

Documentation for pattern printing project

Introduction

The pattern printing project was first made in order to understand the working of C++ programming language and to test our logical thinking by making us to connect what we have learnt so far in our first semester of programming class. But then after we learned modular programming it was necessary to take look back and see the previous project in the new concept, and refactoring was inevitable.

This refactor of previous project on pattern printing was done using the basic concepts of modular programming like of functions, that are used to simplify the previous code by reducing complications and sophistication of activities on the main function and simplify the program by dividing the program in to simpler logical two functions that print the pattern in simple and flowing manner. As with the idea of modular programming which is divide and conquer in the refactor we the project using functions to perform an activity in a function.

- Components of the program

Basic component

In the basic part we have the preprocessor directives. Which is in this case `#include <upstream>....` Which is responsible for the output and input indicator `cout` and `cin` and using namespace `std` to not redundantly use `std` when using input and output insertions.

Used global variable

Here I this code we used global variable to just not use the variable over and over again not to redundantly use a variable over again we prefer using a single common variable used in every code with a same goal. That variable is called “num” and its purpose is to determine the limit of numbers we want to print the amount of numbers we need to be printed as pattern.

Coming to the function part we found three functions here in this code

Two user defined and one built-in which is the “`int main()`” function. Now let us take a closer look of the two user defined functions in detail.

1. **The upper_part function:** This function prints the upper triangle or the upper part of the pattern in this refactored code. In this function we used different variables and we will explain in detail about these as following.

- Firstly the upper_part function is a void function because it doesn't return any thing it simply prints out pattern

Coming to the how the code functions:

```
void upper_part(){
    for(int i=num;i>=0;i--){
        for(int k =num-i;k>0;k--)
            cout<<" ";
```

In this first beginning part the local variable “i” is declared to be equal to the user input “num” and the value of “i” decreases by each iteration. So “i” is basically used to print the row part. Then for every row there is there is local variable “k” which is used to print the space from the left most side of the pattern. So again k also decreases with iteration. And until “k” is greater than zero it prints space.

The second half part of the upper_part function is this:

```
        for(int j=i;j>=0;j--){
            cout<<j;

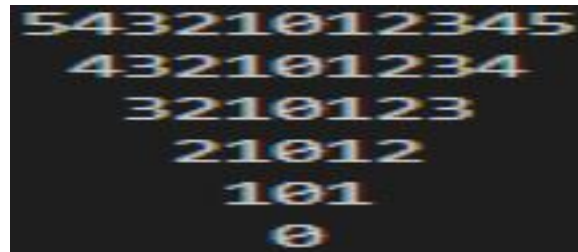
            for(int m=1;m<=i;m++)
                cout<<m;

            cout<<endl;
        }
```

In those lines of code we found local variable “j” which is used to print the first half numbers till zero of the numbers and it goes from the input number all the way to zero meaning decreasing in each iteration.

In the last for loop we found local variable “m” which prints the half other side of the pattern which increases by each iteration by one. It prints until it reaches the input number.

So all to gather the upper part has output if for example we assignee 5 for the input of like this...



2. The lower_part of the function: In this function we used numerous variables which we will explain about it later.

- Firstly this function is void type because it doesn't return anything rather it prints the lower part of the pattern.

```
void lower_part(){  
    for(int n = 1;n<=num;n++){  
        for(int f = num-n;f>0;f--)  
            cout<<" ";  
    }
```

In this function also local variable “n” Is used to determine the row number and the row number increases until it reaches the input number.

Again the local variable “f” is used to print

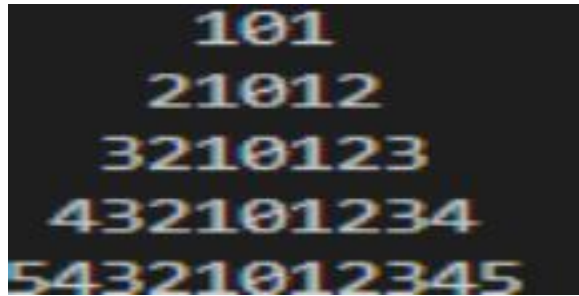
```
        for(int z =n;z>=1;z--)  
            cout<<z;  
        for(int x =0;x<=n;x++)  
            cout<<x;  
        cout<<endl;  
    }
```

In this last part of the lower_part function we have local variable local variable “z” is used to print numbers from the user input up to one and it decreases in each iteration until it reaches one.

The last variable “x” in this code is use to print numbers staring from zero all the way to the input given number.

Let us check the code by inputting 5 then the lower_part function prints:

Printing pattern



```
101
21012
3210123
432101234
54321012345
```

So to conclude, combining these two functions we made a cleaner and simpler code thanks to modular programming.