CONTROL FLOW STATEMENTS

IN C PROGRAMMING LANGUAGE

CREATED BY
TARUN SHARMA
(LECTURER COMPUTER SCIENCE)

WHAT IS CONTROL FLOW STATEMENTS

- A control flow statements is a primary concept in most high-level programming languages. It is a block of code. More specifically, control flow statements are blocks of code that control the flow of a program. In other words, a control structure is a container for a series of function calls, instructions and statements.

INFORMATIQUE TUTORIAL

TYPES OF CONTROL FLOW STATEMENTS

- There are three types of control flow statements:
  - Branching / Decision Making Statements
  - Iterative / Looping Statements
  - Jumping Statements

INFORMATIQUE TUTORIAL

CONTROL FLOW STATEMENTS

- Branching / Decision Making Statements
  - If Statements
    - Simple If
    - If else
    - Nested if
    - Else if ladder
  - Switch Case Statement
  - Conditional Operator
- Looping / Iterative Statements
  - Entry Control Loops
    - While Loop
    - For Loop
  - Exit Control Loop
    - Do While Loop
- Jumping Statements
  - Break Statement
  - Continue Statement
  - Goto Statement

BRANCHING STATEMENTS

- C program executes program sequentially. Sometimes, a program requires checking of certain conditions in program execution. C provides various key condition statements to check condition and execute statements according conditional criteria. These statements are called as 'Decision Making Statements' or 'Conditional Statements.'

TYPES OF CONDITIONAL STATEMENTS

- If Statement
  - Simple If
  - If Else Statement
  - Nested If Statement
  - Else If Ladder
- Switch Case Statements
- Conditional Operator

IF STATEMENTS – SIMPLE IF STATEMENT

- Takes an expression in parenthesis and a statement or block of statements. If the expression is true then the statement or block of statements gets executed otherwise these statements are skipped.
- Syntax:

```
if(Condition)
{
      Statements;
}
```

SIMPLE IF EXAMPLE

```
void main()
{
    int a=5;
    if(a%2==0)
    {
        printf("number %d is even.",a);
    }
}
```

IF STATEMENT – IF ELSE STATEMENTS

- This is also one of the most useful conditional statement used in C to check conditions.
- Syntax:

```
if(condition){
        true statements;
}
else{
        false statements;
}
```

EXAMPLE

```
void main()
{
        int a=5;
        if(a%2==0){
                printf(“number %d is even”,a);
        }
        else{
                printf(“number %d is odd”,a);
        }
}
```

IF STATEMENT – NESTED IF STATEMENTS

- If within a If condition is called Nested If condition. It is a conditional statement which is used when we want to check more than 1 conditions at a time in a same program.
- Syntax:

IF STATEMENT – NESTED IF STATEMENTS – SYNTAX

```
if(condition){
        if(condition){
                true statement;
        }
        else {
                false statement;
        }
}
else{
        statements;
}
```

EXAMPLE

```
void main()
{
        int a,b,c;
        a=5, b=6, c=8;
        if(a>b){
                if(a>c){
                        printf(“a is big”);
                }
                else{
                        printf(“c is big”);
                }
        }
else{
        if(b>c){
                printf(“b is big”);
        }
        else{
                printf(“c is big”);
}
getch();
}
```

IF STATEMENT – ELSE IF LADDER

- Else if() ladder is similar to if else control statement. Else if() ladder is used to check for another condition if the previously specified condition becomes false.
- Syntax:

IF STATEMENT – ELSE IF LADDER – SYNTAX

```
If(condition){
        Statement;
}
Else if(condition){
        Statement;
}
Else{
        Statement;
}
```

EXAMPLE

```
void main()
{
        int a,b,c;
        a=5, b=6, c=7;
        if(a>b && a>c){
                printf(“a is big”);
        }
        else if(b>c){
                printf(“b is big”);
        }
        else{
                printf(“c is big”);
        }
        getch();
}
```

SWITCH CASE STATEMENTS

- This is a multiple or multiway branching decision making statement. When we use nested if-else statement to check more than 1 condition then the complexity of a program increases in case of a lot of conditions. Thus, the program is difficult to read and maintain. So to overcome this problem, C provides 'switch case'. Switch case checks the value of a expression against a case values, if condition matches the case values then the control is transferred to that point.
- Syntax

SWITCH CASE STATEMENT – SYNTAX

```
switch(expression){
        case exp1:
                statement;
                break;
        case exp2:
                statement;
                break;
        default;
                statement;
}
```

EXAMPLE

```
void main()
{
        char c='a';
        switch(c)
        {
        case 'a':
        case 'e':
        case 'i':
        case 'o':
        case 'u':
        printf("character %c is 
vowel",c);
        break;
        default:
                printf("character %c is 
                        constant",c);
        }
        getch();
}
```

CONDITIONAL OPERATOR

- The ? : Operator is just like an if … else statement except that because it is an operator you can use it within expressions. ? : is a ternary operator in that it takes three values, this is the only ternary operator C has.
- Syntax:

Condition-test? First-expression(true): second-expression(false);

EXAMPLE

```
void main()
{
        int a,b,c,d;
        a=5, b=6, c=7;
        d=((a>b && a>c)?a((b>c)?b:c));
        printf("greater number is ::%d",d);
        getch();
}
```

ITERATIONS / LOOPING STATEMENTS

- 'A loop' is a part of code of a program which is executed repeatedly. A loop is used using condition. The repetition is done until condition becomes true.
- There are two types of looping statements.
  - Entry controlled loop
    - While Loop
    - For Loop
  - Exit controlled loop
    - Do While Loop

ENTRY CONTROL LOOP – WHILE LOOP

- This is an entry controlled looping statement. It is used to repeat a block of statements until condition becomes true.
- Syntax:

```
while (condition){
        Statements;
        Increment/decrement;
}
```

INFORMATIQUE TUTORIAL

EXAMPLE

```
void main()
{
	int I;
	clrscr();
	i=1;
	while(i<=10){
		printf("%d\t",i);
		i++;
	}
	getch();
}
```

ENTRY CONTROL LOOP – FOR LOOP

- This is an entry controlled looping statement. In this loop structure, more than one variable can be initialized. One of the most important feature of this loop is that the three actions can be taken at a time like variable initialization, condition checking and increment/decrement.
- Syntax:

```
for(initialization; test-condition; increment/decrement)
{
        Statements;
}
```

EXAMPLE

```
void main()
{
        int i;
        clrscr();
        i=1;
        for(i=1;i<=20; i+=2)
        {
                printf("%d\t",i);
        }
        getch();
}
```

EXIT CONTROL – DO WHILE LOOP

- This is an exit controlled looping statement. Sometimes, there is need to execute a block of statements first then to check condition. At that time such type of a loop is used. In this, block of statements are executed first and then condition is checked.
- Syntax:

```
do {
        Statements ;( increment/decrement);
} while (condition);
```

EXAMPLE

```
void main()
{
        int I;
        clrscr();
        i=1;
        do
        {
                printf("%d\t",i);
                i++;
        }while(i<=5);
        getch();
}
```

JUMPING STATEMENTS

- There are three types of jumping statements:
  - Break
  - Continue
  - Goto

BREAK STATEMENT

- It is necessary to exit immediately from a loop as soon as the condition is satisfied. When break statement is used inside a loop, then it can cause to terminate from a loop. The statements after break statement are skipped.
- Syntax:

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

EXAMPLE

```
void main()
{
        int i;
        for(i=1;i<=10; i++){
                If(i==5){
                        break;
                }
                else{
                        printf("%d\t",i);
                }
        }
        getch();
}
```

CONTINUE STATEMENT

- It is required to skip a part of a body of loop under specific conditions. The working structure of 'continue' is similar as that of that break statement but difference is that it cannot terminate the loop. It causes the loop to be continued with next iteration after skipping statements in between. Continue statement simply skips statements and continues next iteration.
- Syntax:

```
        if(condition){
                continue;
```

EXAMPLE

```
void main()
{
        int i;
        for(i=1;i<=10; i++){
                If(i==5){
                        continue;
                }
                else{
                        printf("%d\t",i);
                }
        }
        getch();
}
```

GOTO STATEMENT

- The goto statement is a jump statement which jumps from one point to another point within a function or program. The goto statement is marked by label statement. Label statement can be used anywhere in the function above or below the goto statement. Simple break statement cannot work here properly. In this situation, goto statement is used.
- Syntax:

```
if(condition){
        goto err;
}
err:
        statements;
```

EXAMPLE

```
void main()
{
        int i;
        for(i=1;i<=10; i++){
                If(i==5){
                        goto err:
                }
                else{
                        printf("%d\t",i);
                }
        }
        err:
        printf("Error");
}
```

THANK YOU