# A Gentle Introduction to Reusability, Decision Making and Repeatability

اَللّٰهُمَّ اُرْزُقْنِی عِلْمًا نَافِعًا وَاسِعًا عَمِيْقًا

اَللّٰهُمَّ اُرْزُقْنِى رِزْقًا وَاسِعًا حَلَالًا طَيِّبًا
مُبَارَكًا مِنْ عِنْدِكَ

# A Gentle Introduction to Reusability, Decision Making and Repeatability

# Problem: Print KAKA on the Console

Can you write a program that print KAKA with capital alphabets on the Console.

```
##  ##
## ##
####
###
####
## ##
##  ##


  ##
 ####
##  ##
######
##  ##
##  ##
##  ##


##  ##
## ##
####
###
####
## ##
##  ##


  ##
 ####
##  ##
######
##  ##
##  ##
##  ##
```

# Problem: Print KAKA on the Console

How can i do that?

```
##  ##
## ##
####
###
####
## ##
##  ##


  ##
 ####
##  ##
######
##  ##
##  ##
##  ##


##  ##
## ##
####
###
####
## ##
##  ##


  ##
 ####
##  ##
######
##  ##
##  ##
##  ##
```

# Solution: Print KAKA on the Console

```
#include <iostream>
using namespace std;
main()
{
    cout << " ##  ## " << endl;
    cout << " ## ##  " << endl;
    cout << " ####   " << endl;
    cout << " ###    " << endl;
    cout << " ####   " << endl;
    cout << " ## ##  " << endl;
    cout << " ##  ## " << endl;
    cout << endl << endl;

    cout << "   ##   " << endl;
    cout << "  ####  " << endl;
    cout << " ##  ## " << endl;
    cout << " ###### " << endl;
    cout << " ##  ## " << endl;
    cout << " ##  ## " << endl;
    cout << " ##  ## " << endl;
    cout << endl << endl;
```

```
    cout << " ##  ## " << endl;
    cout << " ## ##  " << endl;
    cout << " ####   " << endl;
    cout << " ###    " << endl;
    cout << " ####   " << endl;
    cout << " ## ##  " << endl;
    cout << " ##  ## " << endl;
    cout << endl << endl;

    cout << "   ##   " << endl;
    cout << "  ####  " << endl;
    cout << " ##  ## " << endl;
    cout << " ###### " << endl;
    cout << " ##  ## " << endl;
    cout << " ##  ## " << endl;
    cout << " ##  ## " << endl;
    cout << endl << endl;
}
```

# Solution: Print KAKA on the Console

```
#include <iostream>
using namespace std;
main()
{
    cout << " ##  ## " << endl;
    cout << " ## ##  " << endl;
    cout << " ####   " << endl;
    cout << " ###    " << endl;
    cout << " ####   " << endl;
    cout << " ## ##  " << endl;
    cout << " ##  ## " << endl;
    cout << endl << endl;

    cout << "   ##   " << endl;
    cout << "  ####  " << endl;
    cout << " ##  ## " << endl;
    cout << " ###### " << endl;
    cout << " ##  ## " << endl;
    cout << " ##  ## " << endl;
    cout << " ##  ## " << endl;
    cout << endl << endl;
```

Is there any code redundant?

```
    cout << " ##  ## " << endl;
    cout << " ## ##  " << endl;
    cout << " ####   " << endl;
    cout << " ###    " << endl;
    cout << " ####   " << endl;
    cout << " ## ##  " << endl;
    cout << " ##  ## " << endl;
    cout << endl << endl;

    cout << "   ##   " << endl;
    cout << "  ####  " << endl;
    cout << " ##  ## " << endl;
    cout << " ###### " << endl;
    cout << " ##  ## " << endl;
    cout << " ##  ## " << endl;
    cout << " ##  ## " << endl;
    cout << endl << endl;
}
```

# Solution: Print KAKA on the Console

Is there any code redundant?

```
#include <iostream>
using namespace std;
main()
{
    cout << " ##  ## " << endl;
    cout << " ## ##  " << endl;
    cout << " ####   " << endl;
    cout << " ###    " << endl;
    cout << " ####   " << endl;
    cout << " ## ##  " << endl;
    cout << " ##  ## " << endl;
    cout << endl << endl;

    cout << "   ##   " << endl;
    cout << "  ####  " << endl;
    cout << " ##  ## " << endl;
    cout << " ###### " << endl;
    cout << " ##  ## " << endl;
    cout << " ##  ## " << endl;
    cout << " ##  ## " << endl;
    cout << endl << endl;
```

```
    cout << " ##  ## " << endl;
    cout << " ## ##  " << endl;
    cout << " ####   " << endl;
    cout << " ###    " << endl;
    cout << " ####   " << endl;
    cout << " ## ##  " << endl;
    cout << " ##  ## " << endl;
    cout << endl << endl;

    cout << "   ##   " << endl;
    cout << "  ####  " << endl;
    cout << " ##  ## " << endl;
    cout << " ###### " << endl;
    cout << " ##  ## " << endl;
    cout << " ##  ## " << endl;
    cout << " ##  ## " << endl;
    cout << endl << endl;
}
```

# Solution: Print KAKA on the Console

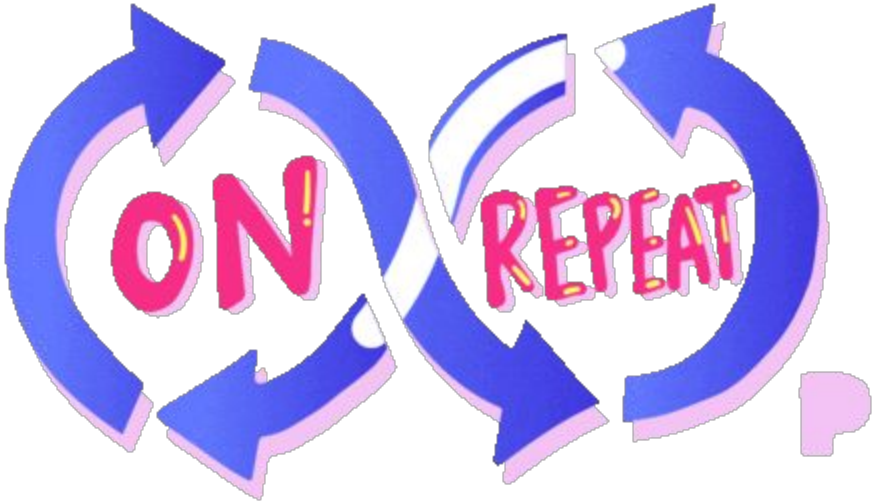
Is there any code redundant?

```
#include <iostream>
using namespace std;
main()
{
    cout << " ##  ## " << endl;
    cout << " ## ##  " << endl;
    cout << " ####   " << endl;
    cout << " ###    " << endl;
    cout << " ####   " << endl;
    cout << " ## ##  " << endl;
    cout << " ##  ## " << endl;
    cout << endl << endl;

    cout << "   ##   " << endl;
    cout << "  ####  " << endl;
    cout << " ##  ## " << endl;
    cout << " ###### " << endl;
    cout << " ##  ## " << endl;
    cout << " ##  ## " << endl;
    cout << " ##  ## " << endl;
    cout << endl << endl;
```

```
    cout << " ##  ## " << endl;
    cout << " ## ##  " << endl;
    cout << " ####   " << endl;
    cout << " ###    " << endl;
    cout << " ####   " << endl;
    cout << " ## ##  " << endl;
    cout << " ##  ## " << endl;
    cout << endl << endl;

    cout << "   ##   " << endl;
    cout << "  ####  " << endl;
    cout << " ##  ## " << endl;
    cout << " ###### " << endl;
    cout << " ##  ## " << endl;
    cout << " ##  ## " << endl;
    cout << " ##  ## " << endl;
    cout << endl << endl;
}
```

# Redundant Tasks

Sometimes in **real world** there are tasks that redundant.



# Redundant Tasks

When we have to do a redundant task, we place the code separately, give a name to it and then use the name whenever we require to do the task again.

# Redundant Tasks

For example, in case of previous example, we can place the code for printing K and A separately, give a name to the code and then call that name to repeat the functionality.

# Redundant Tasks

We have to give a meaningful name to that separate block of code.

```
void printK()
{
    cout << " ##  ## " << endl;
    cout << " ## ##  " << endl;
    cout << " ####   " << endl;
    cout << " ###    " << endl;
    cout << " ####   " << endl;
    cout << " ## ##  " << endl;
    cout << " ##  ## " << endl;
    cout << endl << endl;
}
```

```
void printA()
{
    cout << "   ##   " << endl;
    cout << "  ####  " << endl;
    cout << " ##  ## " << endl;
    cout << " ###### " << endl;
    cout << " ##  ## " << endl;
    cout << " ##  ## " << endl;
    cout << " ##  ## " << endl;
    cout << endl << endl;
}
```

# Functions: Redundant Tasks

The separate block of code that performs a specific task is called a **Function**.

```
void printK()
{
    cout << " ##  ## " << endl;
    cout << " ## ##  " << endl;
    cout << " ####   " << endl;
    cout << " ###    " << endl;
    cout << " ####   " << endl;
    cout << " ## ##  " << endl;
    cout << " ##  ## " << endl;
    cout << endl << endl;
}
```

```
void printA()
{
    cout << "   ##   " << endl;
    cout << "  ####  " << endl;
    cout << " ##  ## " << endl;
    cout << " ###### " << endl;
    cout << " ##  ## " << endl;
    cout << " ##  ## " << endl;
    cout << " ##  ## " << endl;
    cout << endl << endl;
}
```

# Functions: Redundant Tasks

The separate block of code that performs a specific task is called a **Function**.

Function Name

```
void printK()
{
    cout << " ##  ## " << endl;
    cout << " ## ##  " << endl;
    cout << " ####   " << endl;
    cout << " ###    " << endl;
    cout << " ####   " << endl;
    cout << " ## ##  " << endl;
    cout << " ##  ## " << endl;
    cout << endl << endl;
}
```

```
void printA()
{
    cout << "   ##   " << endl;
    cout << "  ####  " << endl;
    cout << " ##  ## " << endl;
    cout << " ###### " << endl;
    cout << " ##  ## " << endl;
    cout << " ##  ## " << endl;
    cout << " ##  ## " << endl;
    cout << endl << endl;
}
```

# Functions: Redundant Tasks

The separate block of code that performs a specific task is called a **Function**.

Function Name

Function return type

```
void printK()
{
    cout << " ##  ## " << endl;
    cout << " ## ##  " << endl;
    cout << " ####   " << endl;
    cout << " ###    " << endl;
    cout << " ####   " << endl;
    cout << " ## ##  " << endl;
    cout << " ##  ## " << endl;
    cout << endl << endl;
}
```

```
void printA()
{
    cout << "   ##   " << endl;
    cout << "  ####  " << endl;
    cout << " ##  ## " << endl;
    cout << " ###### " << endl;
    cout << " ##  ## " << endl;
    cout << " ##  ## " << endl;
    cout << " ##  ## " << endl;
    cout << endl << endl;
}
```

# Functions: Redundant Tasks

The separate block of code that performs a specific task is called a **Function**.

Function Name

Function return type

```
void printK()
{
    cout << " ##  ## " << endl;
    cout << " ## ##  " << endl;
    cout << " ####   " << endl;
    cout << " ###    " << endl;
    cout << " ####   " << endl;
    cout << " ## ##  " << endl;
    cout << " ##  ## " << endl;
    cout << endl << endl;
}
```

Function body

```
void printA()
{
    cout << "   ##   " << endl;
    cout << "  ####  " << endl;
    cout << " ##  ## " << endl;
    cout << " ###### " << endl;
    cout << " ##  ## " << endl;
    cout << " ##  ## " << endl;
    cout << " ##  ## " << endl;
    cout << endl << endl;
}
```

# Functions: Redundant Tasks

We have defined 2 functions. One function with name printK and one function with name printA

```
void printK()
{
    cout << " ##  ## " << endl;
    cout << " ## ##  " << endl;
    cout << " ####   " << endl;
    cout << " ###    " << endl;
    cout << " ####   " << endl;
    cout << " ## ##  " << endl;
    cout << " ##  ## " << endl;
    cout << endl << endl;
}
```

```
void printA()
{
    cout << "   ##   " << endl;
    cout << "  ####  " << endl;
    cout << " ##  ## " << endl;
    cout << " ###### " << endl;
    cout << " ##  ## " << endl;
    cout << " ##  ## " << endl;
    cout << " ##  ## " << endl;
    cout << endl << endl;
}
```

# Functions

Now, let's see how we can call these functions.

```
#include <iostream>
using namespace std;

main()
{

}
```

```
void printK()
{
    cout << " ##  ## " << endl;
    cout << " ## ##  " << endl;
    cout << " ####   " << endl;
    cout << " ###    " << endl;
    cout << " ####   " << endl;
    cout << " ## ##  " << endl;
    cout << " ##  ## " << endl;
    cout << endl << endl;
}
```

```
void printA()
{
    cout << "   ##   " << endl;
    cout << "  ####  " << endl;
    cout << " ##  ## " << endl;
    cout << " ###### " << endl;
    cout << " ##  ## " << endl;
    cout << " ##  ## " << endl;
    cout << " ##  ## " << endl;
    cout << endl << endl;
}
```

# Functions

We just have to write the name of the function in our main function.

```
#include <iostream>
using namespace std;

main()
{

}
```

```
void printK()
{
    cout << " ##  ## " << endl;
    cout << " ## ##  " << endl;
    cout << " ####   " << endl;
    cout << " ###    " << endl;
    cout << " ####   " << endl;
    cout << " ## ##  " << endl;
    cout << " ##  ## " << endl;
    cout << endl << endl;
}
```

```
void printA()
{
    cout << "   ##   " << endl;
    cout << "  ####  " << endl;
    cout << " ##  ## " << endl;
    cout << " ###### " << endl;
    cout << " ##  ## " << endl;
    cout << " ##  ## " << endl;
    cout << " ##  ## " << endl;
    cout << endl << endl;
}
```

# Functions

We just have to write the name of the function in our main function.

```
#include <iostream>
using namespace std;

main()
{
    printK();
}
```

```
void printK()
{
    cout << " ##  ## " << endl;
    cout << " ## ##  " << endl;
    cout << " ####   " << endl;
    cout << " ###    " << endl;
    cout << " ####   " << endl;
    cout << " ## ##  " << endl;
    cout << " ##  ## " << endl;
    cout << endl << endl;
}
```

```
void printA()
{
    cout << "   ##   " << endl;
    cout << "  ####  " << endl;
    cout << " ##  ## " << endl;
    cout << " ###### " << endl;
    cout << " ##  ## " << endl;
    cout << " ##  ## " << endl;
    cout << " ##  ## " << endl;
    cout << endl << endl;
}
```

# Functions

printK() statement will call the function.

```
#include <iostream>
using namespace std;

main()
{
    printK();
}
```

```
void printK()
{
    cout << " ##  ## " << endl;
    cout << " ## ##  " << endl;
    cout << " ####   " << endl;
    cout << " ###    " << endl;
    cout << " ####   " << endl;
    cout << " ## ##  " << endl;
    cout << " ##  ## " << endl;
    cout << endl << endl;
}
```

```
void printA()
{
    cout << "   ##   " << endl;
    cout << "  ####  " << endl;
    cout << " ##  ## " << endl;
    cout << " ###### " << endl;
    cout << " ##  ## " << endl;
    cout << " ##  ## " << endl;
    cout << " ##  ## " << endl;
    cout << endl << endl;
}
```

# Functions

```
##  ##
## ##
####
###
####
## ##
##  ##
```

Following will be the output on the console.

```
#include <iostream>
using namespace std;

main()
{
    printK();
}
```

```
void printK()
{
    cout << " ##  ## " << endl;
    cout << " ## ##  " << endl;
    cout << " ####   " << endl;
    cout << " ###    " << endl;
    cout << " ####   " << endl;
    cout << " ## ##  " << endl;
    cout << " ##  ## " << endl;
    cout << endl << endl;
}
```

```
void printA()
{
    cout << "   ##   " << endl;
    cout << "  ####  " << endl;
    cout << " ##  ## " << endl;
    cout << " ###### " << endl;
    cout << " ##  ## " << endl;
    cout << " ##  ## " << endl;
    cout << " ##  ## " << endl;
    cout << endl << endl;
}
```

# Functions

```
##  ##
## ##
####
###
####
## ##
##  ##
```

Let's call the printA() function now.

```
#include <iostream>
using namespace std;

main()
{
    printK();
    printA();
}
```

```
void printK()
{
    cout << " ##  ## " << endl;
    cout << " ## ##  " << endl;
    cout << " ####   " << endl;
    cout << " ###    " << endl;
    cout << " ####   " << endl;
    cout << " ## ##  " << endl;
    cout << " ##  ## " << endl;
    cout << endl << endl;
}
```

```
void printA()
{
    cout << "   ##   " << endl;
    cout << "  ####  " << endl;
    cout << " ##  ## " << endl;
    cout << " ###### " << endl;
    cout << " ##  ## " << endl;
    cout << " ##  ## " << endl;
    cout << " ##  ## " << endl;
    cout << endl << endl;
}
```

# Functions

```
##  ##
## ##
####
###
####
## ##
##  ##

  ##
 ####
##  ##
######
##  ##
##  ##
##  ##
```

Following will be the output on the console.

```
#include <iostream>
using namespace std;

main()
{
    printK();
    printA();
}
```

```
void printK()
{
    cout << " ##  ## " << endl;
    cout << " ## ##  " << endl;
    cout << " ####   " << endl;
    cout << " ###    " << endl;
    cout << " ####   " << endl;
    cout << " ## ##  " << endl;
    cout << " ##  ## " << endl;
    cout << endl << endl;
}
```

```
void printA()
{
    cout << "   ##   " << endl;
    cout << "  ####  " << endl;
    cout << " ##  ## " << endl;
    cout << " ###### " << endl;
    cout << " ##  ## " << endl;
    cout << " ##  ## " << endl;
    cout << " ##  ## " << endl;
    cout << endl << endl;
}
```

# Functions

```
##  ##
## ##
####
###
####
## ##
##  ##

  ##
 ####
##  ##
######
##  ##
##  ##
##  ##
```

Let's call the printK function again.

```
#include <iostream>
using namespace std;

main()
{
    printK();
    printA();
    printK();
}
```

```
void printK()
{
    cout << " ##  ## " << endl;
    cout << " ## ##  " << endl;
    cout << " ####   " << endl;
    cout << " ###    " << endl;
    cout << " ####   " << endl;
    cout << " ## ##  " << endl;
    cout << " ##  ## " << endl;
    cout << endl << endl;
}
```

```
void printA()
{
    cout << "   ##   " << endl;
    cout << "  ####  " << endl;
    cout << " ##  ## " << endl;
    cout << " ###### " << endl;
    cout << " ##  ## " << endl;
    cout << " ##  ## " << endl;
    cout << " ##  ## " << endl;
    cout << endl << endl;
}
```

# Functions

Following will be the output.

```
## ##
## ##
####
###
####
## ##
## ##

  ##
 ####
## ##
######
## ##
## ##
## ##

## ##
## ##
####
###
####
## ##
## ##
```

```
#include <iostream>
using namespace std;

main()
{
    printK();
    printA();
    printK();
}
```

```
void printK()
{
    cout << " ##  ## " << endl;
    cout << " ## ##  " << endl;
    cout << " ####   " << endl;
    cout << " ###    " << endl;
    cout << " ####   " << endl;
    cout << " ## ##  " << endl;
    cout << " ##  ## " << endl;
    cout << endl << endl;
}
```

```
void printA()
{
    cout << "   ##   " << endl;
    cout << "  ####  " << endl;
    cout << " ##  ## " << endl;
    cout << " ###### " << endl;
    cout << " ##  ## " << endl;
    cout << " ##  ## " << endl;
    cout << " ##  ## " << endl;
    cout << endl << endl;
}
```

# Functions

```
##  ##
## ##
####
###
####
## ##
##  ##


  ##
 ####
##  ##
######
##  ##
##  ##
##  ##


##  ##
## ##
####
###
####
## ##
##  ##
```

Let's call the printA function again.

```
#include <iostream>
using namespace std;

main()
{
    printK();
    printA();
    printK();
    printA();
}
```

```
void printK()
{
    cout << " ##  ## " << endl;
    cout << " ## ##  " << endl;
    cout << " ####   " << endl;
    cout << " ###    " << endl;
    cout << " ####   " << endl;
    cout << " ## ##  " << endl;
    cout << " ##  ## " << endl;
    cout << endl << endl;
}
```

```
void printA()
{
    cout << "   ##   " << endl;
    cout << "  ####  " << endl;
    cout << " ##  ## " << endl;
    cout << " ###### " << endl;
    cout << " ##  ## " << endl;
    cout << " ##  ## " << endl;
    cout << " ##  ## " << endl;
    cout << endl << endl;
}
```

# Functions

Following will be the output

```
##  ##
## ##
####
###
####
## ##
##  ##

  ##
 ####
##  ##
######
##  ##
##  ##
##  ##

##  ##
## ##
####
###
####
## ##
##  ##

  ##
 ####
##  ##
######
##  ##
##  ##
##  ##
```

```
#include <iostream>
using namespace std;

main()
{
    printK();
    printA();
    printK();
    printA();
}
```

```
void printK()
{
    cout << " ##  ## " << endl;
    cout << " ## ##  " << endl;
    cout << " ####   " << endl;
    cout << " ###    " << endl;
    cout << " ####   " << endl;
    cout << " ## ##  " << endl;
    cout << " ##  ## " << endl;
    cout << endl << endl;
}
```

```
void printA()
{
    cout << "   ##   " << endl;
    cout << "  ####  " << endl;
    cout << " ##  ## " << endl;
    cout << " ###### " << endl;
    cout << " ##  ## " << endl;
    cout << " ##  ## " << endl;
    cout << " ##  ## " << endl;
    cout << endl << endl;
}
```

# Functions

Here, printK and printA are the called Functions and main is the calling function

```
#include <iostream>
using namespace std;

main()
{
    printK();
    printA();
    printK();
    printA();
}
```

```
void printK()
{
    cout << " ##  ## " << endl;
    cout << " ## ##  " << endl;
    cout << " ####   " << endl;
    cout << " ###    " << endl;
    cout << " ####   " << endl;
    cout << " ## ##  " << endl;
    cout << " ##  ## " << endl;
    cout << endl << endl;
}
```

```
void printA()
{
    cout << "   ##   " << endl;
    cout << "  ####  " << endl;
    cout << " ##  ## " << endl;
    cout << " ###### " << endl;
    cout << " ##  ## " << endl;
    cout << " ##  ## " << endl;
    cout << " ##  ## " << endl;
    cout << endl << endl;
}
```

# Functions

Now, the question is **where to write** these functions?

```
#include <iostream>
using namespace std;

main()
{
    printK();
    printA();
    printK();
    printA();
}
```

```
void printK()
{
    cout << " ##  ## " << endl;
    cout << " ## ##  " << endl;
    cout << " ####   " << endl;
    cout << " ###    " << endl;
    cout << " ####   " << endl;
    cout << " ## ##  " << endl;
    cout << " ##  ## " << endl;
    cout << endl << endl;
}
```

```
void printA()
{
    cout << "   ##   " << endl;
    cout << "  ####  " << endl;
    cout << " ##  ## " << endl;
    cout << " ###### " << endl;
    cout << " ##  ## " << endl;
    cout << " ##  ## " << endl;
    cout << " ##  ## " << endl;
    cout << endl << endl;
}
```

# Functions

In C++, the code of function declaration should be before the function call.

```
#include <iostream>
using namespace std;

main()
{
    printK();
    printA();
    printK();
    printA();
}
```

```
void printK()
{
    cout << " ##  ## " << endl;
    cout << " ## ##  " << endl;
    cout << " ####   " << endl;
    cout << " ###    " << endl;
    cout << " ####   " << endl;
    cout << " ## ##  " << endl;
    cout << " ##  ## " << endl;
    cout << endl << endl;
}
```

```
void printA()
{
    cout << "   ##   " << endl;
    cout << "  ####  " << endl;
    cout << " ##  ## " << endl;
    cout << " ###### " << endl;
    cout << " ##  ## " << endl;
    cout << " ##  ## " << endl;
    cout << " ##  ## " << endl;
    cout << endl << endl;
}
```

# Functions

In C++, the code of function declaration should be before the function call.

```
#include <iostream>
using namespace std;
void printK()
{
    cout << " ##  ## " << endl;
    cout << " ## ##  " << endl;
    cout << " ####   " << endl;
    cout << " ###    " << endl;
    cout << " ####   " << endl;
    cout << " ## ##  " << endl;
    cout << " ##  ## " << endl;
    cout << endl << endl;
}
void printA()
{
    cout << "   ##   " << endl;
    cout << "  ####  " << endl;
    cout << " ##  ## " << endl;
    cout << " ###### " << endl;
    cout << " ##  ## " << endl;
    cout << " ##  ## " << endl;
    cout << " ##  ## " << endl;
    cout << endl << endl;
}
main()
{
    printK();
    printA();
    printK();
    printA();
}
```

# Functions

If we write the code of function declaration after the function call then we will get the following error.

```
printName.cpp: In function 'int main()':
printName.cpp:6:12: error: 'printK' was not declared in this scope
     printK();
            ^
printName.cpp:7:12: error: 'printA' was not declared in this scope
     printA();
            ^
```

```cpp
#include <iostream>
using namespace std;
main()
{
    printK();
    printA();
    printK();
    printA();
}

void printK()
{
    cout << " ##  ## " << endl;
    cout << " ## ##  " << endl;
    cout << " ####   " << endl;
    cout << " ###    " << endl;
    cout << " ####   " << endl;
    cout << " ## ##  " << endl;
    cout << " ##  ## " << endl;
    cout << endl << endl;
}
void printA()
{
    cout << "   ##   " << endl;
    cout << "  ####  " << endl;
    cout << " ##  ## " << endl;
    cout << " ###### " << endl;
    cout << " ##  ## " << endl;
    cout << " ##  ## " << endl;
    cout << " ##  ## " << endl;
    cout << endl << endl;
}
```

# Functions

However, if we want to define a function **after** the function call, we need to use the **function prototype**.

```
#include <iostream>
using namespace std;
main()
{
    printK();
    printA();
    printK();
    printA();
}

void printK()
{
    cout << " ##  ## " << endl;
    cout << " ## ##  " << endl;
    cout << " ####   " << endl;
    cout << " ###    " << endl;
    cout << " ####   " << endl;
    cout << " ## ##  " << endl;
    cout << " ##  ## " << endl;
    cout << endl << endl;
}
void printA()
{
    cout << "   ##   " << endl;
    cout << "  ####  " << endl;
    cout << " ##  ## " << endl;
    cout << " ###### " << endl;
    cout << " ##  ## " << endl;
    cout << " ##  ## " << endl;
    cout << " ##  ## " << endl;
    cout << endl << endl;
}
```

# Functions

However, if we want to define a function **after** the function call, we need to use the **function prototype**.

```
#include <iostream>
using namespace std;
void printK();
void printA();
main()
{
    printK();
    printA();
    printK();
    printA();
}
void printK()
{
    cout << " ##  ## " << endl;
    cout << " ## ##  " << endl;
    cout << " ####   " << endl;
    cout << " ###    " << endl;
    cout << " ####   " << endl;
    cout << " ## ##  " << endl;
    cout << " ##  ## " << endl;
    cout << endl << endl;
}
void printA()
{
    cout << "   ##   " << endl;
    cout << "  ####  " << endl;
    cout << " ##  ## " << endl;
    cout << " ###### " << endl;
    cout << " ##  ## " << endl;
    cout << " ##  ## " << endl;
    cout << " ##  ## " << endl;
    cout << endl << endl;
}
```

Function Prototype

# Functions

However, if we want to define a function **after** the function call, we need to use the **function prototype**.

```
#include <iostream>
using namespace std;
void printK();
void printA();
main()
{
    printK();
    printA();
    printK();
    printA();
}
void printK()
{
    cout << " ##  ## " << endl;
    cout << " ## ##  " << endl;
    cout << " ####   " << endl;
    cout << " ###    " << endl;
    cout << " ####   " << endl;
    cout << " ## ##  " << endl;
    cout << " ##  ## " << endl;
    cout << endl << endl;
}
void printA()
{
    cout << "   ##   " << endl;
    cout << "  ####  " << endl;
    cout << " ##  ## " << endl;
    cout << " ###### " << endl;
    cout << " ##  ## " << endl;
    cout << " ##  ## " << endl;
    cout << " ##  ## " << endl;
    cout << endl << endl;
}
```

Function Prototype

Function Definition

# Problem: Add 2 Numbers

Write a C++ program that takes 2 numbers from user as input and adds them and then display the output on the console.

```
G:\Programming Fundamentals (Fall 2022)\Week 4>addNumbers.exe
Enter First Number: 4
Enter Second Number: 5
Sum is: 9
```

# Solution: Add 2 Numbers

Beforing writing the solution with separate function, let's write the solution in main function.

```
#include <iostream>
using namespace std;
main(){
    int number1, number2, sum;
    cout << "Enter First Number: ";
    cin >> number1;
    cout << "Enter Second Number: ";
    cin >> number2;
    sum = number1 + number2;
    cout << "Sum is: " << sum;
}
```

# Class Practice: Add 2 Numbers

Write a C++ Function that takes 2 numbers from user as input and adds them and then display the output on the console.

```
G:\Programming Fundamentals (Fall 2022)\Week 4>addNumbers.exe
Enter First Number: 4
Enter Second Number: 5
Sum is: 9
```

# Solution updated: Add 2 Numbers

```
#include <iostream>
using namespace std;

void add();

main()
{
    add();
}

void add()
{
    int number1, number2, sum;
    cout << "Enter First Number: ";
    cin >> number1;
    cout << "Enter Second Number: ";
    cin >> number2;
    sum = number1 + number2;
    cout << "Sum is: " << sum;
}
```

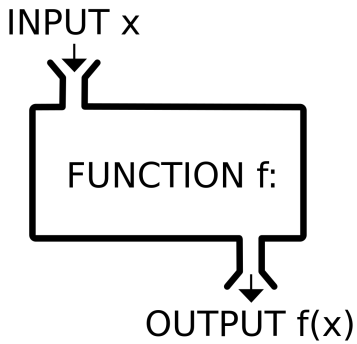
# Solution updated: Add 2 Numbers

There is another improvement in the definition of function.

```
#include <iostream>
using namespace std;

void add();

main()
{
    add();
}

void add()
{
    int number1, number2, sum;
    cout << "Enter First Number: ";
    cin >> number1;
    cout << "Enter Second Number: ";
    cin >> number2;
    sum = number1 + number2;
    cout << "Sum is: " << sum;
}
```

# Functions: Definition

A **function** is a block of code that performs a specific task.
We give **inputs** to the function, it performs some calculations on it, and returns the **output**.



# Problem Updated: Add 2 Numbers

Write a C++ Function that takes 2 numbers from user as its parameters, adds them and then display the output on the console.

```
G:\Programming Fundamentals (Fall 2022)\Week 4>addNumbers.exe
Enter First Number: 4
Enter Second Number: 5
Sum is: 9
```

# Solution updated: Add 2 Numbers

Function Name

Input Parameters

Function Return type

```
void add(int number1, int number2)
{
    int sum;
    sum = number1 + number2;
    cout << "Sum is: " << sum;
}
```

Function Body

# Solution updated: Add 2 Numbers

```
#include <iostream>
using namespace std;

void add(int number1, int number2);

main()
{
    int number1, number2;
    cout << "Enter First Number: ";
    cin >> number1;
    cout << "Enter Second Number: ";
    cin >> number2;

    add(number1, number2);
}
```

Function Call

```
void add(int number1, int number2)
{
    int sum;
    sum = number1 + number2;
    cout << "Sum is: " << sum;
}
```

# Solution updated: Add 2 Numbers

```
#include <iostream>
using namespace std;

void add(int number1, int number2);

main()
{
    int number1, number2;
    cout << "Enter First Number: ";
    cin >> number1;
    cout << "Enter Second Number: ";
    cin >> number2;

    add(number1, number2);
}
```

Function Prototype

```
void add(int number1, int number2)
{
    int sum;
    sum = number1 + number2;
    cout << "Sum is: " << sum;
}
```

Function Call

# Class Practice: Add and Multiply 2 Numbers

Now, we want the user to Enter two numbers and first your function should add the two numbers and then multiply those two numbers.

```
Enter First Number: 5
Enter Second Number: 4
Sum is: 9
Product is: 20
```

# Solution Updated: Add and Multiply 2 Numbers

```
#include <iostream>
using namespace std;

void add(int number1, int number2);
void multiply(int number1, int number2);

main()
{
    int number1, number2;
    cout << "Enter First Number: ";
    cin >> number1;
    cout << "Enter Second Number: ";
    cin >> number2;

    add(number1, number2);
    multiply(number1, number2);
}
```

```
void add(int number1, int number2)
{
    int sum;
    sum = number1 + number2;
    cout << "Sum is: " << sum << endl;
}

void multiply(int number1, int number2)
{
    int product;
    product = number1 * number2;
    cout << "Product is: " << product << endl;
}
```

# A Gentle Introduction to Reusability, Decision Making and Repeatability

# Problem Updated: Add or Multiply 2 Numbers

Now, we want the user to Enter two numbers and then tell that whether he/she wants to add the numbers or multiply the numbers.

```
Enter First Number: 5
Enter Second Number: 4
Enter '+' to add the Numbers or '*' to multiply the Numbers: +
Sum is: 9
```

```
Enter First Number: 5
Enter Second Number: 4
Enter '+' to add the Numbers or '*' to multiply the Numbers: *
Product is: 20
```

# Problem Updated: Add or Multiply 2 Numbers

Now, we have to make a decision based on the condition. Condition is whatever the user inputs i.e., either '+' or '*'.

```
Enter First Number: 5
Enter Second Number: 4
Enter '+' to add the Numbers or '*' to multiply the Numbers: +
Sum is: 9
```

```
Enter First Number: 5
Enter Second Number: 4
Enter '+' to add the Numbers or '*' to multiply the Numbers: *
Product is: 20
```

# Problem Updated: Add or Multiply 2 Numbers

When we add some kind of condition on some task, this is called conditional statement.

```
Enter First Number: 5
Enter Second Number: 4
Enter '+' to add the Numbers or '*' to multiply the Numbers: +
Sum is: 9
```

```
Enter First Number: 5
Enter Second Number: 4
Enter '+' to add the Numbers or '*' to multiply the Numbers: *
Product is: 20
```

# Problem Updated: Add or Multiply 2 Numbers

In C++, we have if statement to add conditions in our code.

# Problem Updated: Add or Multiply 2 Numbers

```
void add(int number1, int number2)
{
    int sum;
    sum = number1 + number2;
    cout << "Sum is: " << sum << endl;
}

void multiply(int number1, int number2)
{
    int product;
    product = number1 * number2;
    cout << "Product is: " << product << endl;
}
```

# Problem Updated: Add or Multiply 2 Numbers

```
#include <iostream>
using namespace std;
void add(int number1, int number2);
void multiply(int number1, int number2);

main()
{
    int number1, number2;
    char op;
    cout << "Enter First Number: ";
    cin >> number1;
    cout << "Enter Second Number: ";
    cin >> number2;
    cout << "Enter '+' to add the Numbers or '*' to multiply the Numbers: ";
    cin >> op;
    if (op == '+')
    {
        add(number1, number2);
    }
}
```

# If Statement

This is a conditional statement. IF statement has following two parts

- IF statement
- Body of IF statement

If Statement

```
if (op == '+')
{
    add(number1, number2);
}
```

Body of IF statement

# If Statement

This is a conditional statement. IF statement has following two parts

- IF statement
- Body of IF statement

Variable

Comparison Operator

Reserved Word

```
if (op == '+')
{
    add(number1, number2);
}
```

Value

# IF Statement: Boolean Expression

This is a conditional statement. IF statement has following two parts

- IF statement
- Body of IF statement

Boolean Expression

```
if (op == '+')
{
    add(number1, number2);
}
```

# IF Statement: Boolean Expression

This is a conditional statement. IF statement has following two parts

- IF statement
- Body of IF statement

Boolean Expression

```
if (op == '+')
{
    add(number1, number2);
}
```

# Problem Updated: Add or Multiply 2 Numbers

```
#include <iostream>
using namespace std;
void add(int number1, int number2);
void multiply(int number1, int number2);

main()
{
    int number1, number2;
    char op;
    cout << "Enter First Number: ";
    cin >> number1;
    cout << "Enter Second Number: ";
    cin >> number2;
    cout << "Enter '+' to add the Numbers or '*' to multiply the Numbers: ";
    cin >> op;
    if (op == '+')
    {
        add(number1, number2);
    }
}
```

This program only adds the two numbers if the user Enters '+'

# Problem Updated: Add or Multiply 2 Numbers

```
#include <iostream>
using namespace std;
void add(int number1, int number2);
void multiply(int number1, int number2);

main()
{
    int number1, number2;
    char op;
    cout << "Enter First Number: ";
    cin >> number1;
    cout << "Enter Second Number: ";
    cin >> number2;
    cout << "Enter '+' to add the Numbers or '*' to multiply the Numbers: ";
    cin >> op;
    if (op == '+')
    {
        add(number1, number2);
    }
}
```

What changes should be made to multiply the numbers if user Enters '*' ?

# Problem Updated: Add or Multiply 2 Numbers

```
#include <iostream>
using namespace std;
void add(int number1, int number2);
void multiply(int number1, int number2);
main(){
    int number1, number2;
    char op;
    cout << "Enter First Number: ";
    cin >> number1;
    cout << "Enter Second Number: ";
    cin >> number2;
    cout << "Enter '+' to add the Numbers or '*' to multiply the Numbers: ";
    cin >> op;
    if (op == '+')
    {
        add(number1, number2);
    }
    if (op == '*')
    {
        multiply(number1, number2);
    }
}
```

# A Gentle Introduction to Reusability, Decision Making and Repeatability

# Keep on Adding or Multiplying 2 Numbers

Now, this program terminates after executing one time. We want it to keep on taking the inputs and performs the calculations until closed forcefully.

```
Enter First Number: 5
Enter Second Number: 4
Enter '+' to add the Numbers or '*' to multiply the Numbers: *
Product is: 20
Enter First Number: 1
Enter Second Number: 2
Enter '+' to add the Numbers or '*' to multiply the Numbers: +
Sum is: 3
Enter First Number: 2
Enter Second Number: 3
Enter '+' to add the Numbers or '*' to multiply the Numbers: *
Product is: 6
```

# Keep on Adding or Multiplying 2 Numbers

## How can we do that?

```
Enter First Number: 5
Enter Second Number: 4
Enter '+' to add the Numbers or '*' to multiply the Numbers: *
Product is: 20
Enter First Number: 1
Enter Second Number: 2
Enter '+' to add the Numbers or '*' to multiply the Numbers: +
Sum is: 3
Enter First Number: 2
Enter Second Number: 3
Enter '+' to add the Numbers or '*' to multiply the Numbers: *
Product is: 6
```

# Keep on Adding or Multiplying 2 Numbers

In C++, if we want some code to keep on running until some condition is met, we use loops.

# Keep on Adding or Multiplying 2 Numbers

In C++, we have while loop.

Keyword  Loop Condition

```
while (condition)
{
        statements
}
```

Body of Loop

# Keep on Adding or Multiplying 2 Numbers

For now, we want the loop to run for infinite time. Therefore the condition will be true.

```
while (true)
{
        statements
}
```

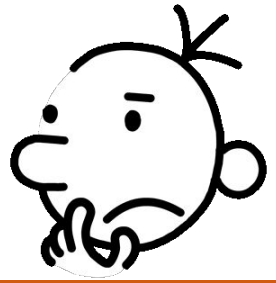
# Keep on Adding or Multiplying 2 Numbers

Now, we have to decide which statements we want to run infinitely.

```
while (true)
{
        statements
}
```

# Keep on Adding or Multiplying 2 Numbers

```
#include <iostream>
using namespace std;
void add(int number1, int number2);
void multiply(int number1, int number2);
main(){
    int number1, number2;
    char op;
    cout << "Enter First Number: ";
    cin >> number1;
    cout << "Enter Second Number: ";
    cin >> number2;
    cout << "Enter '+' to add the Numbers or '*' to multiply the Numbers: ";
    cin >> op;
    if (op == '+')
    {
        add(number1, number2);
    }
    if (op == '*')
    {
        multiply(number1, number2);
    }
}
```



# Keep on Adding or Multiplying 2 Numbers

```
#include <iostream>
using namespace std;
void add(int number1, int number2);
void multiply(int number1, int number2);
main(){
    int number1, number2;
    char op;
    cout << "Enter First Number: ";
    cin >> number1;
    cout << "Enter Second Number: ";
    cin >> number2;
    cout << "Enter '+' to add the Numbers or '*' to multiply the Numbers: ";
    cin >> op;
    if (op == '+')
    {
        add(number1, number2);
    }
    if (op == '*')
    {
        multiply(number1, number2);
    }
}
```

```
#include <iostream>
using namespace std;
void add(int number1, int number2);
void multiply(int number1, int number2);

main()
{
    int number1, number2;
    char op;
    while (true)
    {
        cout << "Enter First Number: ";
        cin >> number1;
        cout << "Enter Second Number: ";
        cin >> number2;
        cout << "Enter '+' to add the Numbers or '*' to multiply the Numbers: ";
        cin >> op;
        if (op == '+')
        {
            add(number1, number2);
        }
        if (op == '*')
        {
            multiply(number1, number2);
        }
    }
}
```

# Learning Objective

Write reusable code to solve the real life problems, repeat the code based on conditions.

# Self Assessment

## Solve the Following Programs

1. Write a program that keeps taking the temperature of a patient in Fahrenheit as input and prints "Normal" if the temperature is equal to 98.6

*Test Cases*

| Input | Output |
|---|---|
| Temperature Of Patient: 98.6 | Normal<br>Program Ends |
| Temperature Of Patient: 100 | Program Ends |

# Self Assessment

## Solve Following Programs

1. Write a Program that keeps taking the total price of the items bought by a customer until closed forcefully. If the price is exactly equal to 500$ then it gives an overall 5% discount to the customer and displays the updated price.

*Test Cases*

| Input | Output |
|---|---|
| Price: 490 | Price after Discount: 490 |
| Price: 500 | Price after Discount: 475 |
| Price: 501 | Price after Discount: 501 |