Unit III – Decision Making & Branching

PROF. MR. MAHIPAL KHOJA

Artificial Intelligence & Data Science

Decision Making

- In C, programs can choose which part of the code to execute based on some condition. This ability is called **decision making** and the statements used for it are called **conditional statements.**

- These statements evaluate one or more conditions and make the decision whether to execute a block of code or not.

Conditional Statements in C

1. if-else
2. Switch
3. Conditional Operator
4. Jump Statements

if-else

- **if** - The if statement is the simplest decision-making statement. It is used to decide whether a certain statement or block of statements will be executed or not i.e. if a certain condition is true then a block of statements is executed otherwise not.

A **condition** is any expression that evaluates to either a true or false or values convertible to true or false.

```
Statement – if(condition){
                    code;
             }
```

if-else

- **if-else** - We can use the **else** statement with the **if** statement to execute a block of code when the condition is false. The if-else statement consists of two blocks, one for false expression and one for true expression.

```
Statement – if(condition){
                    code;
            }
            else{
                    code;
            }
```

if-else

- **Nested if-else** - A nested if in C is an if statement that is the target of another if statement. Nested if statements mean an if statement inside another if statement.

- C allow us to nested if statements within if statements, i.e, we can place an if statement inside another if statement.

if-else

Nested if-else

Statement –
```
if(condition1){
          if(condition2){
                    code;
          }
          else{
                    code;
          }
}
else{
          code;
}
```

if-else

- **if-else-if Ladder** - The if else if statements are used when the user has to decide among multiple options. The C if statements are executed from the top down. As soon as one of the conditions controlling the if is true, the statement associated with that if is executed, and the rest of the C else-if ladder is bypassed. If none of the conditions is true, then the final else statement will be executed. if-else-if ladder is similar to the switch statement.

if-else

if-else-if Ladder

```
Statement – if(condition1){
                        code;
                }
                else if(condition2){
                        code;
                }
                else if(condition3){
                        code;
                }
                else{
                        code;
                }
```

switch Statement

switch Statement

The switch case statement is an alternative to the if else if ladder that can be used to execute the conditional code based on the value of the variable specified in the switch statement.

The switch block consists of cases to be executed based on the value of the switch variable.

switch Statement

Statement –

```
int x ;
switch(x){
          case 1:
                    code;
                    break;
          case 2:
                    code;
                    break;
          case 3:
                    code;
                    break;
          default:
                    code;
                    break;
}
```

Conditional Operator

Conditional Operator

The conditional operator is used to add conditional code in our program. It is similar to the if-else statement.

It is also known as the ternary operator as it works on three operands.

Statement – condition ? True : false

Example – int f = 0 ;

int a = (f == 0 ? 10 : 20);

then a = 10 .

Jump Statements

Jump Statements

These statements are used in C for the unconditional flow of control throughout the functions in a program.

C support four types of jump statements:

1. Break.
2. Continue.
3. Goto.
4. Return.

Jump Statements

- **break -** This loop control statement is used to terminate the loop. As soon as the break statement is encountered from within a loop, the loop iterations stop there, and control returns from the loop immediately to the first statement after the loop.

Statement –

```
for(){
        if(condition){
                break;
        }
}
```

Jump Statements

- **continue -** This loop control statement is just like the break statement. The continue statement is opposite to that of the break statement, instead of terminating the loop, it forces to execute the next iteration of the loop.

- As the name suggests the continue statement forces the loop to continue or execute the next iteration. When the continue statement is executed in the loop, the code inside the loop following the continue statement will be skipped and the next iteration of the loop will begin.

Jump Statements

Statement –

```
for(){
        if(condition){
                continue;
        }
        else{
                code;
        }
}
```

Jump Statements

- **goto** - The goto statement in C also referred to as the unconditional jump statement can be used to jump from one point to another within a function.

Statement –

```
label:
        code;
if(condition){
        goto label;
}
```

Jump Statements

- **return** - The return in C returns the flow of the execution to the function from where it is called. This statement does not mandatorily need any conditional statements. As soon as the statement is executed, the flow of the program stops immediately and returns the control from where it was called.
- The return statement may or may not return anything for a void function, but for a non-void function, a return value must be returned.

Statement –

```
int sum(int a , int b){
        int s = a + b ;
        return s;
}
```

Q n A

What will be the output of the following code:

```
#include<stdio.h>
int main(){
        int a = 5 ;
        if(a<=5){
                if(a>2){
                        printf(“High”);
                }
                else{
                        printf(“Low”);
                }
        }
        else{
                printf(“High High”);
        }
return 0;
}
```

Q n A

What will be the output of the following code:

```
#include<stdio.h>
int main(){
        int a = 12 ;
        switch(a){
                case 10:
                        printf(“10”);
                        break;
                default:
                        printf(“12”);
                        break;
        }
        return 0;
}
```

LOOPS IN C

- **Loops** in C programming are used to repeat a block of code until the specified condition is met.
- It allows programmers to execute a statement or group of statements multiple times without writing the code again and again.

- There are 3 looping statements in C
  1. For loop.
  2. While loop.
  3. Do-while loop.

LOOPS IN C

The various parts of the loops are:

- **Initialization:** Initialize the variable to some initial value.
- **Test Condition:** This specifies the test condition. If the condition evaluates to true, then body of the loop is executed. If evaluated false, loop is terminated.
- **Update Expression:** After the execution loop's body, this expression increments/decrements the loop variable by some value.
- **Body of Loop**: Statements to repeat. Generally enclosed inside {} braces.

for LOOP IN C

for loop is an entry-controlled loop, which means that the condition is checked before the loop's body executes.

Statement –

```
for( initialization ; condition ; updation ){
          body of loop;
}
```

while LOOP IN C

A while loop is also an entry-controlled loop in which the condition is checked before entering the body.

Statement –

```
 initialization ;
while( condition ){
            body of loop ;
             updation ;
}
```

do-while LOOP IN C

The do-while loop is an exit-controlled loop, which means that the condition is checked after executing the loop body.

Due to this, the loop body will execute at least once irrespective of the test condition.

Statement –

```
 initialization ;
do{
           body of loop ;
            updation ;
} while( condition );
```

Infinite Loop IN C

An **infinite loop** is executed when the test expression never becomes false, and the body of the loop is executed repeatedly. A program is stuck in an Infinite loop when the condition is always true. Mostly this is an error that can be resolved by using Loop Control statements.

Statement –

```
for( ; ; ){
        printf(“Infinite loop”);
}
```

Infinite Loop IN C

Statement –

```
while( 1 ){
        printf(“Infinite loop”);
}
```

Statement –

```
do{
        printf(“Infinite loop”);
}while(1);
```

Parul® University

DIGITAL LEARNING CONTENT

Nested Loops IN C

- Nesting loops means placing one loop inside another. The inner loop runs fully for each iteration of the outer loop.
- This technique is helpful when you need to perform multiple iterations within each cycle of a larger loop, like when working with a two-dimensional array or performing tasks that require multiple levels of iteration.

Statement –

```
for( initialization 1 ; condition 1 ; updation 1 ){
          for( initialization 2 ; condition 2 ; updation 2 ){
                    body of loop;
          }
}
```

Loop Control Statements

- Loop control statements in C programming are used to change execution from its normal sequence.

Name	Description
break	The break statement is used to terminate the loop statement.
continue	When encountered, the continue statement skips the remaining body and jumps to the next iteration of the loop.
goto	goto statement transfers the control to the labeled statement.

Q n A

What will be the output of the following code:

```
#include<stdio.h>
int main(){
        int a = 1 ;
        for(a ; a < 10 ; a++){
                printf(“%d”,a);
        }
        return 0;
}
```

Q n A

What will be the output of the following code:

```
#include<stdio.h>
int main(){
        int a = 1 ;
        while(a<12){
                printf("%d",a);
                a=a+10;
        }
        return 0;
}
```

Q n A

What will be the output of the following code:

```
#include<stdio.h>
int main(){
        int a = 1 ;
        while(a<12){
                printf("%d",a);
                a++;
                if(a>5){
                        break;
                }
        }
        return 0;
}
```

DIGITAL LEARNING CONTENT

Parul® University

www.paruluniversity.ac.in