Chapter 4

Decision Control statement

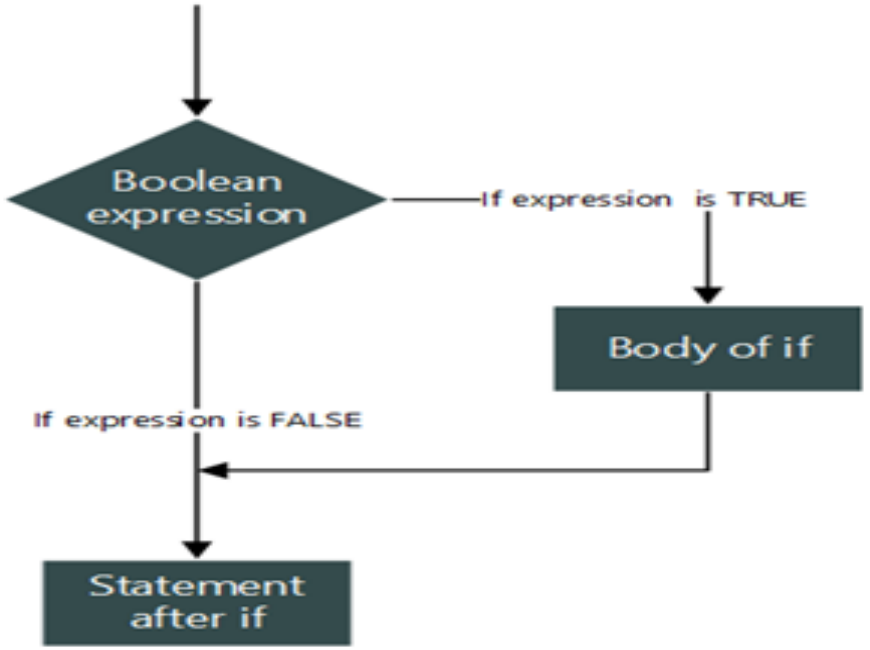
- #include<stdio.h>
- #include <math.h>
- int main(){
- printf("\n%.2f",ceil(3.6));
  printf("\n%f",ceil(3.3));
  printf("\n%f",floor(3.6));
  printf("\n%f",floor(3.2));
- printf("\n%f",sqrt(7));
  printf("\n%f",pow(2,4));
  printf("\n%f",pow(3,3));
  printf("\n%f",sqrt(16)); printf("\n%d",abs(-12)); return 0; }

Introducing to C If Statement

- **C if statement** is a basic control flow structure of C programming language.
- C *if* statement is used when a unit of code need to be executed by a condition true or false.

  If the condition is true, the code in *if* block will execute otherwise it does nothing.

Flowchart of If Statement



- **C If statement syntax:**

```
if(condition)
{
/* unit of code to be executed */
}
```

UNIVERSITY of KIGALI

- The condition can be a variable or expression. The condition must be a boolean expression or value. C *if* statement has it own scope that defines the range over which condition affects, for example:

```
/* all code in bracket are affected by if condition*/
if(x == y)
{
    x++;
    y--;
}

/* only expression x++ is affected by if condition*/
if(x == y)
{
x++;
y--;
}
```

EXAMPLE

```
#include <stdio.h>
main(){
 int b;
 printf("Enter a value:");
 scanf("%d", &b);
 if (b % 2 == 0)
     printf("The value is even\n");
 else
      printf(" The value is odd");
}
```

UNIVERSITY Of KIGALI

```
/ * C program to find maximum between two numbers */
#include <stdio.h>
main()
{
    int num1, num2;
    printf("Enter two numbers:\n ");
    scanf("%d%d", &num1, &num2);
    if(num1 > num2)
    {
        printf("This first number entered is the biggest %d\n ", num1);
    }
    if(num2 > num1)
    {
        printf(" This second number entered is the biggest %d\n ", num2);
    }
    if(num1 == num2)
    {
        printf("Both numbers entered are equal");
    }
}
```

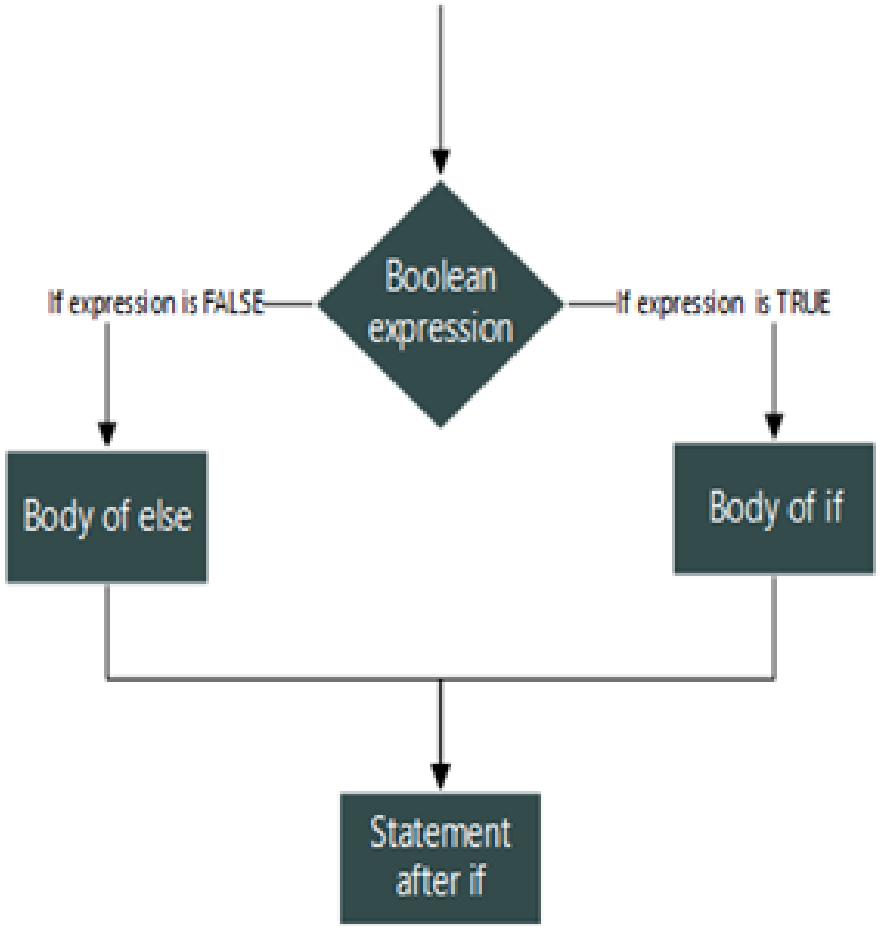
C if-else Statement

- In case you want to use both conditions of *if* statement, you can use if-else statement.
- If the condition of *if* statement is false the code block in *else* will be executed.

The syntax of *if-else* statement:

```
if(condition)
{
/* code block of if statement */
}
Else
{
/* code block of else statement */
}
```

Flowchart of If – else Statement



UNIVERSITY Of KIGALI

TRY ALSO WITH THIS Example 4

```
if( age < 18 )
printf("Minor");
else if( age < 65 )
printf("Adult");
else printf( "Senior Citizen");
```

C else-if Statement

If we want to use several conditions we can use *if-else-if* statement.

The syntax of the if-else-if statement is as follows:

```
if(condition-1)
{       /* code block if condition-1 is true */
}
else if (condition-2)
{   /* code block if condition-2 is true */
}
else if (condition-3)
{   /* code block if condition-3 is true */
}
Else
{   /* code block all conditions above are false */
}
```

UNIVERSITY Of KIGALI

```
#include <stdio.h>
main()
{
    float
physics,chemestry,biology,maths,computer;
    float per;
    printf("Enter five subjects each out of 20
marks:\n");
scanf("%f %f %f %f
%f",&physics,&chemestry,&biology,&maths,&
computer);

per=(physics+chemestry+biology+maths+c
omputer);
    printf("Student marks Percentage =
%.2f\n", per);
```

```
if(per>= 80)
  {
     printf("Grade A");
  }
  else if(per>= 70)
  {
     printf("Grade B");
  }
  else if(per >= 60)
  {
     printf("Grade C");
  }
  else if(per >= 50)
  {
     printf("Grade D");
  }
  else
  {
     printf("Grade F");
  }}
```

Nested if…else statement in C

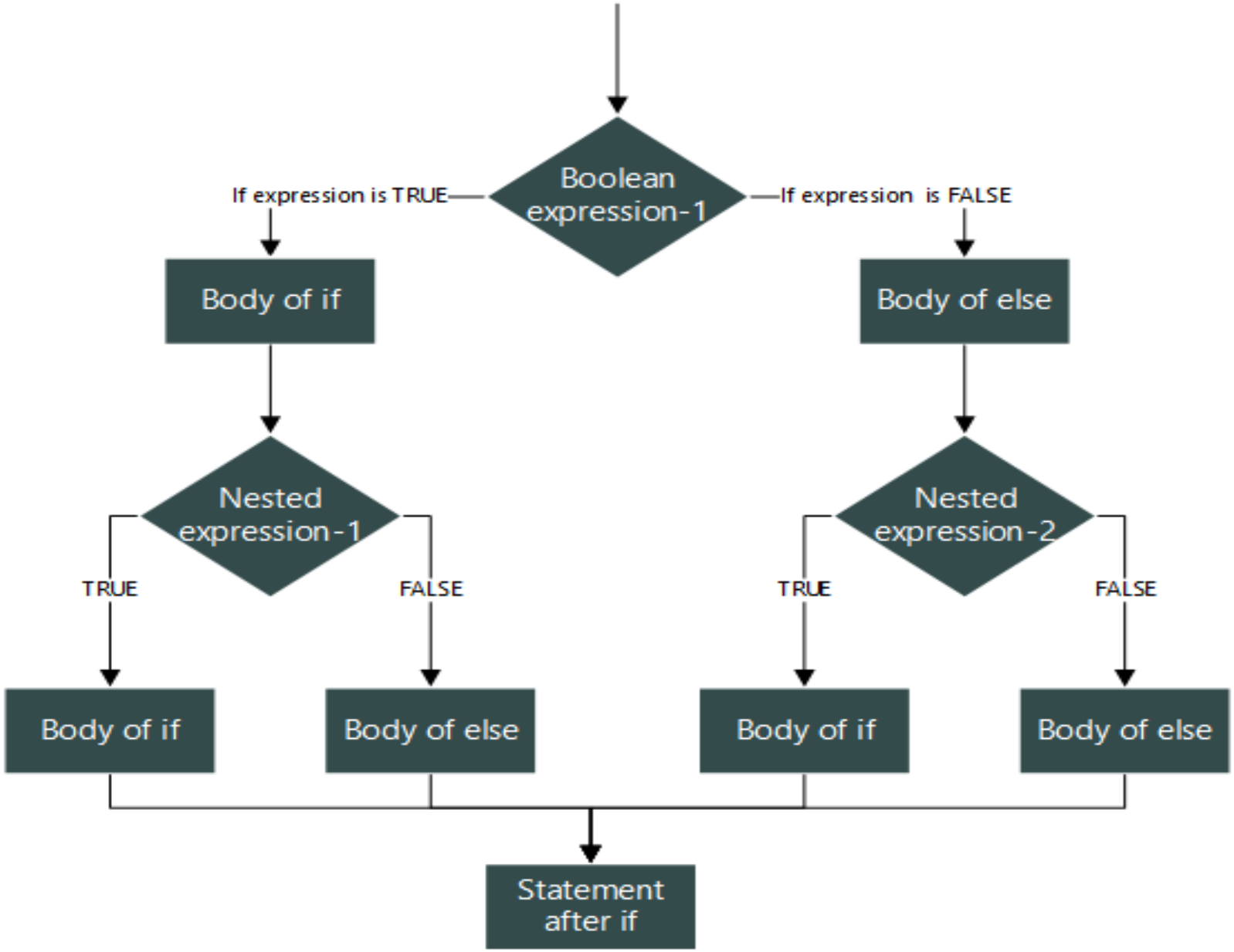
Simple if and if...else...if statements provide a great support to control programs flow.
Simple if is single condition based task i.e. "if some condition is true, then do the task".
In contrast if...else...if statement provides multiple condition checks i.e. "if some condition is true, then do some task. If the condition is false, then check some other condition and do some task. If all conditions fails, then do some default task."

Nested if…else statement in C

```
if (boolean_expression_1)
{
    if(nested_expression_1)
    {
        //* If boolean_expression_1 and  nested_expression_1 both are true*/
    }
    else
    {
        /* If boolean_expression_1 is true  but nested_expression_1 is false*/
    }
    /* If boolean_expression_1 is true*/
}
else
{
    if(nested_expression_2)
    {
        /* If boolean_expression_1 is false  but nested_expression_2 is true*/
    }
    else
    {
        /* If both boolean_expression_1 and  nested_expression_2 is false*/
    }     /* If boolean_expression_1 is false*/
}
```

```
#include<stdio.h>
main()
{
int a,b,c;
printf("enter the values of a,b,c");
scanf("%d,%d,%d",&a,&b,&c);
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");
}
}
printf("a=%d\n b=%d \n c=%d",a,b,c);
getch();
}
```

Flowchart of nested `if...else` statement



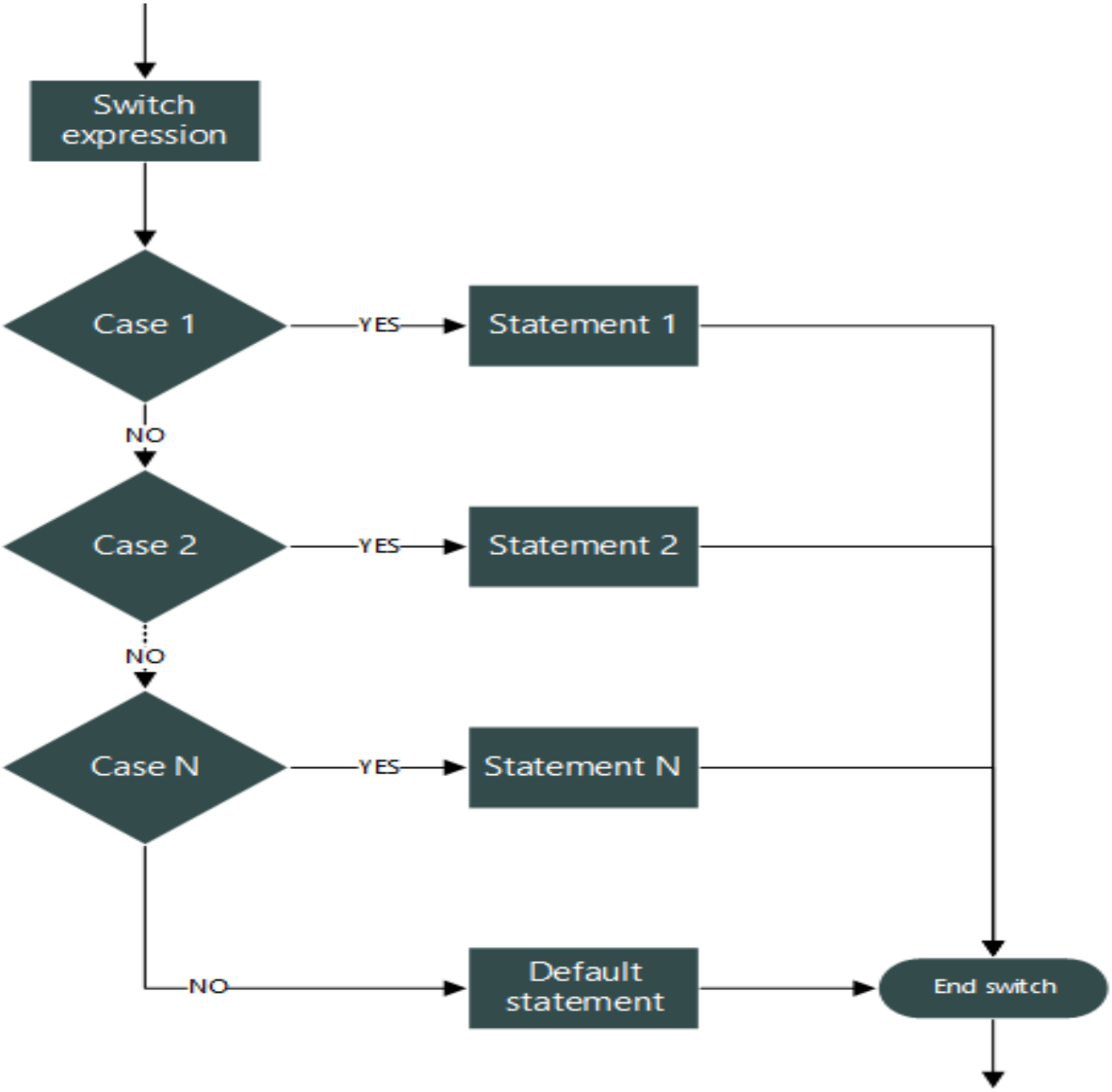
Cont…

- The *else-if* statement is used as the most general way to code multi-way decisions.
- The conditions (condition-1, condition2 ...etc.) are evaluated in sequence from top to bottom.
- If a *condition* is true, the statement or code block associated with it is executed, and this terminates the whole chain.
- In theory, you can have as many conditions as you want but it is not recommended.
- If the *else-if* have many conditions, it is good practice to use the C switch statement.
- The last *else* handles the "none of above" or default case where none of the conditions is met.
- Sometimes, the last *else* is omitted to indicate that no explicit action for handle the default case.

c *switch* statement

- C *switch* statement is a multiway decisions that tests whether a variable or expression matches one of a number of constant integer values, and branches accordingly.
- The switch case statement is a better way of writing a program when a series of if else occurs.

This is the flow chart of the *C switch statement:*



Switch statement

- The following illustrates the C switch statement syntax:

```
switch(expression)
{
    case 1:
        /* Statement/s */
        break;
    case 2:
        /* Statement/s */
        break;
    case n:
        /* Statement/s */
        break;
    default:
        /* Statement/s */
}
```

Rules for switch statements

- Values for 'case' must be integer or character constants.
- The order of the 'case' statements is unimportant.
- The default clause may occur first (convention places it last)

Let's examine the C switch statement syntax in more detail:

- In C *switch* statement, the selection is determined by the value of an expression that you specify, which is enclosed between the parentheses after the keyword *switch*. The data type of value, which is returned by expression, must be an integer value otherwise the compiler will issue an error message.
- The case constant expression must be a constant integer value.
- When a *break* statement is executed, it causes an immediate exit from the *switch*. The control pass to the statement following the closing brace for the switch.

  The *break* statement is not mandatory, but if you don't put a *breaks* statement at the end of the statements for a case, the statements for the next case in sequence will be executed as well, through to whenever another break is found or the end of the switch block is reached. This can lead some unexpected program logic happens.
- The *default* statement is the default choice of the switch statement if all case statements are not satisfied with the expression. The ***break* after** the **default statements** is not necessary unless you put another case statement below it.

If…else…if vs switch…case

- Complexity
- Depending on situation if...else...if as well as switch...case can be simple or complex. Complexity of if...else...if statement increases with increase in conditions. At one stage if statements become confusing with increase in level of ladder if conditions. Nesting of if...else...if also increases the level of complexity.
- Compared to if...else...if statements switch...case is easy to read, code and maintain. However, switch can get confusing if nested.

Use if...else...if statement when –

- There are conditions instead of list of choices.
- There are few number of conditions.

Use switch...case when –

- There is a list of choices, from which you need to take decision.
- Choices are in the form of integer, character or enumeration constant.

Example of C Switch Statement

```
#include <stdio.h>
 main()
{
    int color ;
    printf("Please choose a
   color(1: red,2: green,3:
   blue):\n");
    scanf("%d", &color);
    switch (color)
{
        case 1:
            printf("you chose red
   color\n");
        break;
        case 2:
            printf("you chose green
   color\n");
        break;
        case 3:
            printf("you chose blue
   color\n");
        break;
        default:
            printf("you did not
   choose any color\n");
}    return 0;
}
```

```
#include <stdio.h>
main()
{
int menu, numb1, numb2, total;
printf("enter in two numbers -->");
scanf("%d %d", &numb1, &numb2 );
printf("enter in choice\n");
printf("1 = addition\n");
printf("2 = subtraction\n");
scanf("%d", &menu );
switch( menu )
{
case 1: total = numb1 + numb2; break;
case 2: total = numb1 - numb2; break;
default: printf("Invalid option selected\n");
}
if( menu = = 1 )
printf("%d plus %d is %d\n", numb1, numb2, total );
else if( menu = = 2 )
printf("%d minus %d is %d\n", numb1, numb2, total );
}
```

UNIVERSITY of KIGALI