Introduction to C Programming

— Functions · Global Variables —

Waseda University

Today's Topics

- Creating Functions
  - How to define a function
  - Calling a function
  - Local variables
  - Scope of a local variable
- Use global variables
  - declaring a global variable
  - Scope of a global variable
  - Arrays as global variables

```
#include <stdio.h>
Type_of_function Name(Parameters){
      Description
      return Return_value;  /*Give the return value of this function*/
}
int main(void){
      double a;
      a = Name(Parameters);  /*Call to the function Name*/
}
```

- Each function has Parameters and Return value.
- Define a function before `#include` statement after `main` function. (You can use the function after you define it.)
- Function name should not be the same as reserved words.
- Functions exit with the return statement.
- Multi parameters are accepted but the return value should be one.
- When we call a function, assign Parameters to the function and get a return value.

Making a linear function

```c
#include <stdio.h>

int main(void){
    double a[3];
    a[0]=2.5*0+1.0;
    a[1]=2.5*1+1.0;
    a[2]=2.5*2+1.0;
    return 0;
  }
```

```c
#include <stdio.h>
double linear(int x){
     double y;
     y = 2.5*x+1.0
     return y; /* Return value */
}
int main(void){
    double a[3];
    a[0]=linear(0);
    a[1]=linear(1);
    a[2]=linear(2);
    return 0;
  }
```

- Avoid writing many times the common parts “2.5*n+1.0”.

Making a linear function

```c
#include <stdio.h>

int main(void){
    double a[3];
    a[0]=2.5*0+1.0;
    a[1]=2.5*1+1.0;
    a[2]=2.5*2+1.0;
    return 0;
  }
```

```c
#include <stdio.h>
double linear(int x){
     double y;
     y = 2.5*x+1.0
     return y; /* Return value */
}
int main(void){
    double a[3];
    a[0]=linear(0);
    a[1]=linear(1);
    a[2]=linear(2);
    return 0;
  }
```

- Avoid writing many times the common parts “2.5*n+1.0”.

More remarks on functions

- All programs of C language consist of functions.
  - `int main(void){...}` is also a function.
  - Every program need the main function.
- Types of functions
  - main function (called at the beginning of the program)
  - Standard functions (with `#include` statement)
  - Functions for obtaining the return value (ex. $\sin(x)$, $\log(x)$)
  - Operations in the functions with no return value
    (ex. printf("Hello, world\n"); etc.)
- Why making functions?
  - Functions allow us to group commonly used code into a compact unit.
  - We can write a program in units easy to understand.

Exercise

Exercise: maximum function of four variables

- When we input a x that is type of double, compute x, $-x$, x^2, $\sqrt{|x|}$ and returns the maximum value of them. (dmax.c)
- Output of this program is as follows:

  Input x: -0.5 【Enter】
  Answer is 0.707107.

A hint of exercise

- You make a function $\max(a, b)$ that returns a bigger value of a and b. We call this function in the main function.

```
double max(double a, double b){
  if (a<b) return  ··· ;
  else     return  ··· ;
}
```

- Absolute value is given by "`fabs()`" function, square root is given by "`sqrt()`" function, respectively.
- When you compile with the math library,
  (1) `#include <math.h>` statement is needed,
  (2) -lm option is necessary.

A hint of exercise

Here is an example of the main function.

the main function

```
int main(void){
  double x, y;
  printf("Input x:");
  scanf("%lf",&x);
  y = max(x, -x);
  y = max(y, x*x);
    ...
  printf("Answer is %f.¥n",y);
  return 0;
}
```

- A variable x is an input value.
- The y is assigned a temporary value of the maximum.
- Call the max function repeatedly and assign each return values to y.

Parameters and Variables

- When a value of the parameter is changed in a function, the input value is never changed in the main function.
- You can declare a variable of the same name in different functions but these are considered as a different thing.

```
#include <stdio.h>
void func(int x){    /* x is declared only in this function. */
  x=7;              /* The x below is different from this x. */
}
int main(void){
  int x=3;
  func(x);  /* Parameter x is 3 and copies 3 to the x above. */
  printf("x is %d.¥n", x);                    /* Here "x is 3." */
  return 0;
}
```

Scope of variables

- The scope of a variable is the area of the program in which the variable is valid.
- A local variable (including parameters) has a scope that is limited to the function in which it is declared.
- You can declare a local variable with the same name as another local variable in a different function. These are considered as different.

```
double functionA (int x, int y){
    ...                              /* The scope of  x and  y. */
}
int functionB(int x){
  double y;
    ...                              /* The scope of  x and  y. */
}
```

Global variables

⇒A global variable is valid everywhere, so its scope is the whole program.

- Declared outside of functions (usually after `#include` or `#define` statement)
- Scope: entire program

```
#include <stdio.h>
double ParamA, ParamB;                    /* 1. Global variables */

double linear(double x){
    return ParamA*x+ParamB;  /* 3. Global can be used here. */
}

int main(void){
    ParamA=5.0; ParamB=3.2;               /* 2. Assigned values */
    printf("Result is %f.¥n", linear(1.2));
    printf("Result is %f.¥n", linear(2.0));
    return 0;
}
```

Arrays as global variables

Huge arrays must be declared as global variables.

ex.

```
int main(void){
    double Array[2000000000];
      /* Huge array cannot declare as a local variable */
      ...
```

- Arrays as global variables are initialized all 0 arrays.
- The huge array is not recommended unless it is necessary.

```
double Array[2000000000] = {1.0, 2.0};    /* No way... */
```

Examples of functions

`int method1(int a, int b)`
— Parameters: a, b (type of `int`)
Type of return value: `int`

`double method2(int a, double b)`
— Parameters: a, b (type of `int` and `double` respectively)
Type of return value: `double`

`boolean method3(int a, int b, int c)`
— Parameters: a, b, c (type of `int`)
Type of return value: `bloolean` (`true` or `false`)

`void method4(double x)`
— Parameters: `x` (type of `double`), Type of return value: none

`double method5(void)`
— Parameters: none, Type of return value: `double`

Summary

- Creating Functions
  - How to define a function
  - Calling a function
  - Local variables
  - Scope of a local variable
- Use global variables
  - declaring a global variable
  - Scope of a global variable
  - Arrays as global variables