Function introduction in C++

method

- So far, we put all codes in main method. But as problem becomes more complicate, the size of codes become bigger.
- Analog: in a small company, every task is run by CEO. But as business grows, it is a good idea to divide the duties and hire additional employees, let one employee finish a specific task. The CEO then coordinate among corresponding employees to finish the job.
- A method is like an employee, performing a specified task. Specify duty for an employee is like to define a method, that is, what the inputs and expected return of a method, and how to process the inputs to get the return.
- Method main is also an employee, but its duty now changes to call the corresponding methods to finish the job. Previously main method just call Java provided methods. Now we need to write user-defined methods, and call these methods.

Input and return

- Device concentrates on one useful task.
  - For example, a bread machine.
- Input is like the materials needed to feed into the device for it to start working.
- Return is like the results given to the user that uses the device.

Input and Return: quarter changer

Input dollar bill and return quarters

one input,
one return

Input and Return: vending machine

Input: money in bills and coins

Return: a metro card with that amount of money

two inputs, one return

Input and Return: alarm clock

What is input?

What is the return?

one input,
no return

We use alarm time as input, get some audio effect (alarm sound), but no physical return.

What is a method

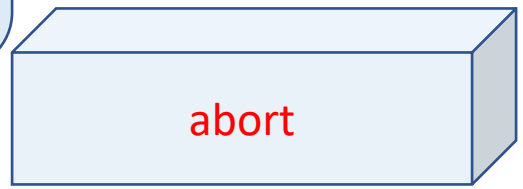
A method like a recipe, it is a set of instructions that process the given input, and return something if necessary.

Components of method

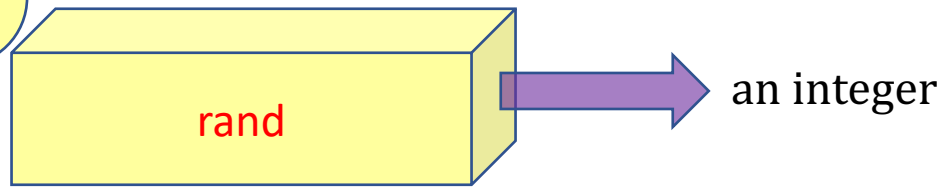
- A method has a meaningful name, normally as a verb (print, toString, …).
- zero, one, or many inputs
- zero or one return
  - If we have more than one return, pack them in a composite type -- a structure composed of several types. So, the return is still one, but it contains many stuffs.
- A method has sor  y.
  - One method normally concentrates only on one task.

Existing methods with no input

Input: none
Return: none
Functionality: Aborts the current process, producing an abnormal program termination.

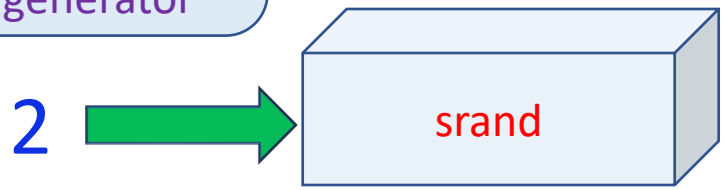


Input: none
Return: an integer
Functionality: Returns a pseudo-random integral number in the range between 0 and RAND_MAX.

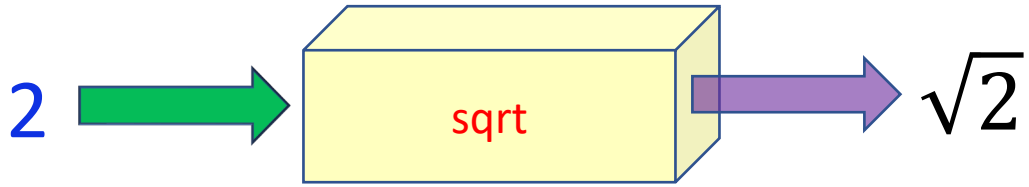


Existing methods with one input

Input: a number
Return: none
Functionality: Initialize random number generator

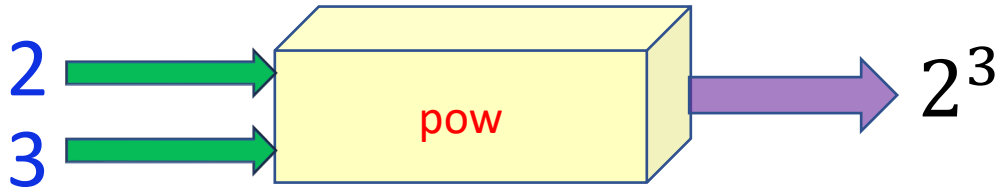


Input: a number
Return: the square root
Functionality: calculate the square root of input

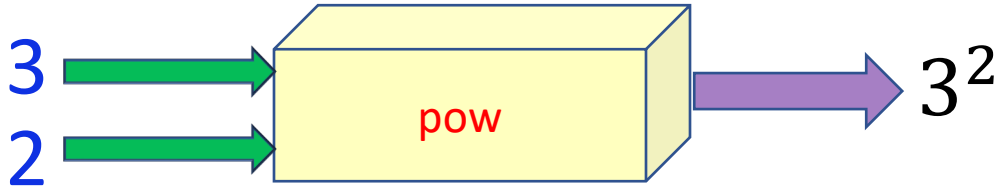


Existing method with two inputs

pow(2, 3) order matters



pow(3, 2)



Have we defined method before?

```
int main() {
	// TODO Auto-generated method stub
	// Your code goes here …
}
```

- What is the name of the method?
- What is the input of the method?
- Does this method return anything?
- Where to define the functionality of the method?

Define a method

```
public static void main(String[] args) {
    // TODO Auto-generated method stub
    // Your code goes here …
}
```

A method has a header, followed by a body.

1. Method header

    1.1 return **type**; **if there is no return**, use void

    1.2 method name (should be meaningful)

    1.3 Declare input parameters inside a pair of parentheses. **If there is no input**, use a pair of empty parentheses.

2. Method body contains code to implement specific functionality, enclosed in a pair of braces {}.

{} is needed even the method body has only one statement.

Why method?

- Abstraction
- Simplify the code
  - Method repeat(String str, int times) simplifies the code in drawing triangle patterns.

Why method: II

- Method main is like CEO of a company.
- When the company expands, more employees are needed.
- A method is like an employee, whose job is to run a specific task.
- A method can call (or use) other methods to finish its functionality. For example, Method main can call nextInt and print methods.

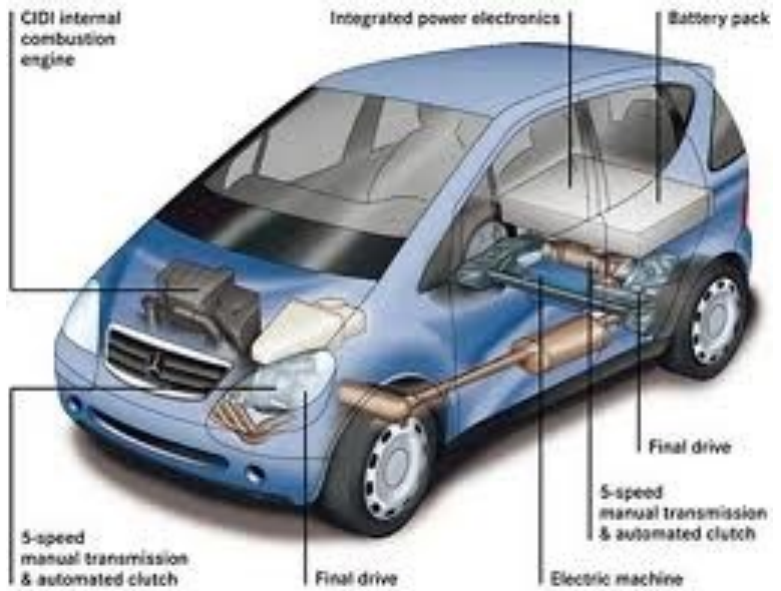
Why method: III

- Modular design: subdivides a system into smaller parts (modules) that can be independently created and used.
- debug (find which method has errors) and test independently
- Expand
- Define once; use many times.
- Define methods for other programmers to use; use methods defined by other programmers.

Why method? IV

To make a car, we need

- Engine
- Tire
- Car head
- Car frame



Each component can be built independently.

If a tire has problem, we only need to check the method to build tires.

Rules for designing method

- It needs experience. Each method has an easily defined singular purpose.
- When reading code from a single method, you should be able to remember what it does from beginning to end.
- Rule of thumb: a method normally contains only 25-30 lines.

Caller vs. callee

- Callee: the method being called
- Caller: the method that calls another method
- The roles of caller and callee are relative.
  - main method calls isPrime method, and isPrime method can call isFactor method.
  - So, isPrime is both a caller and a callee

When a method is called

- Caller saves
  - local variables
  - return point (what is the next instruction to run after finishing calling a method)
- Caller passes parameter list to callee.
- Caller yields control to callee.
- Callee works with the input parameters and returns something if required.
- Callee yields control to caller.
- Caller continue to work from the return point.

An example on calling method

- You call a pizza store. Tell them what kind of pizza you want.
- Someone delivered a pizza to you.
- You get the pizza, then continue to work with it: you may eat it, or bring it to a party, or study for the ingredients… whichever way you like for the pizza.
- Start from 18th second: https://www.youtube.com/watch?v=D_5Ee4OTt1o
  - Questions: who is a caller and who is a callee?

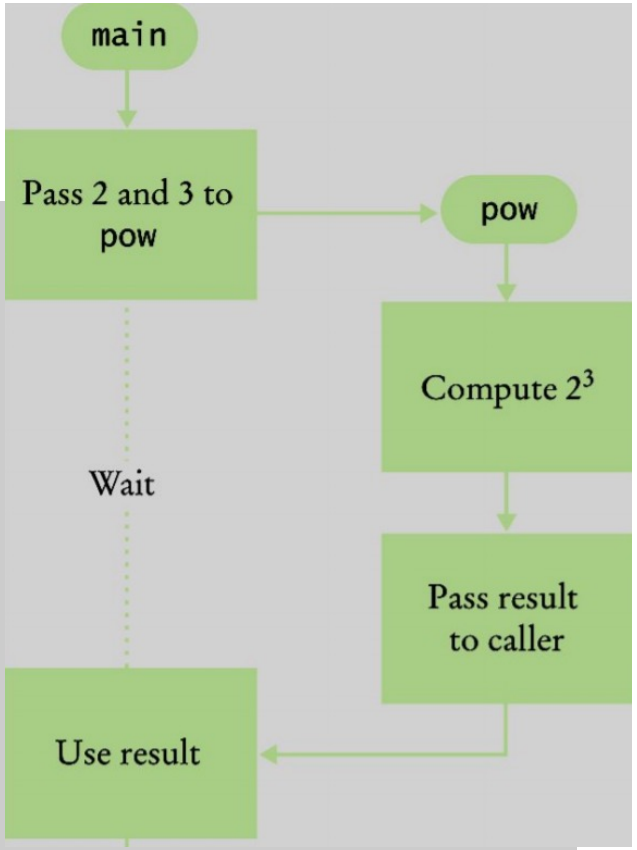
Relation between caller and callee

- Given the following code.

```
#include<iostream>
#include<cmath> //pow
using namespace std;

int main()
{
    double result = pow(2, 3);

    cout << "pow(2, 3) = " << result << endl;
    return 0;
}
```



Related: function in math

- Definition: $f(x) = x^2$, where f is function name, x is parameter, and x^2 is the functionality of a function (aka, what a function can do).
- Call the above function to find out f(3), it will return 3^2, which is 9.
- In math, only work with numbers, but in computer, the types can range from int, double, char, …, to string, and so on.

Function in C++

- In C++, when define a function,

```
return_type function_name(parameter_list)
    //there can be 0, 1, or more parameters,
    //parameter list is type parameter_name separated by comma (,)
{
    //functionality, what to do with the parameters
}
```

- In C++, to call a function

```
functionName(actual_parameter_list); //if return type is not void, can put the return of function to a variable with the same type as return type.
```