202018021
Fu, Ziyu
Programming I Chapter 1 Task 1, 2, 3
2021/04/29

Task 1

1. Objective
   To understand the sample code provided in the textbook and explore the meaning of "¥n" in strings.

2. Strategy of solving
   - Inspect the structure of the provided sample code
   - Run the code
   - Observe the output

3. Program code

```c
#include <stdio.h>
int main(){
   printf("Today I am learning \n the C language.\n");
   printf("\n I am wondering if it is easy or not,");
   printf("however,trying hard \n as far as I can.");
   return 0;
}
```

4. Results and discussions
   Console output (modified program):

```
Today I am learning
 the C language.

 I am wondering if it is easy or not,however,trying hard
```

   Console output (original program):

```
Today I am learning ¥n the C language.¥n¥n I am wondering if it is easy or
not,however,trying hard ¥n as far as I can.
```

   The console prints everything in one line with **¥n** included in the string, but when the output string is changed to include **\n** instead, it is not hard to see that this particular mark acts like pressing a `Enter` key on the keyboard, which escapes the current line and starts a new one.

   \n is the standard general escape sequence for end of line in many modern programming languages including C.

   How **\n** can be confused with **¥n** is a very interesting story.

   The backslash sign(\) was originally assigned to codepoint `5C` in the 7-bit US-ASCII standard in 1960, but some early encoding standards in Japan (like JIS X 0201 in 1969) reassigned `5C` to the yen sign(**¥**), as the backslash character was less used.

With the arrival of 8-bit encoding, the yen sign(**¥**) was properly assigned codepoint `A5`, according to ISO/IEC 8859-1 in 1985, and Unicode today continues this standard by assigning yen sign to `U+00A5`.

However, because the JIS standard was widely adopted in Japan, Microsoft decided to keep 0x5C as the yen sign in Japanese-locale fonts on Windows, regardless of what it was intended to be. Because of that, things like the directory separator(for example, **c:¥users¥**), or the general escape character(**¥n**), are all mapped incorrectly to the yen sign on Japanese computers. The won sign (₩) has similar issues in Korean versions of Windows. Most other OSs today besides Windows adopt the Unicode standard, which doesn't have this problem.

Task 2

1. Objective
   - To understand the structure and hierarchy of functions in a simple C program
   - To create a program that outputs a square calculation

2. Strategy of solving
   - Read the task statement and understand the inputs and outputs
   - Read the sample code given in the chapter and fully understand it
   - Analyze the problem and redesign the program
   - Modify the program and test
   - Match the results to desired outputs

3. Program code

```c
#include <stdio.h>

int addup(int i);

int main(){
   int i = 1;
   int sum;
   sum = addup(i);
   // printf("i was squared - it now equals %d\n",sum);
}

int addup(int i) {
   while (i <= 5){
       printf("i is %d\n", i*i);
       i = i + 1;
   }
   // i = i * i;
   return (i);
}
```

4. Results and discussions

   Console output:

```
i is 1
i is 4
i is 9
i is 16
i is 25
```

   The original `addup()` function iterates the input `i` and prints it out in each iteration until up until 6 (5+1 because the print is post-increment). All I need to do for it to output `i` squared is to make it print `i*i` stead of `i`. I also need to adjust the order of the print and the increment command so that the input is from 1 to 5 instead of 2 to 6. I also commented out the print in the `main()` function as it wasn't necessary any more. The execution outputs the expected results.

Task 3

1. Objective
   To explain the structure of a simple C program by referring to concepts introduced by the text.

2. Strategy of solving
   Read the chapter and sample code

3. Program code
   N/A

4. Results and discussions
   When inspected at a higher level, the structure of most simple C programs can be considered as a collection of definitions of variables, data types, and functions. Inside of each function, you can find a similar structure of variable definitions and function definition. A C program always starts its execution by the `main()` function, and as a result, the leading executable part of a C project also has to include a `main()` function. A C compiler will translate these sets of function, definitions and declarations in (possibly) multiple files into one executable file.