

CONFIDENTIAL

C Programming Introduction

week 2: Introduction to C programming language

Dept of Software Engineering
Hanoi University of Technology

For HEDSPI Project

Topic of this week

- C programming language
 - Class Lecture Review
 - C language structure
 - compiling and running programs
 - keywords
 - Programming Exercises



What is a computer program?

- A sequence of processor instructions designed to achieve a specific purpose.
- The instructions are executed sequentially.
- Each instruction has a numerical code.



Examples of instructions

- Load data (from an address in the memory)
- Store data (in an address)
- Add two numbers
- If two numbers are equal, jump to another part of the program
- Instructions are numbers!

C Language Structure

- General format

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

main() { → Begin
    [function-body];
} → End
:
type func() {
    [function-body];
}
```

Preprocessor / Library include

Function main: [declaration-list] + [statement-list]

Semicolon

Function func: [declaration-list] + [statement-list]

C Language Structure (Cont)

- The first C program (hello.c)

```
#include <stdio.h>
int main() {
    printf("Hello CP\n");
    return 0;
}
```

C Language Structure (Cont)

- `#include <stdio.h>`
 - To declare using the standard I/O library. Other libraries: string, time, math...
- `int main()`
 - To declare the `main()` function. An C program must declare only one `main()` function. The first line in the `main()` will implement when the program starts.
- `{ ... }`
 - The syntax to open and close a block of codes.
- `printf`
 - the `printf()` function sends the output to standard output (monitor). This function will be taught in the next week.
- `return 0;`
 - Stop the program.

C Language Structure (Cont)

- Another example C code

```
#include <stdio.h>
main() {
    int sum;    /* Variable declaration */
               /* sum is a variable hold the
               sum of two integer */
    sum = 75 + 25; /* Value assignment */
    printf("The sum of 75 and 25 is %d\n", sum);
}
```

The sum of 75 and 25 is 100



Keywords of C

- Flow control (6) – `if`, `else`, `return`, `switch`, `case`, `default`
- Loops (5) – `for`, `do`, `while`, `break`, `continue`
- Common *types* (5) – `int`, `float`, `double`, `char`, `void`
- *structures* (3) – `struct`, `typedef`, `union`
- Counting and sizing things (2) – `enum`, `sizeof`
- Rare but still useful *types* (7) – `extern`, `signed`, `unsigned`, `long`, `short`, `static`, `const`
- Evil keywords which we avoid (1) – `goto`
- Wierdies (3) – `auto`, `register`, `volatile`



Compiling with gcc

- GNU C Compiler
- Available in the OS Linux
- Perform one or more of the following
 - C pre processing
 - Compilation
 - Linking



Basic gcc examples

- `gcc hello.c` (compile `hello.c` produce executable `a.out`)
- `gcc -o hello hello.c` (compile `hello.c` produce executable `hello`)
- `gcc -o hello hello.c other.c` (compile `hello.c` and `other.c` produce executable `hello`)



Using intermediate files

- From any source file, you can produce an object file to be linked in later to an executable

```
gcc -c hello.c
gcc -c other.c
gcc -o hello hello.o other.o
```



Other important gcc options

- `-g`: include debugging symbols in the output
- `-l<name>`: include a library
- For example, to use mathematic library of ANSI C: `gcc -lm`



Exercise 2.1

- *Use gcc to compile* the file `hello.c` in previous exercise last week.
- To view what the program do, run:
`./a.out`

If the Program has an Error

```
/* Your name - your class */  
/* This is my first program in C */  
#include <stdio.h>  
main(  
    {  
        printf("Welcome to C Programming  
        Introduction.\n");  
    }
```

- If this program is compiled, we get the message:
- hello.c : in function 'main'
- hello.c:4: parse error before '}'

Line number

How to correct the mistake?

- Open the "hello.c" in emacs
- Identify the errors, and fix them
- Save the modified file
- Compile it again and then run it



Exercise 2.2

- Use *gcc* to compile the file `hello.c` in previous exercise to an executable program named `sayhello`
- Run the `sayhello`:
`./sayhello`



Exercise 2.3

- Use *emacs* to modify `hello.c` as follow. Then save file with the name `hello1.c`

```
/* Your name - your class */  
/* This is my second program in C */  
  
#include <stdio.h>  
main()  
{  
    printf("Welcome to C");  
    printf("Programming Introduction.\n");  
}
```

- Use *gcc* to compile `hello1.c` to a file named `hello1`.
- Run this file and view if the result is different with `hello`?



Exercise 2.4

- Write a program as below then compile it to a executable file and run to view the result:

```
/* Your name – your class */
/* This is my second program in C */

#include <stdio.h>
main()
{
    printf("Welcome to C\n");
    printf("Programming Introduction.\n");
}
```



Exercise 2.5

- Now try to write yourself a program that print a sentence that introduce your self. And say hello to the user.
- For example:

```
*****
My name is Binh Nguyen.
Nice to meet you.
Hope you will have happy time
*****
```

Exercise 2.6

- Edit the following program and save it as pi.c. Compile it to pi.out and run. Place all the files into your directory **week2**. Check that you understand the purpose and output of this program.

```
#include <stdio.h>
#define PI 3.142

main()
{
    double r, c, ac, as, v;
    r = 5.678;
    printf("Radius = %f\n", r);
    c = 2.0 * PI * r;
    printf("Circle's circumference = %f\n", c);
    ac = PI * r * r;
    printf("Circle's area = %f\n", ac);
    as = 4.0 * PI * r * r;
    printf("Sphere's area = %f\n", as);
    v = 4.0/3.0 * PI * r * r * r;
    printf("Sphere's volume = %f\n", v);
}
```

Exercise 2.7

1. Write a program that writes a program that writes the name of the person sitting next to you.
2. compile and run your program; redirect its output to neighbor.c



Exercise 2.8: Review by algorithm

- Write an algorithm specifying the procedure to create a simple program.
- Your input: a computer.