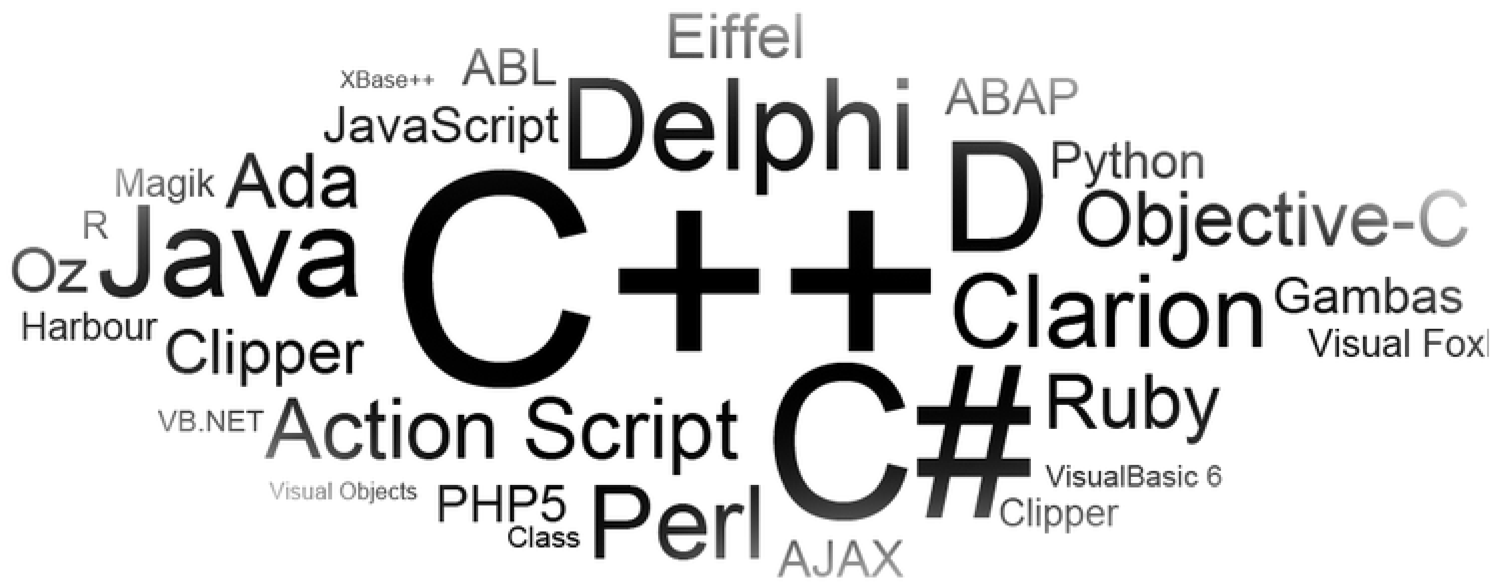
Programming Fundamentals II

Lecture 6: Functions



Outline

- Introduction to Functions
- Types of Functions
    - Standard C++ Library Functions
    - User-Defined Functions
- Examples

Introduction

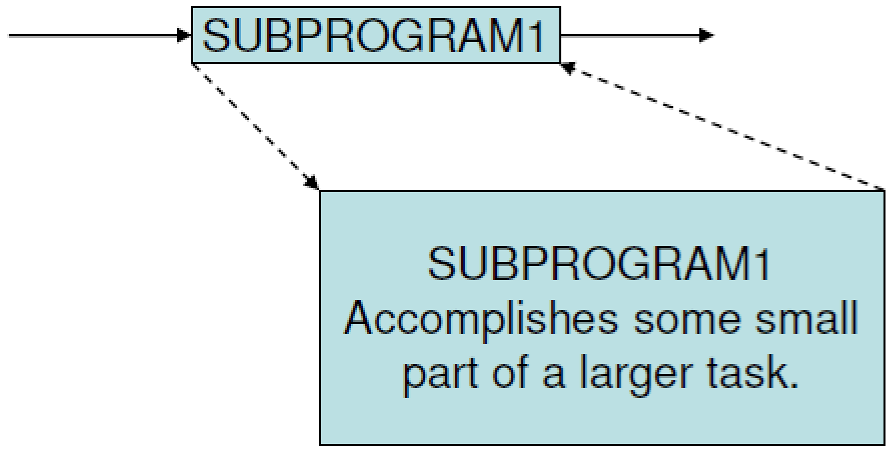
- Two types of Program Design in terms of capacity:
  1. Small Programs
     - Easily understood in a single sequence of steps.
     - Little refinement.
     - A single algorithm is enough.
  2. Larger Programs
     - Difficult to understand and remember a long sequence of steps.
     - Usually consists of several small problems requiring a lot of refinement.
     - Use several small algorithms.

Introduction

- What is a function?
  - A *function* is a block of code with a name. The function can be executed by calling the function.
- Reasons for using functions:
  1. Solving large problems need splitting the problem down into a series of sub-problems, which may be split into further sub-problems etc.
  2. Frequent occurrence of some tasks in a large program.
  3. There are many specialized problem areas that every programmer can not know all of them.

Introduction

- Subprograms in C++
  - C++ supports subprograms through functions.
  - Look like small programs within a program.
  - Flow



Types of Functions

- There are two types of functions:
  1. Standard C++ Library Functions
  2. User-Defined Functions

Standard C++ Library Functions

- The *Standard C++ Library* is a collection of pre-defined functions and other program elements which are accessed through *header files*.
- Example:-

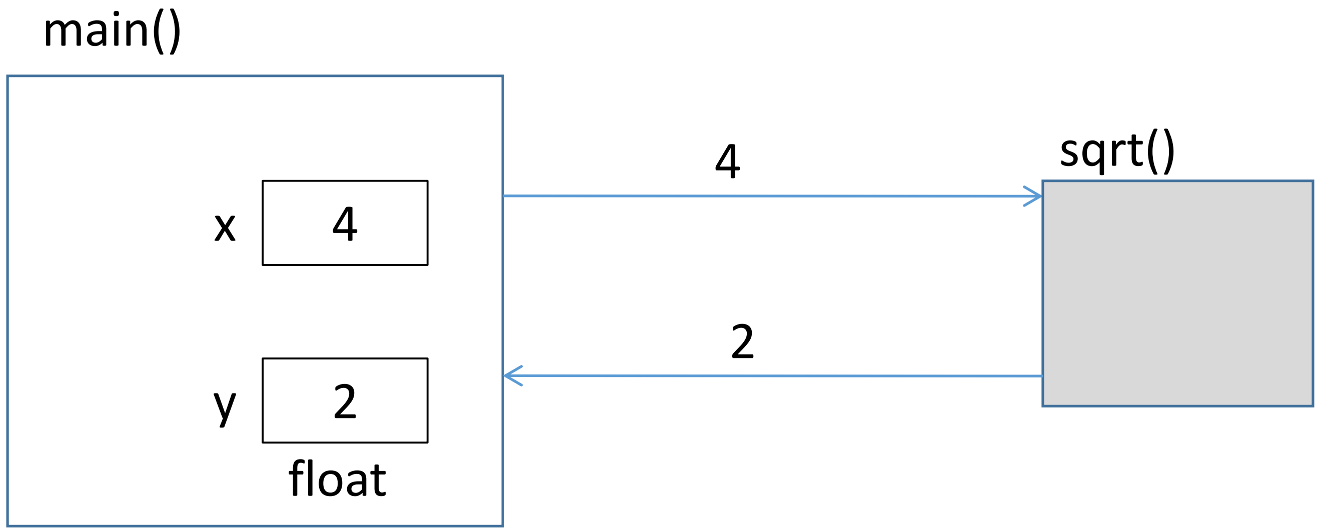
Functions	**Header File**
sqrt()	<cmath>
pow()	<cmath>
sin()	<cmath>
rand()	<cstdlib>

Standard C++ Library Functions

- Example:

```
#include <iostream>
        #include <cmath>
        using namespace std;
        int main(){
            int  x = 4;
            float  y = 0.0;
            y = sqrt(x);
            cout<<" square root of 4 is  "<< y <<endl;
            return 0;
        }
```

Standard C++ Library Functions



User-Defined Functions

- Despite the great variety of functions provided by the Standard C++ Library, programmers still need to be able to define their own functions.
- Three important *concepts* about functions (either Standard C++ Library or User-Defined):
  - *Function call* or *invocation*, tells the program when to use a function.
  - *Function prototype or declaration*, describes the interface of the function.
  - *Function definition*, describes what statements the function can execute.

User-Defined Functions

- A function definition typically consists of a *return type*, a *name*, a *list of zero or more parameters*, and a *body* as shown bellow:

Syntax:

Function Head

```
DataType    functionName(DataType  Parameter1, DataType Parameter2, …etc)
{
            Statement1
            Statement2
            …
}
```

Function Body

User-Defined Functions

- A function may either return a value or may not.
  - If the function returns a value, then the data type of the function can be any of **int**, **float**, **char**, …etc.
  - If the function returns nothing, then the function type must be “**void**”.
- A function is defined either before main() function or after main() function inside a “.cpp” file.
  - If the function is defined before main() function, then it can be invoked (called) directly from main() function or any other function.

User-Defined Functions

- If the function is defined after main() function, then the function prototype (declaration) must be put before main() function so the function can be invoked from main() function or any other function.

Example 1

- **Function with no parameters**

```
void skip3Lines(){
        cout<<endl<<endl<<endl;
}
```

Function Definition

```
int main(){
     int a = 5, b = 4;
     cout<< "value of a is  " <<a;
     skip3Lines(); //calling the function to skip three lines
     cout<< "value of b is  " <<b;
}
```

Example 2

- **Function with parameters and no return value**

```
void skipLines(int  n){
       for (int i = 0; i < n; i++)
            cout << endl;
}
int main(){
     int a = 5, b = 4, x = 6;
     cout<< "value of a is  " <<a;
     skipLines(x); //calling the function to skip six lines
     cout<< "value of b is  " <<b;
}
```

Function Definition

Example 3

- **Function with parameters that returns a value**

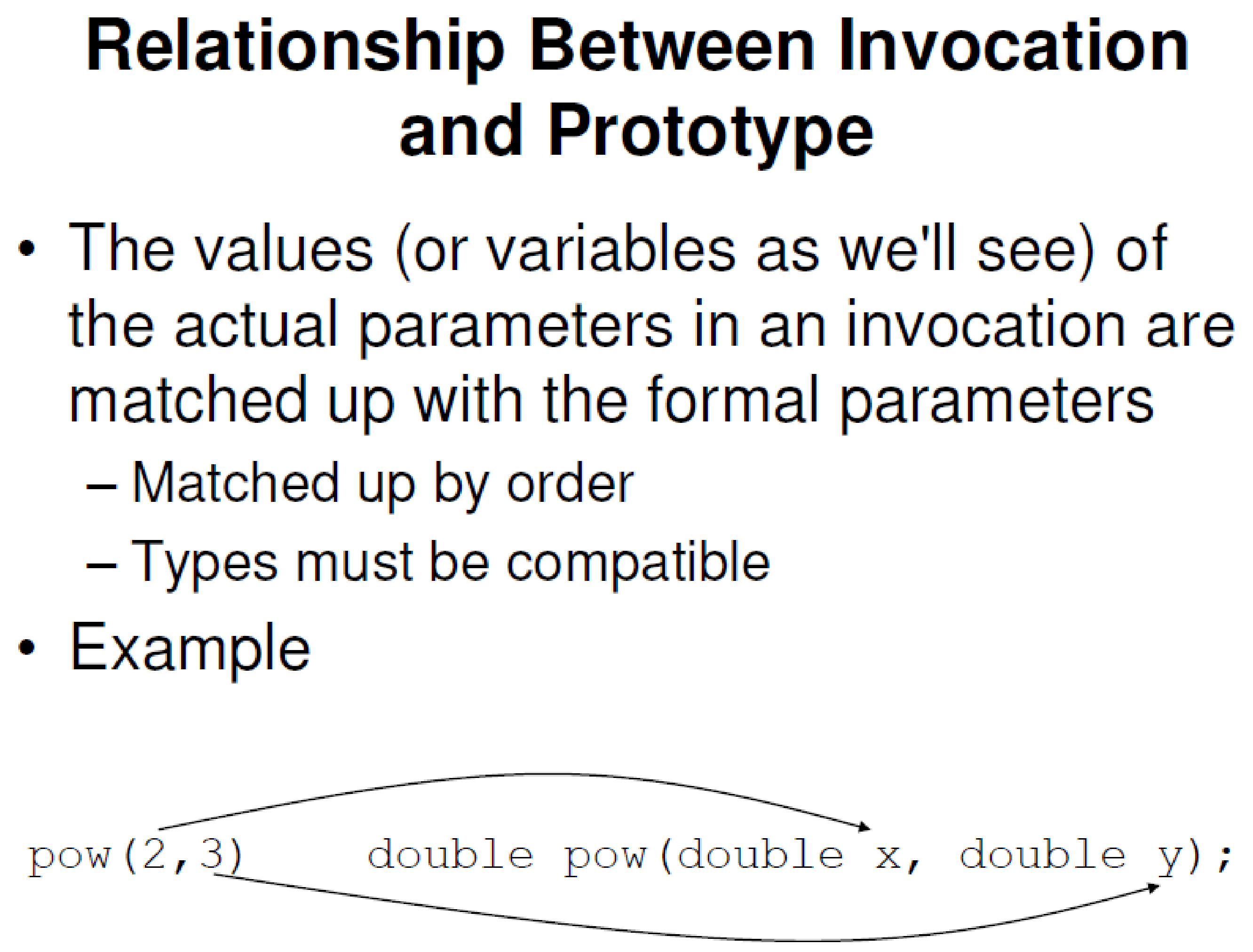
```
int  maxNum(int  n, int m){
       int  max = 0;
       if(n > m)
          max = n;
       else
          max = m;
       return  max;
}
int main(){
     int a = 0, b = 0;
     cin>>a>>b;
     //calling the function to print the maximum value between a and b
     cout<< " Max value of a and b is   " << maxNum(a,b);
}
```

$\left.\vphantom{\begin{matrix}1\\1\\1\\1\\1\\1\\1\\1\end{matrix}}\right\}$ ***Function Definition***

Function Declaration (Prototype)

- The last three examples illustrate one method of defining a function in a program. The complete definition of the function is listed above the main program.
- Another more common way is to list only the function's *head* above the main program, and then list the function's complete definition (head and body) below the main program.
- This way, the function's declaration is separated from its definition.
- So a *function's declaration* is simply the function's head , followed by a semicolon.

Relationship between Invocation and Prototype



Relationship Between Invocation and Prototype

- The values (or variables as we'll see) of the actual parameters in an invocation are matched up with the formal parameters
  - Matched up by order
  - Types must be compatible
- Example

```
pow(2,3)     double pow(double x, double y);
```

Example 4 (Function Declaration)

- **Function with parameters that returns a value**

```
int  maxNum(int  a, int  b);  // Function Declaration (Prototype)
int main(){
     int a = 0, b = 0;
     cin>>a>>b;
     cout<< " Max value of a and b is   " << maxNum(a,b); // Function Invocation
}
int  maxNum(int  n, int  m){
       int  max = 0;
       if(n > m)
          max = n;
       else
          max = m;
       return  max;
}
```

Function Definition

Summary

- Introduction to Functions
- Types of Functions
  - Standard C++ Library Functions
  - User-Defined Functions
- Examples