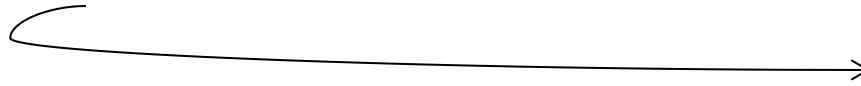


Intro - Tracing a simple program

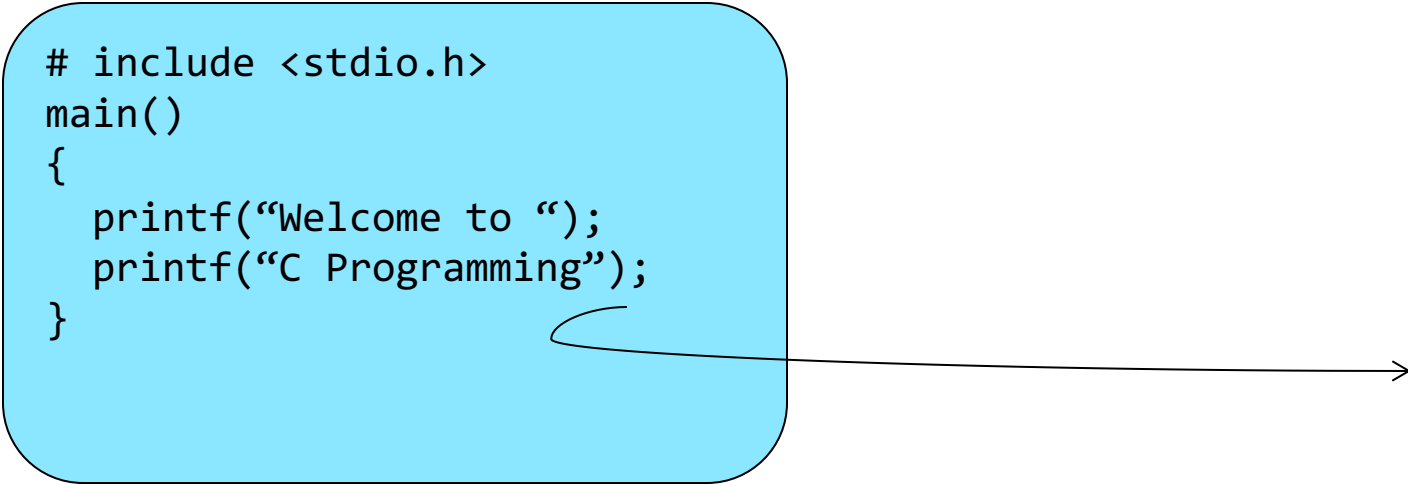
Week1

Another simple program



Another simple program

```
# include <stdio.h>
main()
{
    printf("Welcome to ");
    printf("C Programming");
}
```



sample.c

Another simple program

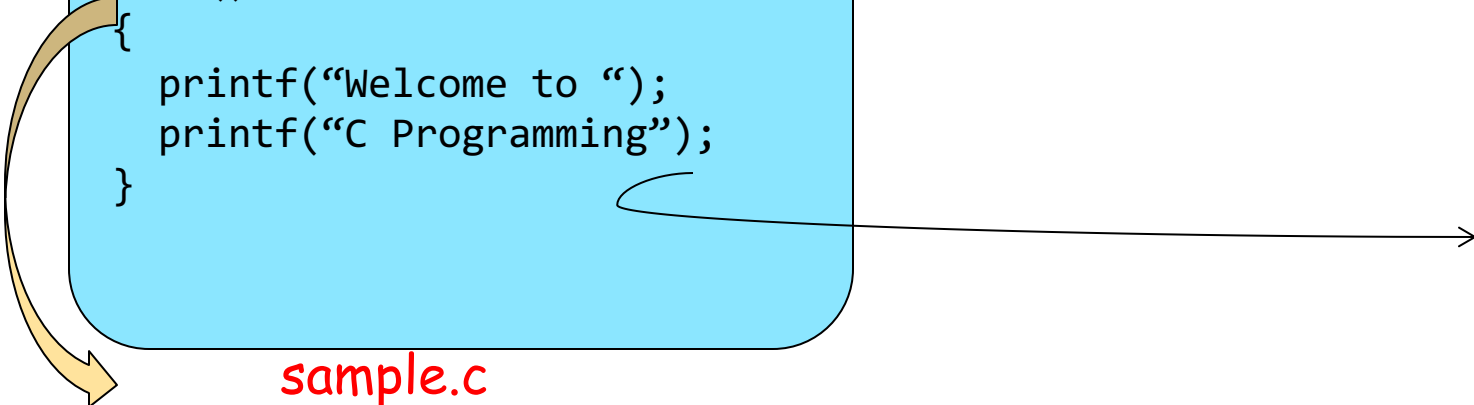
```
# include <stdio.h>
main()
{
    printf("Welcome to ");
    printf("C Programming");
}
```

Tell compiler to include the
standard input output library

sample.c

Another simple program

```
# include <stdio.h>
main()
{
    printf("Welcome to ");
    printf("C Programming");
}
```

A yellow curved arrow points from the opening curly brace of the main function in the code to the first explanatory paragraph. A black arrow points from the closing curly brace of the main function to the right, indicating the end of the program.

sample.c

Defines the main function. The brackets () show that main function takes no arguments.

Execution always begins from the first statement of main function.

First { signals the beginning of the body of main. Last } signals its end.

Another simple program

```
# include <stdio.h>
main()
{
    printf("Welcome to ");
    printf("C Programming");
}
```

sample.c

Defines the main function. The brackets () show that main function takes no arguments.

Execution always begins from the first statement of main function.

First { signals the beginning of the body of main. Last } signals its end.

There are two statements in main
→ Statement 1; Statement 2

- Each statement is terminated by semi-colon;
- Curly braces enclose a set of statements.
- Statements are executed in sequence.

Another simple program

```
# include <stdio.h>
main()
{
    printf("Welcome to ");
    printf("C Programming");
}
```

sample.c

Defines the main function. The brackets () show that main function takes no arguments.

Execution always begins from the first statement of main function.

First { signals the beginning of the body of main. Last } signals its end.

There are two statements in main
→ Statement 1; Statement 2

- Each statement is terminated by semi-colon;
- Curly braces enclose a set of statements.
- Statements are executed in sequence.

Compile and Run

```
%gcc sample.c
%./a.out
Welcome to C Programming%
```


Tracing the Execution

Tracing the Execution

```
# include <stdio.h>
main()
{
    printf("Welcome to ");
    printf("C Programming");
}
```

Tracing the Execution

```
# include <stdio.h>
main()
{
     printf("Welcome to ");
    printf("C Programming");
}
```

- Program counter  starts at the first executable statement of main.

Tracing the Execution

Line
No.

```
1 # include <stdio.h>
2 main()
3 {
4      printf("Welcome to ");
5     printf("C Programming");
6 }
```

- Program counter starts at the first executable statement of main.
- Line numbers of C program are given for clarity.

Tracing the Execution

Line
No.

```
1 # include <stdio.h>
2 main()
3 {
4      printf("Welcome to ");
5     printf("C Programming");
6 }
```

- Program counter starts at the first executable statement of main.
- Line numbers of C program are given for clarity.
- Let us run the program, one step at a time.

Tracing the Execution

Line
No.

```
1 # include <stdio.h>
2 main()
3 {
4      printf("Welcome to ");
5     printf("C Programming");
6 }
```

Output:

- Program counter starts at the first executable statement of main.
- Line numbers of C program are given for clarity.
- Let us run the program, one step at a time.
- Program terminates gracefully when main ``returns``.

Tracing the Execution

Line
No.

```
1 # include <stdio.h>
2 main()
3 {
4     printf("Welcome to ");
5     printf("C Programming");
6 }
```



Output: After lines 3,4

Welcome to

- Program counter starts at the first executable statement of main.
- Line numbers of C program are given for clarity.
- Let us run the program, one step at a time.
- Program terminates gracefully when main ``returns``.

Tracing the Execution

Line
No.

```
1 # include <stdio.h>
2 main()
3 {
4      printf("Welcome to ");
5     printf("C Programming");
6 }
```



Output:

After lines 5,6

Welcome to C Programming%

- Program counter starts at the first executable statement of main.
- Line numbers of C program are given for clarity.
- Let us run the program, one step at a time.
- Program terminates gracefully when main ``returns``.

Tracing the Execution

Line
No.

```
1 # include <stdio.h>
2 main()
3 {
4      printf("Welcome to ");
5     printf("C Programming");
6 }
```

Output:

After lines 5,6

Welcome to C Programming%

- Program counter starts at the first executable statement of main.
- Line numbers of C program are given for clarity.
- Let us run the program, one step at a time.
- Program terminates gracefully when main ``returns``.


Program Comments

Program Comments

```
# include <stdio.h>
/* a simple C program */
main()
{
    printf("Welcome to ");    /* first print */
    printf("C Programming"); /* second print */
}
```

Program Comments

```
# include <stdio.h>
/* a simple C program */
main()
{
    printf("Welcome to ");
    printf("C Programming");
}
/* first print */
/* second print */
```



- These are called **COMMENTS**.
- Any text between successive **/*** and ***/** is a comment and will be ignored by the compiler.
- Comments are **NOT** part of the program.
- They are written for us to understand or explain the program better.
- Comments can be short or long. Any number of comments may be included.
- It is a very good idea to comment your programs. For larger programs, industry, this is a must. Will help you and other developers understand and maintain programs.

Notes*

- Just as `main()` is a function, `printf("...")` is also a function. `printf` is a library function from the standard input output library, which is why we inserted the statement

```
#include <stdio.h>
```

- `printf` takes as arguments a sequence of characters in double quotes, like "Welcome to". A sequence of characters in double quotes is called a *string constant*.
- We "call" functions that we define or from the libraries.

Printing in different lines

The newline character

Printing in different lines

The newline character

■ All letters, digits, comma, underscore are called characters. There are 256 characters in C.

`'a' ... 'z' 'A' .. 'Z' '0' ... '9' '@' \. ' , ' \! ' \' ' % '
'^' '&' etc..`

Printing in different lines

The newline character

■ All letters, digits, comma, underscore are called characters. There are 256 characters in C.

`'a' ... 'z' 'A' .. 'Z' '0' ... '9' '@' \. ' , ' \! ' \' ' % '
'^' '&' etc..`

■ There is a special character called newline. In C it is denoted as `'\n'`

Printing in different lines

The newline character

■ All letters, digits, comma, underscore are called characters. There are 256 characters in C.

`'a' ... 'z' 'A' .. 'Z' '0' ... '9' '@' \. ' , ' \! ' \' ' % '
'^' '&' etc..`

■ There is a special character called newline. In C it is denoted as `'\n'`

■ When used in `printf`, it causes the current output line to end and printing will start at the next line.

The newline character

- Newline character '\n' is like any other letter and can be used multiple times in a line
- "... \nC ..." is treated as ... '\n' followed by 'C'.

```
#include <stdio.h>
main()
{
    printf("Welcome to \n");
    printf("C programming\n");
}
```

When we compile and execute,

```
$/a.out
Welcome to
C programming
$
```

Last on newlines

- To repeat, newline character '\n' is like any other character. It can be used multiple times. Another example.

```
#include <stdio.h>

main()
{
    printf("Welcome to\n\nC\n");
}
```

- When we compile and execute, we have the following.

```
$/a.out
Welcome to

C
$
```

Acknowledgments: This lecture slide is based on the material prepared by Prof. Sumit Ganguly, CSE, IIT Kanpur. The slide design is based on a template by Prof. Krithika Venkataramani.

“The instructor of this course owns the copyright of all the course materials. This lecture material was distributed only to the students attending the course ESC-101 of IIT Kanpur, and should not be distributed in print or through electronic media without the consent of the instructor. Students can make their own copies of the course materials for their use.”

