CS 31 Discussion 1A, Week 4

Zengwen Yuan (zyuan [at] cs.ucla.edu)
Humanities A65, Friday 10:00—11:50 a.m.
Today’s focus

• Notes from the project 2 grading

• Function
  • call predefined function
  • define a function
  • call a function by reference
  • scope
Project 2 quick comments

• coding style & comments
  • magic number
• report
  • test cases
• homework
  • concise description
• coding problem
Function: predefined

• An example

```c
#include <cmath>

int main() {
    double x = sqrt(9.0);
}
```

• What to know:
  • declare what library you would like to use
  • figure out the parameter(s) and the return type
### Some Predefined Functions

<table>
<thead>
<tr>
<th>NAME</th>
<th>DESCRIPTION</th>
<th>TYPE OF ARGUMENTS</th>
<th>TYPE OF VALUE RETURNED</th>
<th>EXAMPLE</th>
<th>VALUE</th>
<th>LIBRARY HEADER</th>
</tr>
</thead>
<tbody>
<tr>
<td>sqrt</td>
<td>Square root</td>
<td>double</td>
<td>double</td>
<td>sqrt(4.0)</td>
<td>2.0</td>
<td>cmath</td>
</tr>
<tr>
<td>pow</td>
<td>Powers</td>
<td>double</td>
<td>double</td>
<td>pow(2.0, 3.0)</td>
<td>8.0</td>
<td>cmath</td>
</tr>
<tr>
<td>abs</td>
<td>Absolute value for int</td>
<td>int</td>
<td>int</td>
<td>abs(-7)</td>
<td>7</td>
<td>cstdlib</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>abs(7)</td>
<td>7</td>
<td>cstdlib</td>
</tr>
<tr>
<td>labs</td>
<td>Absolute value for long</td>
<td>long</td>
<td>long</td>
<td>labs(-70000)</td>
<td>70000</td>
<td>cstdlib</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>labs(70000)</td>
<td>70000</td>
<td>cstdlib</td>
</tr>
<tr>
<td>fabs</td>
<td>Absolute value for double</td>
<td>double</td>
<td>double</td>
<td>fabs(-7.5)</td>
<td>7.5</td>
<td>cmath</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>fabs(7.5)</td>
<td>7.5</td>
<td>cmath</td>
</tr>
<tr>
<td>ceil</td>
<td>Ceiling (round up)</td>
<td>double</td>
<td>double</td>
<td>ceil(3.2)</td>
<td>4.0</td>
<td>cmath</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>ceil(3.9)</td>
<td>4.0</td>
<td>cmath</td>
</tr>
<tr>
<td>floor</td>
<td>Floor (round down)</td>
<td>double</td>
<td>double</td>
<td>floor(3.2)</td>
<td>3.0</td>
<td>cmath</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>floor(3.9)</td>
<td>3.0</td>
<td>cmath</td>
</tr>
<tr>
<td>exit</td>
<td>End program</td>
<td>int</td>
<td>void</td>
<td>exit(1);</td>
<td>None</td>
<td>cstdlib</td>
</tr>
<tr>
<td>rand</td>
<td>Random number</td>
<td>None</td>
<td>int</td>
<td>rand( )</td>
<td>Varies</td>
<td>cstdlib</td>
</tr>
<tr>
<td>srand</td>
<td>Set seed for rand</td>
<td>unsigned int</td>
<td>void</td>
<td>srand(42);</td>
<td>None</td>
<td>cstdlib</td>
</tr>
</tbody>
</table>

All these predefined functions require using namespace std; as well as an include directive.
Quiz: PRNG

• You work at a company called Zzyzx Inc. Your manager would like to host a sweepstake event. For that, he asked for your help to write a program. It randomly generates two integer numbers among 1–100 to decide two lucky winners.
Function: define a function

• Steps to follow
  • function declaration (function prototype/signature)
  • function body (function implementation)
  • function call

Keep your code DRY
Function: big picture

```cpp
#include <iostream>

// function declaration
return_type func_name (param_1_type param_1_name, param_2_type param_2_name, ...);

int main() {
    // function call
    return_type var_x = func_name (arg_1, arg_2, ...);
}

/*
 * Note: The top level comment above a function
 * goes here using the multi-line comment, and usually
 * should describe the function's input and output.
 */
return_type func_name (param_1_type param_1_name, param_2_type param_2_name, ...) {
    // func_name do stuff
}
```
Function declaration

- A function declaration is its name, what kind of object it returns, and what kind of object(s) it accepts.

```c
// function declaration
return_type func_name (param_1_type param_1_name,
param_2_type param_2_name, ...);
```
Function implementation

• Good practice: put it after the main() function

• Return type, parameter type must match the function declaration

• Parameters’ order must also match

```c
/*
 * Note: The top level comment above a function
 * goes here using the multi-line comment, and usually
 * should describe the function's input and output.
 */
return_type func_name (param_1_type param_1_name,
param_2_type param_2_name, ...) {
    // func_name do stuff
}
```
Example: what it does?

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

// function prototype for foo
double foo(int x);

int main() {
    cout << foo(5) << endl;
    cout << foo(0) << endl;
}

// function implementation for foo
double foo(int x) {
    return 1.0 / x;
}
```
Example: what it does?

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

// function prototype for foo
int foo(int x);

int main() {
    cout << foo(2) << endl;
    cout << foo(0) << endl;
}

// function implementation for foo
int foo(int x) {
    x *= 2;
    if (x < 100)
        return foo(x);
    return x;
}
```
Quiz: Fibonacci series

- A series of number in which each number is the sum of preceding two numbers is known as Fibonacci series.

0, 1, 1, 2, 3, 5, 8, 13, 21, 34...

- The first two numbers in the series are seed values, i.e., F0 = 0 and F1 = 1.

- Define a function named `fib` to generate this series, i.e., `fib(6) = 8` and so on.
A reference variable is an alias, or put simply, another name for an already existing variable.

Once a reference is initialized with a variable, either the variable name or the reference name may be used to refer to the variable.

Syntax for a reference variable:

\[<\text{type}> & <\text{name}>\]
Example: change on reference variable

```cpp
#include <iostream>
#include <string>
using namespace std;

int main() {
    string applesauce = "Pure applesauce!";
    string& nonsense = applesauce;

    cout << "nonsense is: " << nonsense << endl;

    // Now, let's use another quote...
    applesauce = "Jiggery pokery."
    // ...and see what happens to nonsense
    cout << "nonsense is: " << nonsense << endl;
}
```
#include <iostream>
using namespace std;

// function declaration
void swap(int& x, int& y);

int main () {
    int a = 10;
    int b = 20;
    cout << "Before swap, value of a: " << a << endl;
    cout << "Before swap, value of b: " << b << endl;
    swap(a, b);
    cout << "After swap, value of a: " << a << endl;
    cout << "After swap, value of b: " << b << endl;
}

// function definition to swap the values.
void swap(int& x, int& y) {
    int temp;
    temp = x; /* save the value at address x */
    x = y;    /* put y into x */
    y = temp; /* put x into y */
}
Scope: an example

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

void foo(int x);
int x = 6;

int main() {
    foo(x);
    int x = 5;
    foo(x);

    if (x > 5) {
        int x = 4;
        foo(x);
    } else {
        int x = 3;
        foo(x);
    }
}

void foo(int x) {
    cout << "x = " << x << endl;
}
```
Scope: explained

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

void foo(int x);
int x = 6;

int main() {
    foo(x);
    int x = 5;
    foo(x);

    if (x > 5) {
        int x = 4;
        foo(x);
    } else {
        int x = 3;
        foo(x);
    }
}

void foo(int x) {
    cout << "x = " << x << endl;
}
```