Introduction to Programming

Chapter 6

Functions

Functions

C++ functions (aka methods)

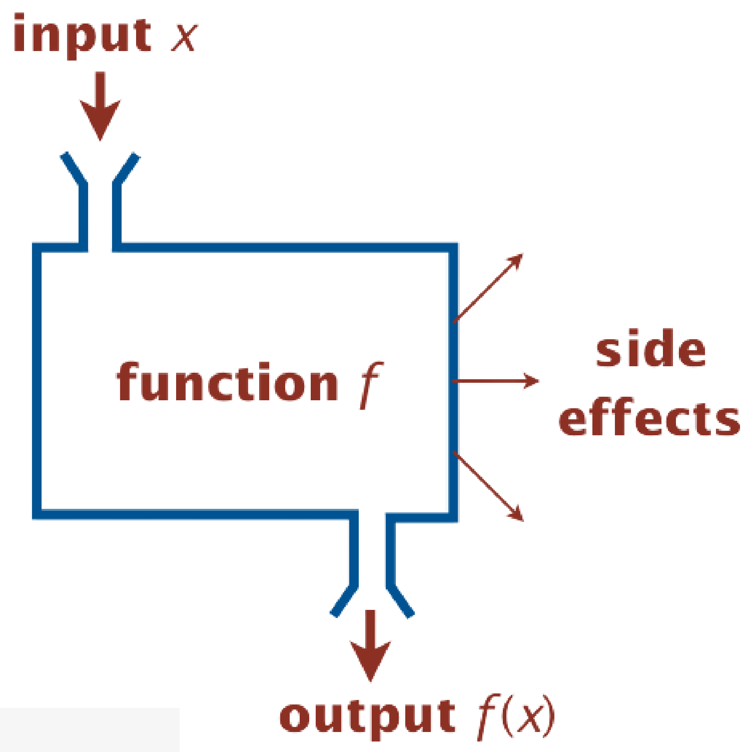
- Takes zero or more input arguments.
- Returns zero or one output value.
- May cause side effects (e.g., output to screen).

Applications

- Scientists use mathematical functions to calculate formulas.
- Programmers use functions to build modular programs.
- You use functions for both.

Examples seen so far

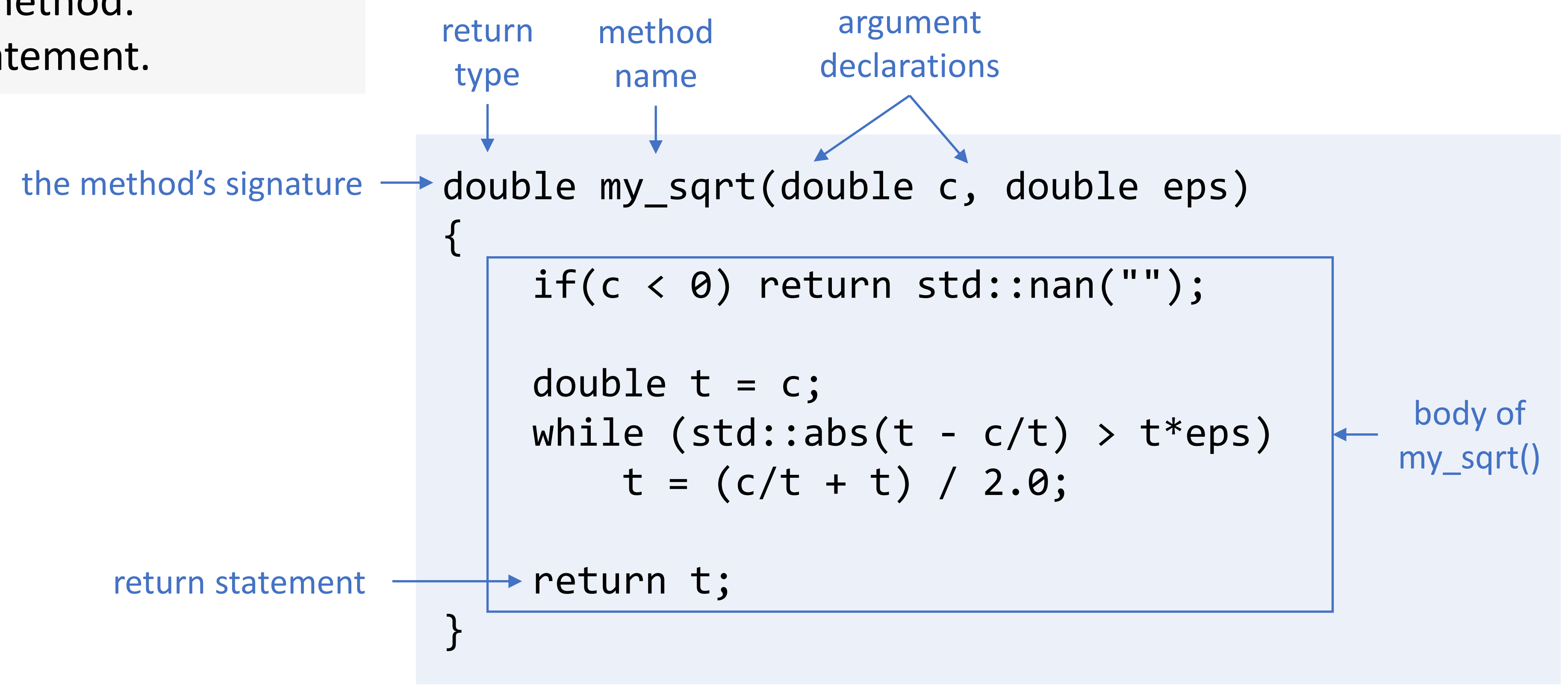
- Standard library: `rand(), abs(), to_string(), max(), min()`
- raylib: `DrawLine(), DrawRectangle(), DrawText()`
- User-defined functions: `main()`



Anatomy of a C++ method

To define a function (method)

- Create a *name*.
- Declare type and name of *argument(s)*.
- Specify type for *return value*.
- Implement *body* of method.
- Finish with *return* statement.



```
double my_sqrt(double c, double eps)
{
    if(c < 0) return std::nan("");

    double t = c;
    while (std::abs(t - c/t) > t*eps)
        t = (c/t + t) / 2.0;

    return t;
}
```

Function declaration and definition

In C++, a function need to be declared before use

Function declaration only

```
double my_sqrt(double c, double eps);
```

Function declaration and definition

```
double my_sqrt(double c, double eps)
{
    if(c < 0) return std::nan("");

    double t = c;
    while (std::abs(t - c/t) > t*eps)
        t = (c/t + t) / 2.0;

    return t;
}
```

Flow of Control

Functions provide a new way to control the flow of execution.

```
#include<iostream>
#include<cmath>
```

`my_sqrt()` method

```
double my_sqrt(double c, double eps) {
    if(c < 0) return std::nan("");
    double t = c;
    while (std::abs(t - c/t) > t*eps)
        t = (c/t + t) / 2.0;
    return t;
}
```

`main()` method

Program execution always start here

```
int main() {
    double x;
    do {
        std::cout << "Enter a number (0 to quit): ";
        std::cin >> x;
        std::cout << "Square root of " << x << " is " << my_sqrt(x, 1E-5) << "\n";
    } while(x!=0);
}
```

Note: We are using my_sqrt() from Chap 3 here to illustrate the basics with a familiar function.

Our focus is on control flow here. See Chap 3 for technical details.

You can use sqrt() from <cmath>

Scope

Def. The scope of a variable is the code that can refer to it by name.

A variable's scope is the code following its declaration, in the same block.

```
double my_sqrt(double c, double eps) {
    if(c < 0) return std::nan("");
    double t = c;
    while (std::abs(t - c/t) > t*eps)
        t = (c/t + t) / 2.0;
    return t;
}

int main() {
    double x;
    do {
        std::cout << "Enter a number (0 to quit): ";
        std::cin >> x;
        std::cout << "Square root of " << x << " is " << my_sqrt(x, 1E-5) << "\n";
    } while(x!=0);
}
```

scope of c and eps

scope of t

scope of x

cannot refer to c, eps, or t in main()

Best practice. Declare variables so as to limit their scope.

Flow of Control

```
#include<iostream>
#include<cmath>

double my_sqrt(double c, double eps) {
    if(c < 0) return std::nan("");
    double t = c;
    while (std::abs(t - c/t) > t*eps)
        t = (c/t + t) / 2.0;
    return t;
}

int main() {
    double x;
    do {
        std::cout << "Enter a number (0 to quit): ";
        std::cin >> x;
        double y = my_sqrt(x, 1E-5);

        std::cout << "Square root of " << x << " is "
                  << y << "\n";
    } while(x!=0);
}
```

Summary of flow control for a function call

- Control transfers to the function code.
- Argument variables are declared and initialized with the given values.
- Function code is executed.
- Control transfers back to the calling code (with return value assigned in place of the function name in the calling code).

"pass by value"
(other methods will be discussed later)

Note: OS calls main() on execution

Pop quiz 1a on functions

Q. What happens when you compile and run the following code?

```
#include<iostream>
int cube(int i) {
    int j = i * i * i;
    return j;
}

int main() {
    int N; std::cin >> N;
    for (int i = 1; i <= N; i++)
        std::cout << i << " " << cube(i);
}
```

Pop quiz 1a on functions

Q. What happens when you compile and run the following code?

```
#include<iostream>
int cube(int i) {
    int j = i * i * i;
    return j;
}

int main() {
    int N; std::cin >> N;
    for (int i = 1; i <= N; i++)
        std::cout << i << " " << cube(i);
}
```

A. Takes N from user, then
prints cubes of integers from 1 to N

Pop quiz 1b on functions

Q. What happens when you compile and run the following code?

```
#include<iostream>
int cube(int i) {
    int i = i * i * i;
    return i;
}

int main() {
    int N; std::cin >> N;
    for (int i = 1; i <= N; i++)
        std::cout << i << " " << cube(i);
}
```

Pop quiz 1b on functions

Q. What happens when you compile and run the following code?

```
#include<iostream>
int cube(int i) {
    int i = i * i * i;
    return i;
}

int main() {
    int N; std::cin >> N;
    for (int i = 1; i <= N; i++)
        std::cout << i << " " << cube(i);
}
```

A. Won't compile. Argument variable `i` is declared and initialized for function block, so the name cannot be reused.

Pop quiz 1c on functions

Q. What happens when you compile and run the following code?

```
#include<iostream>
int cube(int i) {
    i = i * i * i;
}

int main() {
    int N; std::cin >> N;
    for (int i = 1; i <= N; i++)
        std::cout << i << " " << cube(i);
}
```

Pop quiz 1c on functions

Q. What happens when you compile and run the following code?

```
#include<iostream>
int cube(int i) {
    i = i * i * i;
}

int main() {
    int N; std::cin >> N;
    for (int i = 1; i <= N; i++)
        std::cout << i << " " << cube(i);
}
```

A. Won't compile. Need return statement.

Pop quiz 1d on functions

Q. What happens when you compile and run the following code?

```
#include<iostream>
int cube(int i) {
    i = i * i * i;
    return i;
}

int main() {
    int N; std::cin >> N;
    for (int i = 1; i <= N; i++)
        std::cout << i << " " << cube(i);
}
```

Pop quiz 1d on functions

Q. What happens when you compile and run the following code?

```
#include<iostream>
int cube(int i) {
    i = i * i * i;
    return i;
}

int main() {
    int N; std::cin >> N;
    for (int i = 1; i <= N; i++)
        std::cout << i << " " << cube(i);
}
```

A. Works. The `i` in `cube()` is

- Declared and initialized as an argument.
- Different from the `i` in `main()`.

BUT changing values of function arguments is sufficiently confusing to be deemed bad style for this course.

Pop quiz 1e on functions

Q. What happens when you compile and run the following code?

```
#include<iostream>
int cube(int i) {
    return i * i * i;
}

int main() {
    int N; std::cin >> N;
    for (int i = 1; i <= N; i++)
        std::cout << i << " " << cube(i);
}
```

Pop quiz 1e on functions

Q. What happens when you compile and run the following code?

```
#include<iostream>
int cube(int i) {
    return i * i * i;
}

int main() {
    int N; std::cin >> N;
    for (int i = 1; i <= N; i++)
        std::cout << i << " " << cube(i);
}
```

A. Works. Preferred (compact) code.