VG101: Introduction to Computer and Programming

Week7 Checklist

A short introduction about Linux

History

- DOS vs. Unix
  - DOS -> MS-DOS -> Windows
  - Unix: MINIX, Linux, macOS
- CLI (Command Line Interface) vs. GUI (Graphical User Interface)

Some useful command in command line (Bash)

- `.`
- `..`
- `cd`: `cd pathname`: change directory
- `ls`: `ls directory`: list files & folders under a directory
- `mkdir`: `mkdir dir`: make directory
- `rmdir`: `rmdir dif`: remove directory
- `rm`: `rm file`: remove file
- `cat`: `cat file`: concatenate files
- `diff`: `diff file1 file2`: compare the difference of two files
- `pwd`: print working directory
- `man`: open manual page
- `apt-get install xxx` (for Ubuntu) or `brew install xxx` (for macOS)
- `./executable_filename`

Compiling in command line

- What does "compile" mean?
- Compiler
  - `gcc`
  - `clang`
- Command:
  - `gcc -o executable_filename source_code_filename.c`
  - May add some options for compiling
- Example
  - Compiling command on JOJ: `gcc -O2 -Wall -Wextra -Werror -pedantic -Wno-`

`unused-result -std=c11 -lm -o /out/helloworld /in/helloworld.c`

- Text + Command Line Compiling vs. IDE

Overview C

C

- A Lower-level higher-level language.
- A procedure-oriented programming language.
- Good for system programming.

C and C++

- C++ is based on C.
- C++ is An object-oreiented programming language.
- C++ integrates many modern programming ideas.

Before we start

- Be prepared to get rid of bad programming habits.
- Submitting program on JOJ that result to an error = Submitting nothing.
- Warning == Error on JOJ.
- Start early.

Features of C

```c
/* Filename: helloWorld.c */
#include <stdio.h>
int main()
{
    printf("Hello world!\n");
    return 0;
}
```

- Filename: end with `.c`
- Include header files (libraries)
- A main function **with an int return value**
- `;` at the end of each line
- `{}` as block
- Different comment style
    - inline comment

```c
// comment in line
```

```
- block comment
```

```
/*
\(0.0)/
comment as a block
\(=.0)-
*/
```

Header files

- What is source files (.c) and header files (.h)?
- Library: functions written by professional programers.
- Common libraries:

```
#include <stdio.h>   // For standard input and output.
#include <math.h>    // For basic mathematic operations like sqrt, sin,
pow.
#include <stdlib.h>  // For random number generation and dynamic memory
allocation.
#include <string.h>  // For efficient string manipulation.
#include <time.h>    // For recording program running time.
```

- Why we need to include library?
- Function: Declaration vs. Definition.
- Include: link multiple files (including library files).

```
#include <stdio.h>
#include <math.h>
#include "myFunctions.h"
```

- Header files: functions declarations written by you.
- Difference between `<>` and `""`

`main` function

- `main` function is where a program starts executing.
- A program only contains **one** main function.
- `main` function is not callable for you. (No recursion for `main` function)
- Return type and return value for `main` should be `int` and `0` (non-zero return value is to indicate some errors)

```
int main() // Return type: a integer
{
    // Write your code here
    return 0; // Return value: 0
}
```

- Your program will fail on JOJ if you don't add return type and return value.
- Please **don't** write everything in the main function.

Variable

- A variable need a name, a data type, a value
- Variable name rules similar to MATLAB
- Requires **declaration before use**
    - Why variable declaration is needed?

```
int a;
float b;
unsigned int c;
char d;
long e;
```

Data types

- char: 8 bits ASCII code representing a character
- int: not guaranteed how many bytes, commonly 4 bytes (32 bits)
- unsigned int
- long
- float & double (both floating point)
- No bool type (logical type) in C. Number 0 = False. Number not 0 = True.

Data type conversion

- C will automatically convert data type if necessary.

```
int a = 1 + 1.5;      // What is a?
double b = 3 / 2;       // What is b?
double c = 3 / 2.0;     // What is c?
int d = a + b + c;    // What is d?
char e = 'a';
e = e + 1;
```

- You can manually convert data type.

```
double pi = 3.1415926;
int integerPi = (int)pi; // Convert double into int
```

- Tips: some automatical type conversion may result in some information loss (e.g. long long -> int). It will generate warning, and in JOJ, every warning will be regarded as error :) So in this case, do it type conversion manually to tell the compiler that you indeed intend to do this conversion.
- All the constant in C will be an `int` by default, and `double` for float point number

Constant variable

- Require initial value assigned, or the variable is meaningless.

- Changing the value of constant variable causes error.
- Why constant variable? Sometimes we want to garantee we will not mistakely change variable that should remain constant. In future learning, you will see many usage of `const` and find it is more like a promise that "I will not change this value."

```
const int a = 1; // Constant variable must has initial value assigned
a = 2;           // Error!
```

Input / Output

- You must include `<stdio.h>`.
- Output function: `printf`
- Input function: `scanf`

`printf`

- Similar to MATLAB function `fprintf`.
- Only print to standard output stream (`stdout`).
- Use C function `fprintf` for file output.

```
#include <stdio.h>
int main()
{
    printf("Hello world!\n");                           // string uses
double quotation mark
    int a = 0; float b = 1; double c = 2; char d = 'd'; // char uses
single quotation mark
    printf("a=%d\tb=%f\tc=%lf\td=%c.\n", a, b, c, d);   // \t for tab, \n
for new line
    return 0;
}
```

`scanf`

- Similar to MATLAB function `fscanf`.
- Only scan from standard input stream (`stdin`).
- Use C function fscanf for file input.
- Visual studio may require you use `scanf_s` but it is not a standard library function. It will fail compiling on JOJ.

```c
#include <stdio.h>
int main()
{
    int a; float b; double c; char d;
    printf("Please input a, b, c, d:\n");              // scanf does not
support output
    scanf("%d, %f, %lf, %c", &a, &b, &c, &d);          // don't forget &
    //scanf("%d %f %lf %c", &a, &b, &c, &d);           // Whats the
difference?
    printf("a=%d\tb=%f\tc=%lf\td=%c.\n", a, b, c, d);
    return 0;
}
```

Operators

- Assignment operator: `=`

```c
int a, b;
a = (b = 1); // Never write anything like this unless you want to get shot
by your college
```

- Arithmatic: `+`, `-`, `*`, `/`
- Remainder: `%`
- No `^`, use `pow()`

Syntax sugar (to be lazy)

- increment: `++`
- decrement: `--`
- `i++` vs. `++i`

```c
int i=0;
printf("%d\n", i++); // Print out 0
```

```c
int i;
i = (i++) + (++i); // DO NOT write anything like this in real life
```

- Shorthand assignment operators
  - a += b ⇔ a = a + b
  - a -= b ⇔ a = a - b
  - a *= b ⇔ a = a * b
  - a /= b ⇔ a = a / b
- (condition) ? expression1 : expression2

```c
max = (x > y) ? x : y;
```

Logical operators

operator	operation
&&	And
||	Or
!	Not
==	Equal to
!=	Not equal to

- different with `&` and `|`

Control statements

- `if`, `switch`, `while`, `for`. Similar to MATLAB.
- blocks {}: enclose multiple statesments
- Variable scope: inside block.

```
#include <stdio.h>
int main()
{
    if (1) // An if statement they will always excute
    {
        int a = 1;
    }
    printf("a = %d", a); // Wrong! variable a no longer exists!
    return 0;
}
```

- block `{}` can be omitted if it only enclose one statement.
- Again, **Indent is useful!**

`if`

- No major difference with MATLAB
- No elseif, use else if

`switch`

- Fast way to deal with multiple situations.
- break is important. What will happen without break?

```
#include <stdio.h>
int main()
{
```

```
    int n;
    printf("What is the rank of the card (1-13)? ");
    scanf("%d", &n);
    switch (n)
    {
        case 1: printf("Ace\n"); break; // break is important
        case 11: printf("Jack\n"); break;
        default: printf("%d\n", n); break;
    }
    return 0;
}
```

`while`

- No major difference with MATLAB
- `Do-while`: another version of while loop.
- `while(1)` for infinite loop.
- `break` & `continue`: same as MATLAB

`for`

- Most common loop statement
- How to use?

```
#include <stdio.h>
int main()
{
    int i;
    for (i=0; i<10; i++) //Initial value; ending condition; step
        printf("%d\n", i);
    return 0;
}
```

- The above code equals code below:

```
#include <stdio.h>
int main()
{
    int i=0;
    while (i<10)
    {
        printf("%d\n", i);
        i++;
    }
    return 0;
}
```

- What is the scope of variable `i` if we do this?

```
for (int i=0; i<10; i++)
    DoSomething();
```

- A common mistake

```
for (int i=0; i<10; i++);
    DoSomething();
```

Function

- Allow code written outside main function
- Syntax:

```
int add(int a, int b)   // returnType functionName (parameterType
parameter, ...)
{                       // block enclose function content
    return (a + b);     // return value (must consistent with returnType)
}
```

- parameter passed by value
- same variable name can be used in different function (they do not share value)
- Use void function if function return nothing.
- Use void parameter if function needs no input parameters. (can be omitted)

```
void helloWorld(void)           // No input parameter, no return value.
void helloWorld() is also okay
{
    printf("Hello world\n");    // Only do something.
}
```

Use function in main

- Function prototype (definition) should appear above where it is used.
- Function implementation can appear after main function.

```
#include <stdio.h>
int add(int a, int b);          // Function prototype, ; required

int main()
{
    int a, b, c;
    scanf("%d, %d", &a, &b);
    c = add(a, b);              // Call the function, return value stored
into c
    printf("a+b=%d\n", c);
    return 0;
}
```

```
int add(int a, int b)           // Function implementation
{
    return (a + b);             // block {} should not be omitted for
function even with only single statement
}
```

- Put prototype and implentation together:

```
#include <stdio.h>
int add(int a, int b)           // Function prototype + implementation, ;
not required
{
    return (a + b);
}

int main()
{
    int a, b, c;
    scanf("%d, %d", &a, &b);
    c = add(a, b);              // Call the function, return value stored
into c
    printf("a+b=%d\n", c);
    return 0;
}
```

- Both are OK in this case.
- Sometimes you have to use the former
    - Using function in other `.c` files, prototype written in a `.h` file.
    - Two functions calling each other.