Structured Programming

- Functions

Donglong Chen

With Thanks to Dr. Xin Feng and Dr. Haipeng Guo

Outline

- Structured programming
- Function
- Function declaration
- Library functions and user-defined functions
- Function call

Structured Programming

- Structured Programming is a disciplined software development approach
  - Top-down design
  - Stepwise refinement
  - Using 3 basic program structures
    - Sequence
    - Decision
    - Loop

This is actually only for the programming of a function

Function

```
int main()
{
  int i = 1;
  int number = 100;
  int sum = 0;
  while (i <= number){
    sum = sum + i;
    i++;
  }
  printf( “the sum of integers from 1 to
      100 is %d”, sum);
}
```

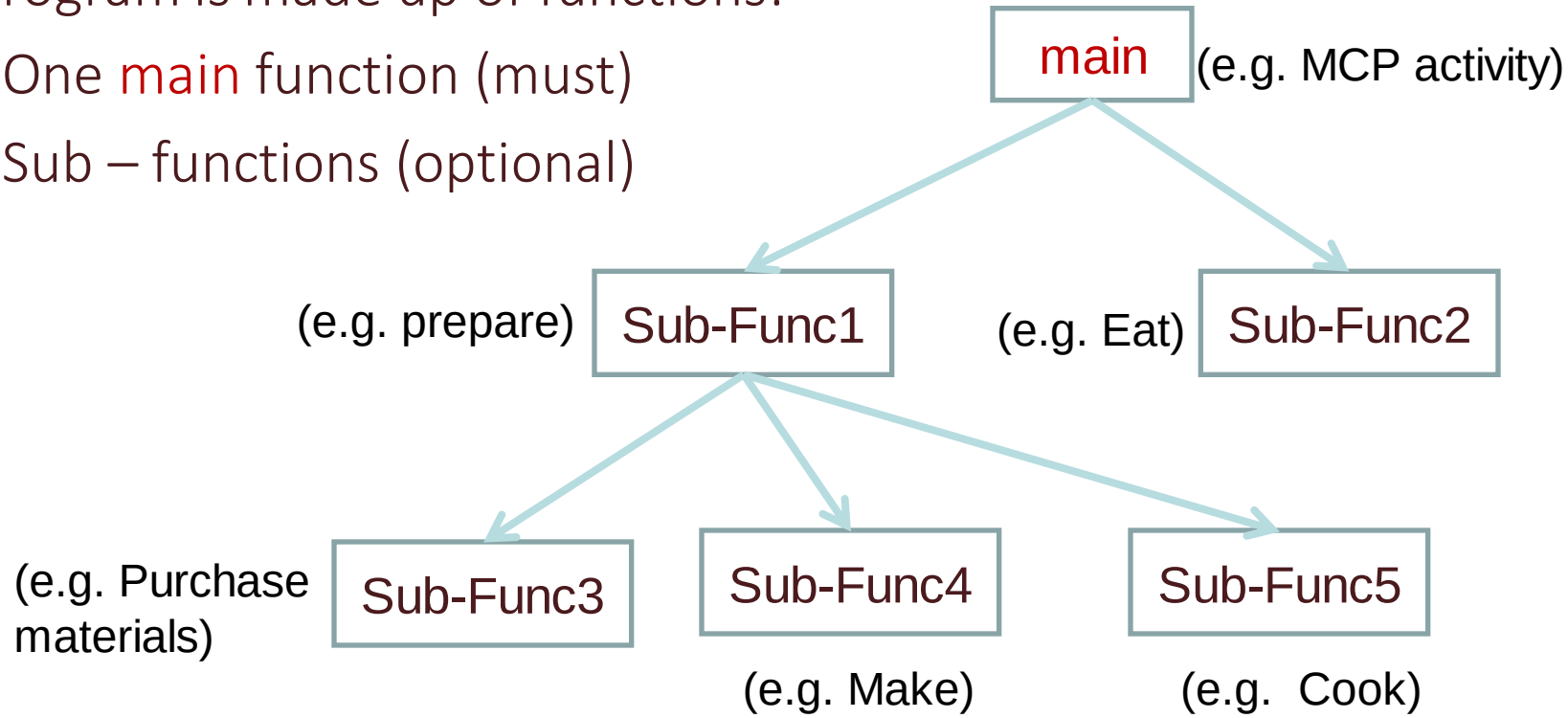
This is a definition of a function with main as its name.

Function

How about the handling of a complex problem?
Can we write a function that include a thousand or more lines?

Structured Programming

- C programming language is a structured programming language
- A program is made up of functions:
  - One main function (must)
  - Sub – functions (optional)



Functions

- A complex problem is often easier to solve by dividing it into several smaller sub-problems (tasks)
  - These smaller sub-problems are sometimes made into packaged code chunks called functions in C.
  - A C program is a collection of functions.

Function

```
int main()
{
  int i = 1;
  int number = 100;
  int sum = 0;
  while (i <= number){
    sum = sum + i;
    i++;
  }
  printf( “the sum of integers from 1 to
      100 is %d” , sum);
}
```

Function

- To write a function, we normally need to specify
  - function declaration (prototype)
  - function definition

Function Declaration

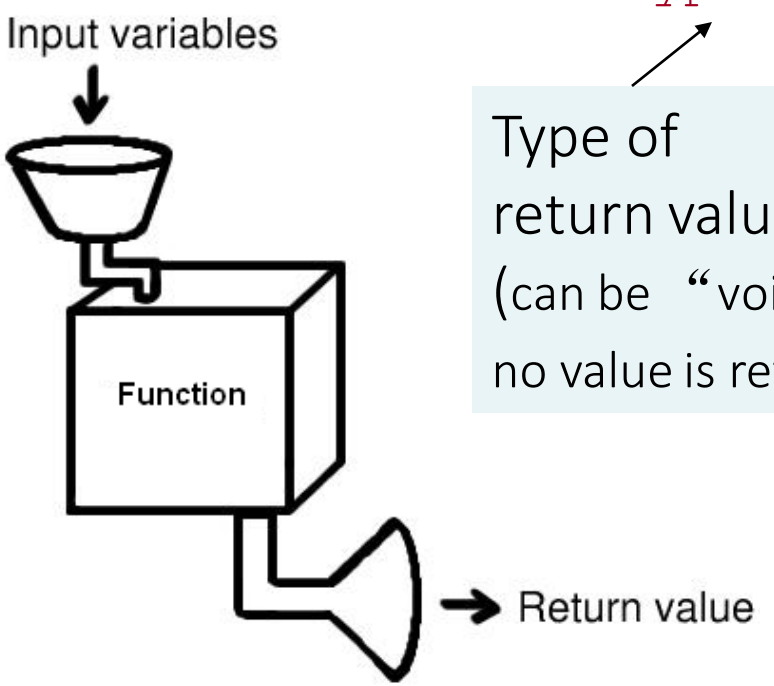
A function is a block of program code which deals with a particular task.

Function prototype syntax:

```
<type> <function name>(<type list>);
```

Type of return value (can be "void" if no value is returned)

List of types of input variables (can be empty if there is no input, can also be written as same as first line of function defintion)



A function declaration is also called a function prototype.

An Example of Function Prototype Exercises #1

- `float averageGrade(int grade1, int grade2);`
  - Function name?
  - How many input variables?
  - What are the types of input variables?
  - What is the type of the return value (output value)

An Example of Function Prototype Exercises #2

- `void printASCIICode(char code);`
  - Function name?
  - How many input variables?
  - What are the types of input variables?
  - What is the type of the return value (output value)

Function Name

- A function name must be meaningful
  - If we want to write a function to calculate the average of grades, which of the following function names is better?
    - averageGrade
    - Abc
    - aG
    - avaragegrade
    - ag

Function Definition

- The function definition can be placed anywhere in the program after the function prototypes.
- Syntax

```
<type> <function name>(<formal parameter list>){
      <local declarations>
      <sequence of statements>
}
```

Examples of Function Definition

```
int sum(int oprand1, int oprand2)
{
  int s;
  s = operand1 + operand2;
  return s;
}
```

```
int absolute(int x){
  if (x >= 0)
    return x;
  else
    return -x;
}
```

```
void printASCIICode(char code)
{
  printf("%c 's ASCII code is %d", code, code);
  return;
}
```

Function Call

- A function can be called to process a task
- Syntax

```
<function name>(<actual parameter list>)
```

The name of the function that has been defined

The list of the input values in the same sequence as defined

- If the function definition is placed before it is called
  - No function prototype is needed
- If the function definition is placed after it is called
  - The function prototype is required

An Example of Function Call Exercise #3

```
void printASCIICode(char);
//void printASCIICode(char code);


int main()
{
  printASCIICode('A');
  printASCIICode('C');
  printASCIICode('F');
}


void printASCIICode(char code)
{
  printf("%c 's ASCII code is %d\n",code, code);
  return;
} Run this program and submit your CPP and outputs
```

Function declaration

Function calls

Function definition

Three Steps in Defining and Using a Function

- First, declare a function using a function prototype
  - `<type> name(<type1>,<type2>,…,<typen>);`
- Second, define the function
  - `<type> name(<type1 p1>, <type2 p2>, …, <typen pn>)`
- Third, call the function
  - `name(v1, v2,…, vn);`

Another Example of Function Call Exercise #4

```
#include <stdio.h>
int absolute(int);
int main()
{
      int value, answer;
      printf("Input an integer (positive or negative) : ");
      scanf("%d", &value);
      answer = absolute(value);
      printf("The absolute value is %d.\n", answer);
      return 0;
}
int absolute(int x){
     if (x >= 0)
       return x;
     else
       return -x;
}
```

Function **declaration**

Function **call**

Function **definition**

Compile this program and submit your CPP and answer using two numbers 16 and -20

The Structure of C Programs

```
Part 1:   preprocessing statements

Part 2:   function prototypes

Part 3:   main() function

Part 4:   user-defined functions
```

Library Functions

- The standard C library (18 Standard C Headers) contains a large collections of functions that can be called from a C program.
  - math.h
  - stdio.h
  - stdlib.h
  - string.h
  - time.h
  - ……
- Search from Internet for more information.

https://www.tutorialspoint.com/c_standard_library/index.htm
https://www.programiz.com/c-programming/library-function

Library Functions

- A library is a collection of precompiled object files which can be linked into programs
- Three steps
  - Find the header file which includes the library function to call
  - Use include directive #include to encompass the header file
  - Call the function in the program

```
#include<stdio.h>
#include<math.h>
int main(){
  double x = sqrt(2.0); //sqrt: declared in math.
  printf("x = %f", x); //printf: declared in stdio.h
  return 0;
}
```

Library Functions
Exercise #5

- Based on the previous example, write a C program to input a number and calculate the:
  1) Square Root (using sqrt()) and
  2) Log base 10 (using log10())
  3) Natural log (using log())
- Using number 10 to test for it.

Write and compile this program and submit your CPP and answer using 10 for testing.

Advantages of Functions

- Functions make programs easier to understand.
- Functions can be called several times in the same program, allowing the code to be reused.

Advantages of Functions

```
#include <stdio.h>
int absolute(int);

int main()
{
      int value1, value2;
      int answer1, answer2;
      scanf("%d %d", &value1, &value2);
      answer1 = absolute(value1);
      answer2 = absolute(value2);
      printf("The absolute values are %d, %d.\n", answer1,
              answer2);
      return 0;
}

int absolute(int x){
    if (x >= 0)
      return x;
    else
     return -x;
}
```

Think about it ….
If we do not use function calls, how to write this program?

Summary

- A program is comprised of one or more functions. Functions make the program more modular.
- A function has declaration, definition and call.
- A function can have parameters. It can also return a value.
- Actual parameters in the function call should be in the same order as formal parameters in the function declaration and definition.
- Variable names can be ignored in function declaration.

Please submit your class exercises Ex1-5 into iSpace before deadline.