Function

- Need to perform a same task more than once.
  - Use the same set of statements every time you want to perform the task
  - Create a function to perform that task, and just call it every time you need to perform that task.

- A function is a block of statements that performs a specific task.

- Types of functions
  - Predefined standard library functions – such as printf(), scanf() , rand(), sqrt() etc..
  - User Defined functions

- There are 3 aspects in each C function

- **Function declaration or prototype** – This informs compiler about the function name, function parameters and return value's data type.
- **Function call** – This calls the actual function
- **Function definition** – This contains all the statements to be executed.

Defining a Function

```
return_type function_name( parameter list )
{
……….
body of the function
……..
}
```

Defining a Function using C

```
int addTwoNumbers(int intNumber1,
                              int intNumber2)
{
     int intSum;
     intSum = intNumber1 +  intNumber2 ;
     return intSum;
}
```

Function declaration

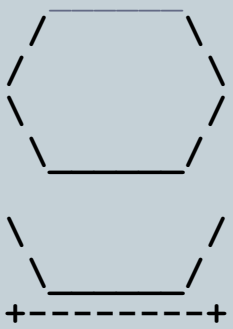
```
#include <stdio.h>

// ------[BEGIN]-------- function declaration  -------
// add two integer values
int addTwoNumbers (int intNumber1, int intNumber2);
// ------ [END] -------- function declaration  -------

int main( void )
{
  ……..
  ……..
  return 0 ;
}
```

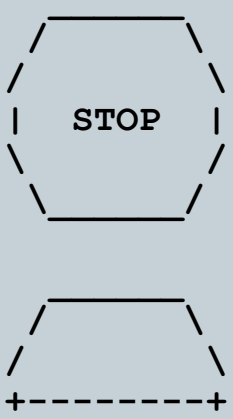
Calling a Function

```
int main( void )
{
    int intNumber1; //store first number input by the user
    int intNumber2; //store second number input by the user
    int intAddition;
    // variable in which addition will be stored
    ……
    ……
       // Calling the function  to Addition 2 integers
       intAddition = addTwoNumbers (intNumber1, intNumber2);
    ……
    ……
    return 0 ;
}
```

Problem-solving methodology

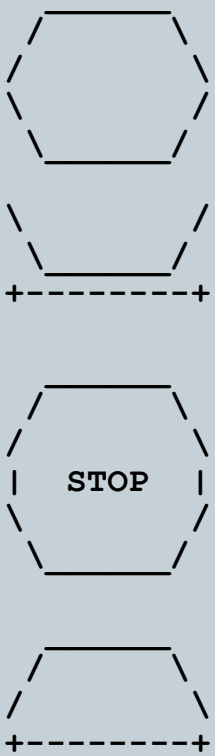
- Write a program to print the following figures





Problem-solving methodology

- Some steps we can use to print complex figures:

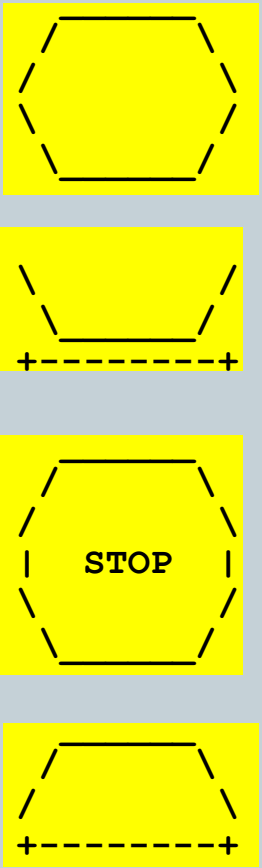


- First version of program (unstructured):
- Create an empty program with a skeletal header and `main` method.
- Write expected output using **printf();** syntax.
- Run our first version and verify that it produces the correct output.

Problem-solving methodology

- DrawPtn_V1.c

Problem-solving 2 answer



The structure of the output:

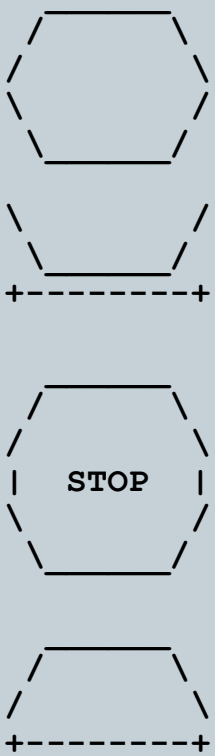
- initial "egg" figure
- second "teacup" figure
- third "stop sign" figure
- fourth "hat" figure

This structure can be represented by methods:

- `drawEgg`
- `drawTeaCup`
- `drawStopSign`
- `drawHat`

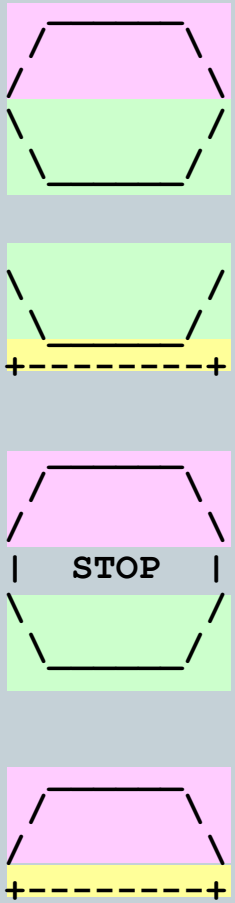
- DrawPtn_V2.c

Problem-solving 3



- Third version of program (structured without redundancy):
- Identify any redundancy in the output, and further divide the program into static methods to eliminate as much redundancy as possible.
- Add comments to the program to improve its readability.

Problem-solving 3 answer



The redundancy in the output:

top half of egg:	reused on stop sign, hat
bottom half of egg:	reused on teacup, stop sign
divider line:	used on teacup, hat

a single line, so making it a method is optional

This redundancy can be fixed by methods:

`drawEggTop`
`drawEggBottom`
`drawLine` (optional)

- DrawPtn_V3.c