VG101 Recitation Class - Week 6

Zhang Yichi

UM-SJTU Joint Institute

June 20, 2018

Short Introduction to C

In the lectures before, you must have heard about the concepts of "translated language" and a "compiled language".

- Program written in translated languages (or called scripting languages) will run sentence by sentence. When a program meets an error, it will stop, with all previous steps completed.
- Program written in compiled languages will first be transformed into an executable file (the process is called compilation), and then this executable file will be run.

That means that when you have some errors in programming, the code will not be able to run even a little bit.

C is actually a compiled language. What's more,it has some other differences from Matlab:

- The speed of running a C program will be much higher than MATLAB
- C programming will be closer to the architecture (or say, the hardware issues) of the computers.
- The previous feature offers more freedom for programmers, as well as higher risk for the programs to collapse. So you should be very careful when programming.

How to Compile/Run a C Program

To run a C program, we must have a C compiler (gcc). The specific command to compile the codes in to a executable file is :

gcc -o exec_name f1.c f2.c

exec_name is the name of the executable file generated. You can name it wahtever you want. The things follows are the names of all ".c" files that you write for the program. When you want to run the generated executable file, you just type:

./exec_name

If you are using IDEs, they will offer you convenient ways of compiling/running programs on them. On Clion, you should also include the files you want to compile in the CMakeList.txt

A Simple Template of a C Program

```
1   #include<stdio.h>
2
3   int factorial(int n);
4
5   int main(){
6     int n;
7     scanf("%d",&n);
8     printf("%d\n",factorial(n));
9     return 0;
10  }
11
12  int factorial(int n){
13    if (n==0) return 1;
14    return n*factorial(n-1);
15  }
16
```

When you want to write your program in multiple files.

- Usually we will separate the main function with the other functions, and we tend to put functions which do similar tasks or operate on same objects in the same file.
- Header files are used to put the function declarations, some #include and #define commands and the definition of other data types(for example, structures) in a header file
- When you use a header file, please note to use a header guard for it.

Template of a Header guard

```
1    #ifndef MYHEADER_H
2    #define MYHEADER_H
3     ...
4    #endif
5
```

- The name MYHEADER_H should be switched into a unique name for every of your header file. A good way is to change the characters in the name of your header file into upper cases and switch ".h" into "_H"
- The purpose of setting a header file is to prevent your header file from being included for multiple times.

Data Types in C

As you have seen in the lecture, there are mainly those categories of data types:

- Constant: Data whose value cannot be changed throughout your program.(We often use #define to declare a constant data, but there is another specifier for variables called const, please pay attention on their differences)
- Global: Data which can be used by all functions, including main() and the subfunctions. To define one, write the declaration at the outside of all functions.(Not recommended, and you will get a code quality deduction if you use one.)
- Local: Data that can be used in just one function.

And the main data types in C are listed below:

- int: represent an integer value
- float: number with decimals, with single precision
- double: number with decimals, with double precision
- char: character values (in fact, "char" can also be regarded as integers varying in a very short range.)
- void: represent something that is not a value ("void" is often used in the type of the return value of a function to represent a function with no return value)

Data Type Specifiers

To change the performance or behaviour of a certain variable, we can use some specifiers on its data types:

- signed/unsigned: variables with an "unsigned" type can only be assigned a positive or zero value, but such type will occupy less storage.
- long/short: extend/shorten the numeric range of a certain data type, and will cause storage increase/decrease at the same time.
- static: static specifier will prevent a value from disappearing when the function ends. If the function is called for a second time, the variable will be initialized as the value it was at the end of the previous call.

- register: a register specifier will hint the compiler to store the variable in the CPU register, so that the read/write speed will be faster. It is used when a certain variable is going to be used for many times in the function. But "register" cannot be added together with "static"
- extern: an extern specifier will hint the compiler that the definition of the variable might appear in the codes afterwards or in other files. However, such definition must be a global variable definition, so you might not need to use it.
- volatile: a volatile specifier will hint the compiler that the variable might be changed by unpredictable process, so the compiler would not do any optimization on the access of this variable to guarantee stability.

Some Limitations About Different Data Types

- Numeric Range: Data types have their minimum values and maximum values. In the questions in JOJ, sometimes the range of the input samples will be specified, so you should take care about selecting a proper data type (or use proper specifiers)
- Precision: For the float and double data, they have their precision of representing a decimal number. This might cause problem when you try to compare whether two float/double numbers are equal.

Data Type char: Use Numbers to Represent Characters

The data type char actually stores an integer.
It actually stores the (ASCII) code for a certain characters.

Frequently-used ASCII-Character Pairs

49 - '1'
65 - 'A'
97 - 'a'

And when you want to store a string in C, you should use array of char variables.

Structures

Structures is one of the very important part in C programming, because it offers you a way to describe a complex object using the combination of different data types.

```
1    typedef struct _person{
2      char *name;
3      int age;
4    } person;
5
```

And you are also able to implement functions for those structures, so that you can have a more efficient way to describe the behaviour of the complex object.

Example

Use a structure to represent a triangle on a plane coordinate system, and implement three functions to do the following three tasks.

- Calculate the area of the triangle.
- Move the triangle on the plane, given a certain dx and dy.
- Rotate the triangle by 90° around its center(average of the three coordinates of the three points) in the clockwise direction.

Note that your structure definition should be specific enough to determine a unique triangle on the plane. For the latter two functions, use a function whose return value is a structure which is the result of the operation.

```c
#include<stdio.h>
#include<math.h>

typedef struct _triangle{
  double x[3];
  double y[3];
} triangle;

double area(triangle a);
triangle move(triangle a,double dx,double dy);
triangle rotate(triangle a);

```

```
1   double area(triangle a){
2     double s1 = sqrt((a.x[1]-a.x[2])*(a.x[1]-a.x[2])+(a.
    y[1]-a.y[2])*(a.y[1]-a.y[2]));
3     double s2 = sqrt((a.x[1]-a.x[0])*(a.x[1]-a.x[0])+(a.
    y[1]-a.y[0])*(a.y[1]-a.y[0]));
4     double s3 = sqrt((a.x[0]-a.x[2])*(a.x[0]-a.x[2])+(a.
    y[0]-a.y[2])*(a.y[0]-a.y[2]));
5     double p = (s1+s2+s3)/2;
6     return sqrt(p*(p-s1)*(p-s2)*(p-s3));
7   }
8
9   triangle move(triangle a,double dx,double dy){
10    triangle b;
11    for (int i=0;i<3;i++){
12      b.x[i]=a.x[i]+dx;
13      b.y[i]=a.y[i]+dy;
14    }
15    return b;
16  }
17
```

```
1   triangle rotate(triangle a){
2     triangle b;
3     double center_x = (a.x[0]+a.x[1]+a.x[2])/3;
4     double center_y = (a.y[0]+a.y[1]+a.y[2])/3;
5     for (int i=0;i<3;i++){
6       b.x[i] = center_x + (a.y[i]-center_y);
7       b.y[i] = center_y - (a.x[i]-center_x);
8     }
9     return b;
10  }
11
```

Basic Syntax

- Blocks: Block is a group of variables that are run together. If you want to have multiple sentences that should be run in conditional statements or loops,you should use a block.
- Operator: There are some "shorhand" operators such as $++$ and $+=$, carefully use them can bring convenience.
- Goto Statement: It is sometimes convenient but in most cases it will make your code unorganized and unreadable. So you are not allowed to use it in this course.

Conditional Statement

```
1   if (condition){
2     statements;
3   }
4   else{
5     statements;
6   }
7   switch(variable){
8     case case1:
9       statements;
10      break;
11    case case2:
12      statements;
13      break;
14    ...
15    default:
16      statements;
17      break;
18  }
19
```

Logical Operation

- There is no boolean(logical) variables in C, the conditional statements will judge whether the value of the condition is 0.
- Unary Operators: !
- Binary Operatiors on Numbers: $==, !=, >, <, >=, <=$
- Binary Operatiors on Logicals: &&,||(they are short-circuit operators)
- Ternary Operatior: a?b:c

Loops

```
1   while (condition){
2     statements;
3   }
4
5   do {
6     statements;
7   }while(condition);
8
9   for (init;condition;step){
10    statements;
11  }
12
```

- The difference between a do...while loop and a while loop is that a do...while loop doesn't check the condition in the first loop.
- Be careful not to cause infinite loops.
- Use continue to directly go to the next loop and break to end the whole looping process.

Array

Definition

An array is a group of data which is of the same type

To declare an array, you should specify its type, name and size. And to initialize it, you should use curly brackets to contain all the elements.

```
1     int a[10];
2     int b[5] = {1,2,3,4,5};
3
```

- Index of array: The i_{th} element in the array a is actually stored as a[i-1], so if the size of an array is n, then its indeces varies between 0 and $n-1$

Pass an Array into a Function

To take an array as an argument, you should use syntax like the following. Usually a size argument is necessary.

```
1    int function(int arr[],int size);
2
```

NOTE: By passing an array into a function, you can actually change the value of it. But you cannot change the value of a single variable by directly passing it into a function (your operations of changing it will only work inside that function).

Website For Reference

C Reference in cppreference.com:
http://en.cppreference.com/w/c