UESTC 1005 – Introductory Programming

Lecture 1 – Fundamentals of C Programs

Dr Hasan T Abbas
Hasan.Abbas@glasgow.ac.uk

Fall 2019

Glasgow College – UESTC

Machine Languages, Assembly Languages, and High-level Languages

Three types of programming languages

1. Machine languages
   - Strings of numbers giving machine specific instructions
   - Example:

     +1300042774

     +1400593419

     +1200274027
2. Assembly languages
   - English-like abbreviations representing elementary computer operations (translated via assemblers)
   - Example:

     LOAD BASEPAY

     ADD OVERPAY

     STORE GROSSPAY
3. High-level languages
   - Codes similar to everyday English
   - Use mathematical notations (translated via compilers)
   - Example:

```
grossPay = basePay + overTimePay
```

Fundamentals of C – A High level Language

```
Directives

int main(){

   statements …
}
```

Structure of a C Program

Directives – some commands that modify the program prior to compiling

Functions – block of code that is executable

Statements – commands to be performed when the program is run

Fundamentals of C

```c
/*  UESTC 1005
    A first program in C */
 #include <stdio.h>

 /* function main begins program execution */
 int main()
 {
    printf( "Welcome to C!\n" );

    return 0; /* indicate that program ended successfully */

 } /* end function main */
```

Comments

- Text surrounded by /* and */ is ignored by computer
- Used to describe program

- `#include <stdio.h>`
  - Preprocessor directive
    - Tells computer to load contents of a certain file
  - `<stdio.h>` allows standard input/output operations

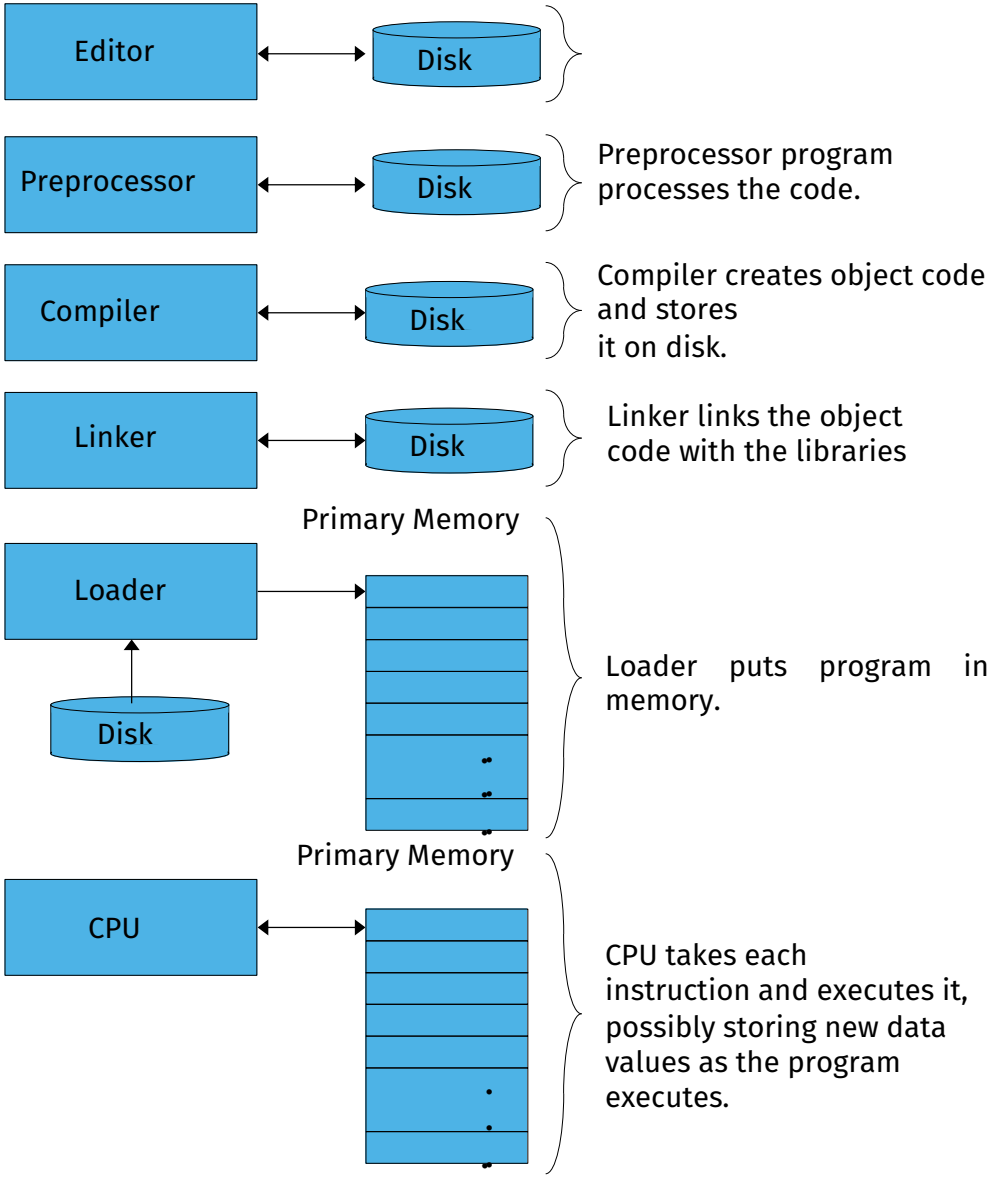
Fundamentals of C

```c
/*  UESTC 1005
    A first program in C */
 #include <stdio.h>

 /* function main begins program execution */
 int main()
 {
    printf( "Welcome to C!\n" );

    return 0; /* indicate that program ended successfully */

 } /* end function main */
```

- `int main()`
  - C programs contain one or more functions, exactly one of which must be `main`
  - Parenthesis used to indicate a function
  - `int` means that `main` "returns" an integer value
  - Braces (`{` and `}`) indicate a block
    - The bodies of all functions must be contained in braces

C Program Execution Flow

1. Edit
2. Preprocess
3. Compile
4. Link
5. Load
6. Execute



Useful Statements

Escape Sequence	Description
`\n`	Newline. Position the cursor at the beginning of the next line.
`\t`	Horizontal tab. Move the cursor to the next tab stop.
`\a`	Alert. Sound the system bell.
`\\`	Backslash. Insert a backslash character in a string.
`\"`	Double quote. Insert a double quote character in a string.

Example C Programs

```
/* Hello
   Printing on one line with two printf statements
*/
#include <stdio.h>

/* function main begins program execution */
int main()
{
   printf( "Welcome " );
   printf( "to C!\n" );

   return 0; /* indicate that program ended successfully */

} /* end function main */
```

```
Welcome to C!
```

Output

Example C Programs

```
/* Hello
   Printing on one line with two printf statements
*/
#include <stdio.h>

/* function main begins program execution */
int main()
{
   printf( "Welcome \n " );
   printf( "to C!\n" );

   return 0; /* indicate that program ended successfully */

} /* end function main */
```

```
Welcome
to C!
```

Output

Fundamentals of C Program – Functions

- Functions are building blocks from which programs are constructed.
- A series of statements that have been grouped together and given a name.
- Some functions compute a value; some *don't*.
- There are *library functions* and *user functions*.

$$f(x) = x + 1$$
$$g(y, z) = y^2 - z^3$$

- C equivalent:
  - `return x + 1;`
  - `return y*y - z*z*z;`

Fundamentals of C Program – Functions

- C program may have many functions.
- `main()` function is necessary.
- It can only be named `main()` not `MAIN()`.
- `main()` function tells the operating system that the program has been terminated.

```c
int main()
{
return 0;
}
```

```c
int main(void)
{
return 0;
}
```

```c
void main()
{
return;
}
```

```c
void main(void)
{
return;
}
```

- Contents of the functions are inside braces { }

Fundamentals of C Program – Statements

- A statement is a command to be executed by the program.

```
1    /*  UESTC 1005
2        A first program in C */
3     #include <stdio.h>
4
5     /* function main begins program execution */
6     int main()
7     {
8        printf( "Welcome to C!\n" );
9
10       return 0; /* indicate that program ended
successfully */
11
12    } /* end function main */
```

- Each Statement ends with a semicolon ;

Fundamentals of C Program – Directives

- Directive is a command intended for the *preprocessor.*

```
1    /*  UESTC 1005
2        A first program in C */
3    #include <stdio.h>
4
5    /* function main begins program execution */
6    int main()
7    {
8        printf( "Welcome to C!\n" );
9
10       return 0; /* indicate that program ended
successfully */
11
12   } /* end function main */
```

- Directive starts with the character #

Fundamentals of C Program – Comments

- Comments are useful information put into the program for documentation,

```
/*  UESTC 1005
    A first program in C */
 #include <stdio.h>

 /* function main begins program execution */
 int main()
 {
    printf( "Welcome to C!\n" );

    return 0; /* indicate that program ended successfully */

 } /* end function main */
```

- Comments start with the characters /* and end with */

Fundamentals of C Program – Comments

- Comment Styles

```
/* Name: welcome.c                              */
/* Purpose: prints a welcome message on screen */
/* Author: X Y Z                                */


/* Name: welcome.c
   Purpose: prints a welcome message on screen
   Author: X Y Z                                */

/***********************************************
*   Name: welcome.c                            *
*   Purpose: prints a welcome message on screen *
*   Author: X Y Z                              *
***********************************************/
```

- We can also use // for single line comments

```
// Name: welcome.c
// Purpose: prints a welcome message on screen
// Author: X Y Z
```

Next Lecture – Friday

- Input Functions
  - `scanf()`

- Declarations
- Variables and Assignments
  - Declarations
  - Assignments
  - Variable Types