Programing Methodology in C

Lecture 6 – Decision Making

By Elubu Joseph

josebulinga@gmail.com

Agenda

1. Decision Making
    i. if statements
    ii. Ternary operator (?:)
    iii. switch Statement and
    iv. Loop functions

Decision Making In C Programing Language

Decision Making in C

Decision making is about deciding the order of execution of statements based on certain conditions or repeat a group of statements until certain specified conditions are met. We use the following statements to instruct our programs to make decision,

1. if statement
2. switch statement
3. Ternary (? :)
4. loops
5. goto statement, Studytonight.com. (n.d.).

Decision making with if statement

The if statement may be implemented in different forms depending on the complexity of conditions to be tested. The different forms are,

1.Simple if statement

2.if....else statement

3.Nested if....else statement

4.Using else if statement

Simple if statement

The general form of a simple if statement is,

```
if(expression) {
    statement inside;
}
statement outside;
```

If the *expression* returns true, then the **statement-inside** will be executed, otherwise **statement-inside** is skipped and only the **statement-outside** is executed.

Example:

```
#include<stdio.h>
void main( ) {
int x, y;
x = 50;
y = 13;
if (x > y ) {
printf("x is greater than y");
}
printf(“y is greater than x“, );
}
```

if...else statement

The general form of a simple if...else statement is,

```
if(expression) {
statement block1;
}
else {
statement block2;
}
```

If the *expression* is true, the **statement-block1** is executed, else **statement-block1** is skipped and **statement-block2** is executed.

Output: y is greater than x

Example: if_else Statement

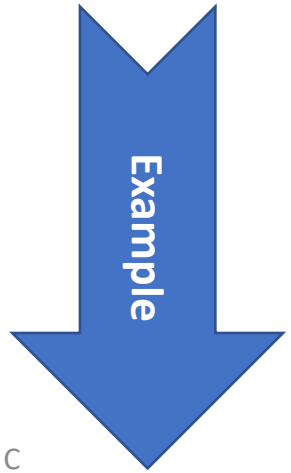
```
#include <stdio.h>
void main( ) {
int x, y; x = 30; y = 40;
if (x > y ) {
printf("x is greater than y");
}
else {
printf("y is greater than x");
}
 }
```

Nested if....else statement

The general form of a nested if...else statement is,

```
if( expression ) {
       if( expression1 ) {
       statement block1;
}
else { statement block2; }
}
else { statement block3; }
```

if ***expression*** is false then **statement-block3** will be executed, otherwise the execution continues and enters inside the first if to perform the check for the next if block, where if *expression 1* is true the **statement-block1** is executed otherwise **statement-block2** is executed.



Example: Nested if .. Else statement

```
#include <stdio.h>
void main( ) {
int a, b, c;
printf("Enter 3 highest marks you got...");
scanf("%d%d%d",&a, &b, &c);
if(a > b) {
        if(a > c) {
        printf("a is the greatest");
        }
        else {
        printf("c is the greatest");
        }
}
else {
        if(b > c) {
        printf("b is the greatest");
        } else {
        printf("c is the greatest");
        }
}
}
```

else if ladder

The general form of else-if ladder is,

```
if(expression1) {
        statement block1;
}
else if(expression2) {
        statement block2;
}
else if(expression3 ) {
        statement block3;
}
else {
        default statement;
}
```

The expression is tested from the top(of the ladder) downwards. As soon as a **true** condition is found, the statement associated with it is executed.

Example: else…..if Ladder

```c
#include <stdio.h>
void main( ) {
int a;
printf("Enter a number...");
scanf("%d", &a);
if(a%5 == 0 && a%8 == 0) {
printf("Divisible by both 5 and 8");
}
else if(a%8 == 0) {
printf("Divisible by 8");
}
else if(a%5 == 0) {
printf("Divisible by 5");
}
else {
printf("Divisible by none");
}
}
```

Note the following

1. In `if` statement, a single statement can be included without enclosing it into curly braces `{ … }`

```
int a = 50;
if(a > 49)
printf("success");
```

No curly braces are required in the above case, but if we have more than one statement inside if condition, then we must enclose them inside curly braces.

2. `==` must be used for comparison in the expression of `if` condition, if you use `=` the expression will always return **true**, because it performs assignment not comparison.

3. Other than **0(zero)**, all other values are considered as **true**.

```
if(27)
printf("Very Mature");
```

In above example, **Very mature** will be printed.

Making decision using Ternary (?:) operator

Ternary Operator ? : Operator

It is actually the if condition that we use in C language decision making, but using conditional operator, we turn the if condition statement into a short and simple operator.

The syntax of a conditional operator is :

```
Expression 1 ? expression 2: expression 3
```

Explanation:

1. The question mark **"?"** in the syntax represents the **if** part.
2. The first expression (expression 1) generally returns either true or false, based on which it is decided whether (expression 2) will be executed or (expression 3)
3. If (expression 1) returns true then the expression on the left side of **" : "** i.e (expression 2) is executed.
4. If (expression 1) returns false then the expression on the right side of **" : "** i.e (expression 3) is executed.

Example: Ternary ?: based decision making

```
#include<stdio.h>
int main(){
    int k=45, l=50;
    l>k? printf("\nYeah %d>%d\n",l,k):printf("No
%d<%d\n",l,k);
return 0;
}
```

Switch Statement

Switch statement in C

When you want to solve multiple option type problems, for example: Menu like program, where one value is associated with each option and you need to choose only one at a time, then, switch statement is used.

Switch statement is a control statement that allows us to choose only one choice among the many given choices. The expression in switch evaluates to return an integral value, which is then compared to the values present in different cases. It executes that block of code which matches the case value. If there is no match, then **default** block is executed(if present).

The general form of switch statement is,

```
switch(expression) {
     case value-1: block-1;
     break;
     case value-2: block-2;
     break;
     case value-3: block-3;
     break;
     case value-4: block-4;
     break;
     default: default-block;
     break;
}
```

Rules for using switch statement

1. The expression (after switch keyword) must yield an **integer** value i.e the expression should be an integer or a variable or an expression that evaluates to an integer.
2. The case **label** values must be unique.
3. The case label must end with a colon(:)
4. The next line, after the **case** statement, can be any valid C statement.

Points to Remember

1. We don't use those expressions to evaluate switch case, which may return floating point values or strings or characters.

2. break statements are used to **exit** the switch block. It isn't necessary to use break after each block, but if you do not use it, then all the consecutive blocks of code will get executed after the matching block.

Switch Practical example without **beaks**

```
int i = 1;
switch(i) {

     case 1: printf("A");
     // Note No break
     case 2: printf("B");
     // No break
     case 3: printf("C");
     break; // End of statements
}
```

Without the keyword “break”, the output will be OUTPUT : ABC

Key Notes on the above program

1.The output was supposed to be only **A** because only the first case matches, but as there is no break statement after that block, the next blocks are executed too, until it a break statement is encountered or the execution reaches the end of the switch block.

2.default case is executed when none of the mentioned case matches the switch expression. The default case can be placed anywhere in the switch case. Even if we don't include the default case, switch statement works.

Key Notes on the above program+

3.Nesting of switch statements are allowed, which means you can have switch statements inside another switch block.

Note. Nested switch statements should be avoided as it makes the program more complex and less readable.

Switch Practical user User Input with **beaks**

```
#include<stdio.h>
int main(){

int ID;
printf("What is your ID No?: ");
scanf("%d", &ID);

switch(ID){
case 100: printf("Ongyra, welcome.");
break;
```

Switch Practical user User Input with **beaks+**

```
case 200: printf("Thats great Joe.");
break;

case 300: printf("Not assigned");
break;

case 400: printf("Egimu's ID");
break;
case 500: printf("Okiria's ID");
break;
default : printf("Unknown Application");

}
getch(); return(0);
}
```

What will be the output if the user entered 1000?

Unknown Application

Example of switch statement

```
#include<stdio.h>
void main( ) {
        int a, b, c, choice;
        while(choice != 3) {
        /* Printing the available options */
        printf("\n 1. Press 1 for addition");
        printf("\n 2. Press 2 for subtraction");
        printf("\n Enter your choice");
        /* Taking users input */
        scanf("%d", &choice);
        switch(choice) {
                case 1: printf("Enter 2 numbers");
                scanf("%d%d", &a, &b);
                c = a + b; printf("%d", c);
                break; case 2: printf("Enter 2 numbers");
                scanf("%d%d", &a, &b);
                c = a - b; printf("%d", c);
                break;
                default: printf("you have passed a wrong key");
                printf("\n press any key to continue");
        }
        }
}
```

Difference between switch and if

1. if statements can evaluate float conditions. switch statements cannot evaluate float conditions.

2. if statement can evaluate relational operators. switch statement cannot evaluate relational operators i.e they are not allowed in switch statement.

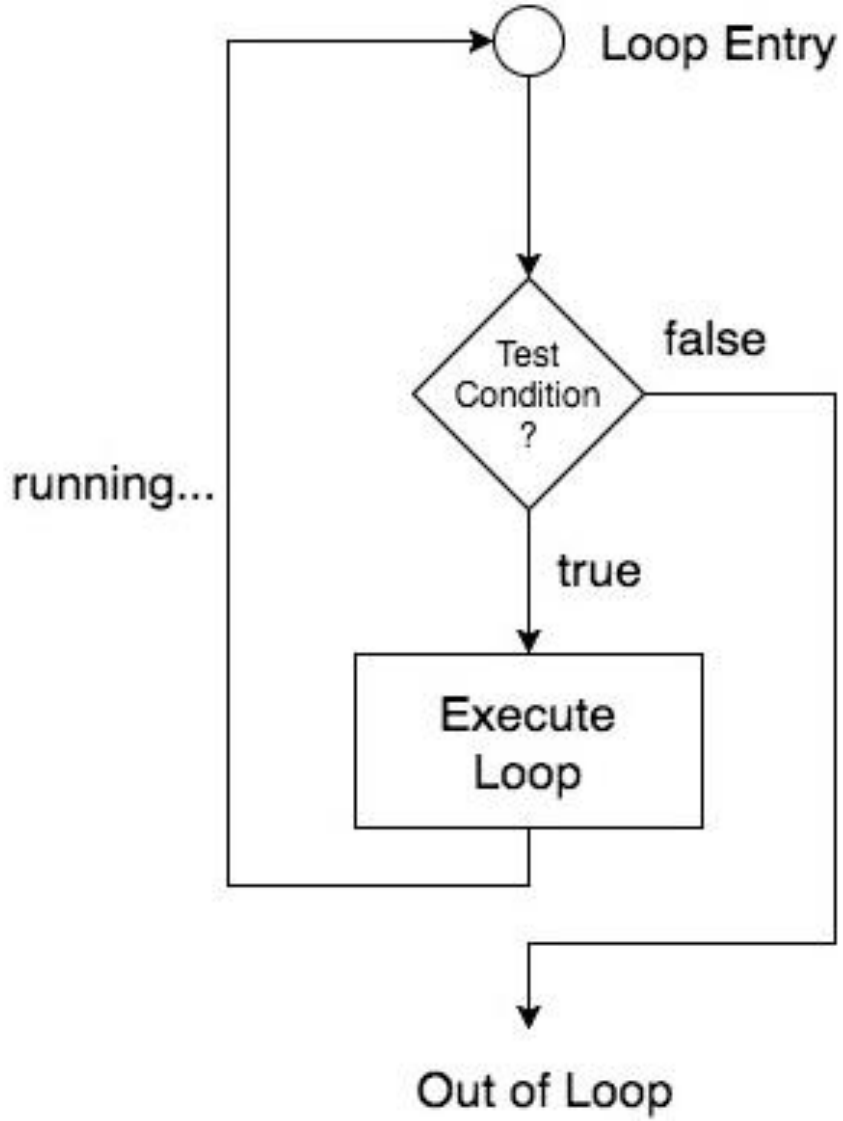
Dealing with Loops in C

How to use Loops in C

In any programming language including C, loops are used to execute a set of statements repeatedly until a particular condition is satisfied.

How it Works

The below diagram depicts a loop execution,



How Loops work explained

As per the above diagram, if the Test Condition is true, then the loop is executed, and if it is false then the execution breaks out of the loop. After the loop is successfully executed the execution again starts from the Loop entry and again checks for the Test condition, and this keeps on repeating.

The sequence of statements to be executed is kept inside the curly braces `{}` known as the **Loop body**. After every execution of the loop body, **condition** is verified, and if it is found to be **true** the loop body is executed again. When the condition check returns **false**, the loop body is not executed, and execution breaks out of the loop.

Types of Loop

There are 3 types of Loop in C language, namely:

1. while loop
2. for loop
3. do while loop

Basic Structure of while loop

can be addressed as an **entry control** loop.

It is completed in 4 steps.

1. Variable initialization.(e.g int x = 0;)
2. condition(e.g while(x <= 10))
3. Statement to be executed
4. Variable increment or decrement

   (x++ or x-- or x = x + 2)

Syntax :

```
int x=4;
while(x<=10) {
printf("I love it\n");
x++;
}
```

while loop

Example: Program to print first 9 natural numbers

```
#include<stdio.h>
void main( ) {
     int x; x = 0;
     while(x < 10) {
          printf("%d\t", x);
/* below statement means, do x = x+1,
increment x by 1*/
          x++;
     }
}
```

Output

0 2 3 4 5 6 7 8 9

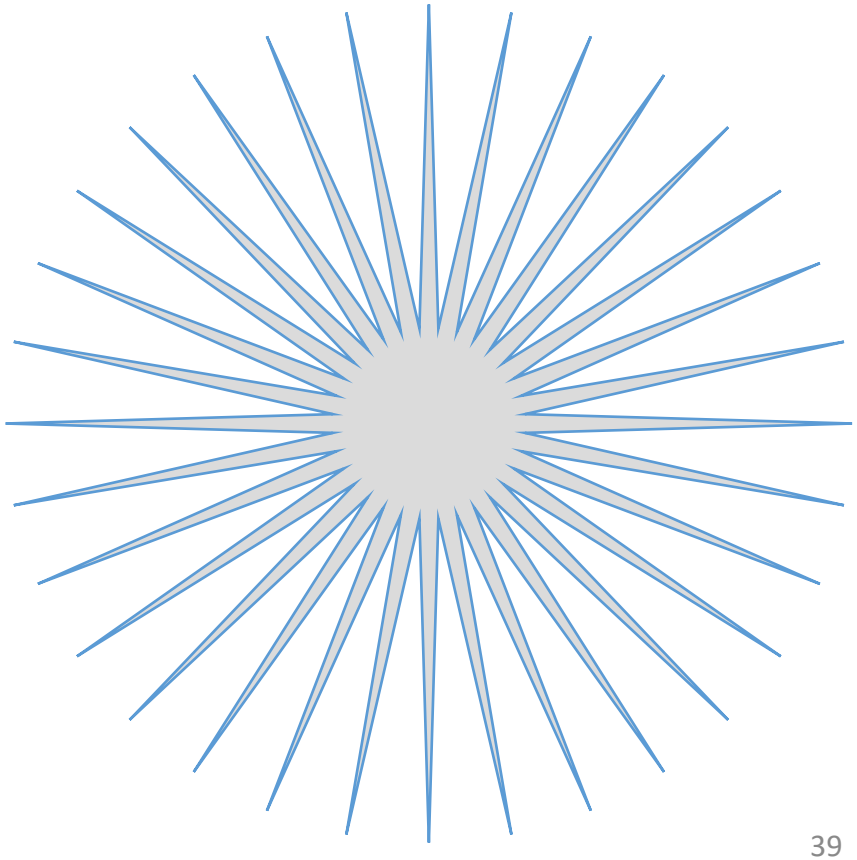
We will look at other loop functions in our next lecture

1. for Loop
2. and do while loop

Summary

1. Decision Making
    i. if statements
    ii. Ternary operator(?:)
    iii. switch Statement and
    iv. Loop functions –introduced three loop functions dealt with while loop(layout ant sample program)

Thank you for you attention



References

Decision making in C. Studytonight.com. (n.d.). Retrieved October 9, 2021, from https://www.studytonight.com/c/decision-making-in-c.php.