

202018021

Fu, Ziyu

Programming I Chapter 8 Task 1, 2, 3

2021/06/24

Task 1

1. Objective

To make a program that reads and comprehends string user input

2. Strategy of solving

- Read user input and save into `char` array
- Iterate through every position of the array and compare it to the next element
- Change it to `**` if they are the same

3. Program code

```
1  #include <stdio.h>
2
3  char input[256] = "";
4
5  int main(){
6
7      printf("Input string shorter than 256: ");
8      scanf( "%s", input);
9
10     for(int i = 0; input[i] != '\0'; ++i){
11         if (input[i] == input[i+1]){
12             input[i] = '*';
13             input[i+1] = '*';
14         }
15     }
16
17     printf("%s\n", input);
18 }
```

4. Results and discussions

Console output:

```
Input string shorter than 256: TTTeeeSsSsssttt
**T**eSsS**s**t
```

The program outputs as expected. There is an important assumption in this code: the user input through `scanf` cannot exceed 256 characters long. A proper way to fix this is through dynamically securing memory space using `realloc` so that the code can deal with arrays of arbitrary lengths. But with the computing power today, you can get away with allocating very large strings while hardly making a dent in the computer's RAM usage. That's what I ended up doing.

Task 2

1. Objective

To modify the sample program and output a slightly different pattern.

2. Strategy of solving

- Read and understand the task statement
- Read the sample code and the sample output
- Identify the modifications and change the code

3. Program code

```
1  #include <stdio.h>
2
3  int main(){
4      char image[21][21]; /* 2D array declaration */
5      int i, j;
6
7      for (j = 0; j < 21; j = j + 1){ /* make an image with this double loop */
8          float y = j - 10;
9          for (i = 0; i < 21; i = i + 1){
10             float x = i - 10;
11             if (x * x + y * y < 90){
12                 image[j][i] = '#';
13             }else{
14                 image[j][i] = '.';
15             }
16
17             if (x > -5 && x < 5 && y > -5 && y < 5){
18                 image[j][i] = 'x';
19             }
20         }
21     }
22
23     for (j = 0; j < 21; j = j + 1){ /* output an image with this double loop */
24         for (i = 0; i < 21; i = i + 1){
25             printf("%c ", image[j][i]);
26         }
27         printf("\n");
28     }
29 }
```

4. Results and discussions

Console output:

[illegible]

The program outputs the correct pattern. This is a very simple modification to the original sample code. Since the “origin” of this graph is at the very center, we just need to draw a square by modifying the original pattern.

Task 3

1. Objective
To create a program that multiplies two 2 by 2 matrices.
2. Strategy of solving
 - Read and understand the task statement
 - Design the code
 - Observe the outputs
3. Program code

```
1  #include <stdio.h>
2
3  int M[2][2] = {6, 1, 5, 1};
4  int U[2][2] = {1, -1, 2, -2};
5
6  int V[2][2] = {0, 0, 0, 0};
7
8  int main(){
9
10     for(int i = 0; i < 2; ++i){
11         for(int j = 0; j < 2; ++j){
12             V[i][j] = M[i][0] * U[0][j] + M[i][1] * U[1][j];
13         }
14     }
15
16     for(int i = 0; i < 2; i++) {
17         for(int j = 0; j < 2; j++) {
18             printf("%d ", V[i][j]);
19         }
20         printf("\n");
21     }
22
23 }
```

4. Results and discussions

Console output:

8 -8

7 -7

The program outputs correctly. Because the task statement restricts the matrix multiplication calculation to two 2 by 2 matrices, and the two matrices given both have integer elements. The program assumes that input pattern. It will not work if other kinds of matrices are inputted.