EEE: 103

Computer Programming

L3: Basics of C programming

Prepared by: Sk Tahmed Salim Rafid

First C Program

```c
#include <stdio.h>
/* This is my first program.
This section is multi line comments*/
void main(){
 // Printing a welcome message
 printf("Welcome to EEE-103");
}
```

Lines 0 | Lines 1-2 | Lines 3-6 | Lines 4 | Lines 5

Output:

```
Welcome to EEE-103
```

Annotations:

Line 1:

Preprocessor

Line 4:

void = Return Type, main = Function

Line 4:

Scope: { to }

Line 6:

printf = Function

Understanding C Program Components

Preprocessor

Begins with #. Header files end with .h. The header stdio.h is used for input/output operations as how the input/output works. Relevant functions are defined in stdio.h

Scope

Scope is defined from one opening brace { to its closing brace }. Everything inside these braces belongs to that scope.

Main Function

In C when we run a code, it looks for main function and the lines in main function are executed sequentially.

Comments

Comments are lines that are not executed. Comments are for programmers to understand what the code means or what a line means.

Variables - What are Variables?

- Variables are used to store data in memory
- We can store different types of data like int, float, char etc.
- Variables can be declared and used to store data and read data
- Variables must be declared before use
- Example: Think of variables as labeled boxes where you store values

How to Declare Variables?

int
Whole number

```
int x = 103;
```

float
Floating point / fraction

```
float y = 10.5;
```

double
Float but more precision

```
double z = 3.14159;
```

char
Single character

```
char c = 'a';
```

Let's assume we will store 103. It's a whole number. So we need one storage that can store int.

```
int x = 103;
```

How to Declare Variables? (Contd.)

✓ `int x = 10;` — *Valid - integer value*

✗ `int x = 10.3;` — *Invalid - float can't be stored in int due to bit limitation*

✓ `float x = 10.3;` — *Valid - float value*

✓ `float x = 10;` — *Valid - 10 will be converted to 10.0*

✓ `char x = 'a';` — *Valid - single character*

Variable Naming Rules

Rules:

- Case sensitive (myVar and myvar are different)
- Can have letters, numbers and underscore (_)
- No comma/space allowed
- First character can't be a number, but underscore (_) is valid
- Variable names should be meaningful

Naming Conventions:

Snake case: my_name, student_id

Camel case: myName, studentId

Variable Naming Rules (Contd.)

✓ Valid Examples

```
int courseCode;
```

```
float _temperature;
```

```
int student1;
```

```
char my_grade;
```

```
int totalMarks;
```

✗ Invalid Examples

```
int 1student;
```
✗ *Starts with number*

```
float my marks;
```
✗ *Contains space*

```
int student,id;
```
✗ *Contains comma*

```
char for;
```
✗ *Reserved keyword*

```
int @price;
```
✗ *Special character not allowed*

Reserved Keywords in C

These keywords are reserved and CANNOT be used as variable names!

`auto`	`break`	`case`	`char`	`const`
`continue`	`default`	`do`	`double`	`else`
`enum`	`extern`	`float`	`for`	`goto`
`if`	`int`	`long`	`register`	`return`
`short`	`signed`	`sizeof`	`static`	`struct`
`switch`	`typedef`	`union`	`unsigned`	`void`
`volatile`	`while`			

Printing Variable

Using %d to print integer variable

```c
#include <stdio.h>
/* This is my 2nd program.
Here I will store the course code in a variable
Finally I will print them*/
void main(){
 int courseCode = 103;
 printf("Welcome to EEE-%d",courseCode);
}
```

Lines 0 | Lines 1-3 | Lines 4-7 | Lines 5 | Lines 6

Output:

```
Welcome to EEE-103
```

Printing Multiple Variables

Using multiple format specifiers

```c
#include <stdio.h>
/* This is my 3rd program.
Here I will store the course code and section variables
Finally I will print them*/
void main(){
 int courseCode = 103;
 int sec = 2;
 printf("Welcome to EEE-%d Section-%d",courseCode,sec);
}
```

Lines 0 | Lines 1-3 | Lines 4-8 | Lines 5 | Lines 6 | Lines 7

Output:

```
Welcome to EEE-103 Section-2
```

Modifiers & Format Specifiers

What are Modifiers?

In case we need more space for data storage or we know that data will be unsigned, we can use modifiers:

```
long int x = 1000000;
```

Use of long makes range from 4 byte to 8 byte

```
unsigned int y = 5000;
```

Gives bigger range but no negative numbers

Format Specifiers for Printing/Input

`%d`	int	*For integer values*
`%f`	float	*For floating point values*
`%.2f`		*After decimal 2 digits will be shown*
`%c`	char	*For character values*

Taking Input - scanf()

- scanf() is used to take input from the user
- Syntax: scanf("specifier", &variableName);
- The & (ampersand) is MANDATORY
- When we write variableName, we get the value stored in that variable
- But we want to access the location and store data in that variable
- To access the location, we use & operator

Example:

```
scanf("%d", &courseCode);
```

& gives the memory address of the variable

Taking Input and Printing

Reading course code from user

```c
#include <stdio.h>

void main(){
 int courseCode;
 printf("Enter the course code:");
 scanf("%d", &courseCode);
 printf("Welcome to EEE-%d",courseCode);
}
```

Lines 0 | Lines 1-7 | Lines 3 | Lines 4 | Lines 5 | Lines 6

Output:

```
Enter the course code: 103
Welcome to EEE-103
```

Operations in C

Arithmetic Operators

+	Addition
-	Subtraction
*	Multiplication
/	Division
%	Modulus (Remainder)
++	Increment by 1
--	Decrement by 1

Bitwise Operators

Example: a = 0b1010 (10), b = 0b1011 (11)

& AND

```
a & b = 0b1010 (10)
```

| OR

```
a | b = 0b1011 (11)
```

^ XOR

```
a ^ b = 0b0001 (1)
```

~ 1's complement

```
~a = 0b0101 (inverted)
```

<< Left shift

```
a << 1 = 0b10100 (20)
```

>> Right shift

```
a >> 1 = 0b0101 (5)
```

Assignment Operators

Basic Operations

```
a = 10;
```

Assign a's value as 10

```
a += 20;
```

This means a = a + 20
// a already has 10, so a becomes 30

```
a++;
```

This means a = a + 1
// so a becomes 31

Pre vs Post Increment

```
printf("Value of a is %d", a++);
```

Output: 31
// But due to ++, value of a is updated to 32
// Print shows 31 (post-increment)

```
a = 20;
printf("Value of a is %d", ++a);
```

Output: 21
// ++a means increment first then work
// (pre-increment)

Example: Area of Rectangle

Calculate area using length and width

```c
#include <stdio.h>
void main() {
 float length, width, area;
 // Input length and width from user
 printf("Enter the length of the rectangle: ");
 scanf("%f", &length);
 printf("Enter the width of the rectangle: ");
 scanf("%f", &width);
 area = length * width;
 printf("The area of the rectangle is: %.2f\n", area);
}
```

Lines 0 | Lines 1-10 | Lines 2 | Lines 3-5 | Lines 6-7 | Lines 8 | Lines 9

Output:

```
Enter the length: 5.5
Enter the width: 3.2
The area is: 17.60
```

Practice Problems

1. Question 1: Write a program to convert feet to centimeters (1 foot = 30.48 cm)

2. Question 2: Write a program to calculate the sum of two numbers entered by user

3. Question 3: Write a program to find the average of three numbers

4. Question 4: Write a program to convert temperature from Celsius to Fahrenheit

5. Question 5: Write a program to calculate the area of a circle (Area = $\pi \times r^2$)

Thank You

Keep Coding!