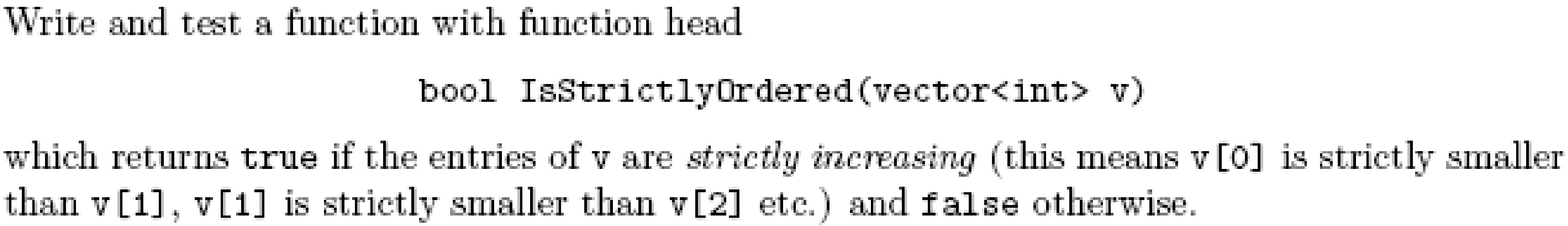


Write and test a function with function head

```
void PrintVector(vector<int> v)
```

that prints the entries of the vector `v` on the screen (separated by spaces).

Problem 10

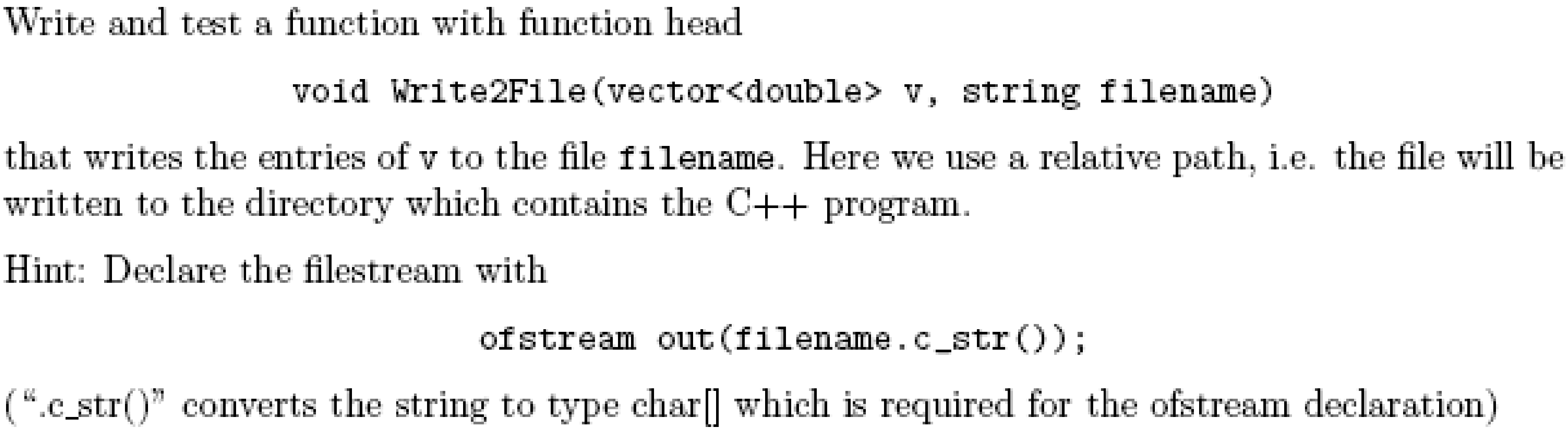


Write and test a function with function head

```
bool IsStrictlyOrdered(vector<int> v)
```

which returns `true` if the entries of `v` are *strictly increasing* (this means `v[0]` is strictly smaller than `v[1]`, `v[1]` is strictly smaller than `v[2]` etc.) and `false` otherwise.

Problem 11



Write and test a function with function head

```
void Write2File(vector<double> v, string filename)
```

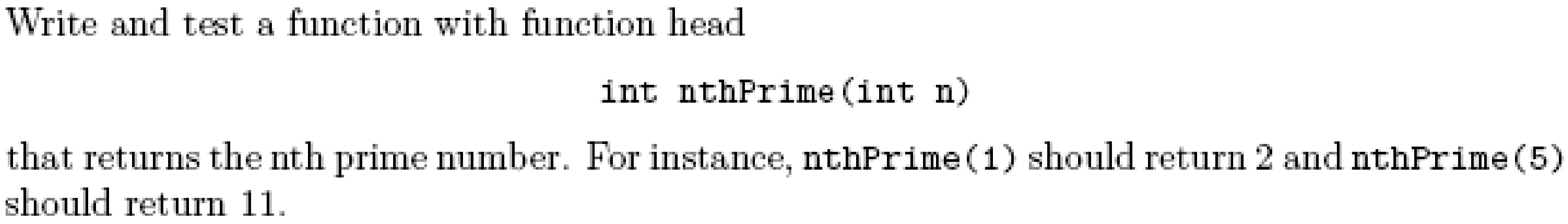
that writes the entries of `v` to the file `filename`. Here we use a relative path, i.e. the file will be written to the directory which contains the C++ program.

Hint: Declare the filestream with

```
ofstream out(filename.c_str());
```

(".c_str()" converts the string to type char[] which is required for the ofstream declaration)

Problem 12



Write and test a function with function head

```
int nthPrime(int n)
```

that returns the nth prime number. For instance, `nthPrime(1)` should return 2 and `nthPrime(5)` should return 11.

It is ok if the function only works for n up to 100,000