SIEC: BASIC C PROGRAMMING

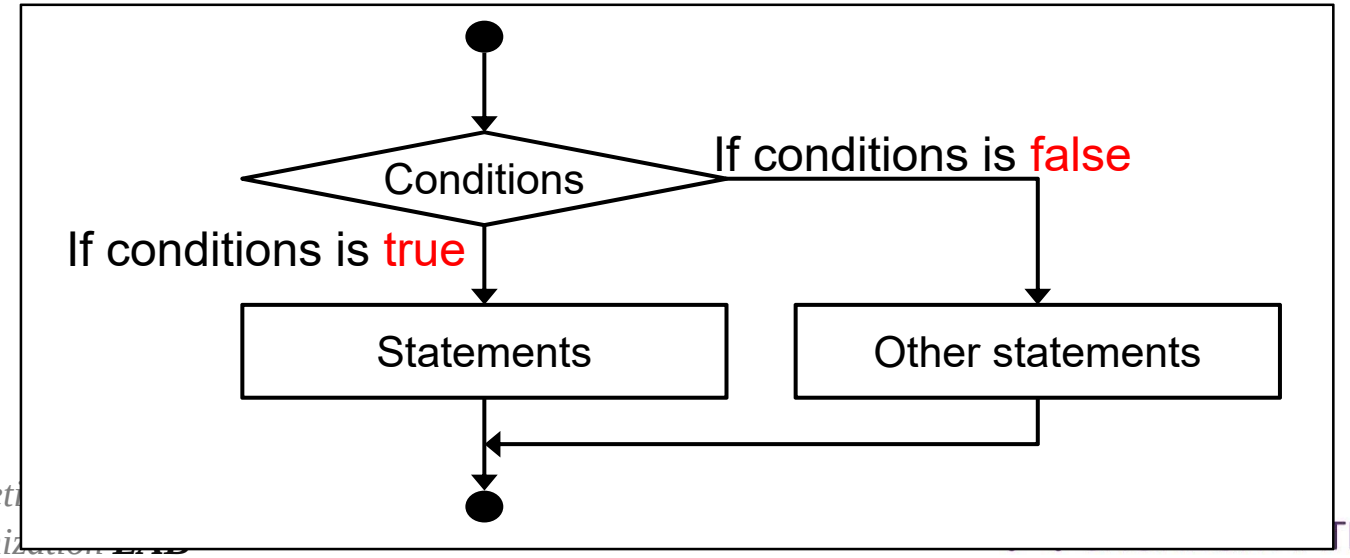
L #08: DECISION MAKING IN C

Seung Beop Lee
School of International Engineering and Science
CHONBUK NATIONAL UNIVERSITY

Decision Making in C

Decision Making in C

- Decision making structures
  - The following flow chart is the general form of a typical decision making structure found in most of the programming languages.
  - Decision making structures require that the programmer specifies one or more conditions to be evaluated or tested by the program, along with a statement or statements to be executed if the condition is determined to be true, and optionally, other statements to be executed if the condition is determined to be false.
  - C programming language assumes any non-zero and non-null values as true, and if it is either zero or null, then it is assumed as false value.



Decision Making in C

- Decision making structures
  - C programming language provides the following types of decision making statements:
    - if statement
    - if...else statement
    - if...else if...else Statement
    - Nested if statements
    - switch statement

Decision Making in C

- if statement
  - An if statement consists of a boolean expression followed by one or more statements.
- Syntax of an if statement
  - The syntax of an **if** statement in C programming language is:

```
if(boolean expression)
{
   /* statement(s) will execute if the boolean expression is true */
}
```

  - If the **boolean condition (or expression)** evaluates to **true**, then the **block of code** inside the if statement will be executed.
  - If **boolean condition (or expression)** evaluates to **false**, then the **other statements** after the end of the if statement (after the closing curly brace) will be executed.

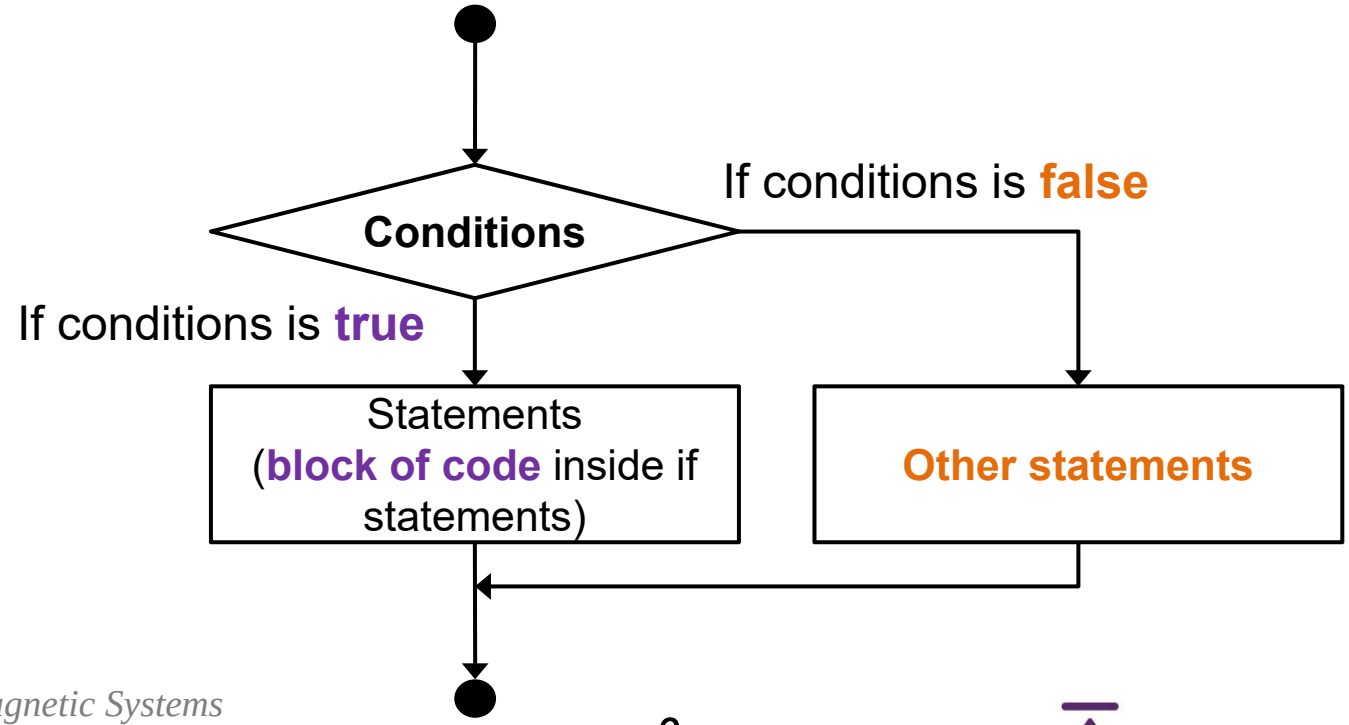
```
/* check the boolean condition using if statement */
if( a < 20 )
{
    /* if condition is true then print the following */
    printf("a is less than 20\n" );
}
printf("value of a is : %d\n", a);
```

Block of code

Other statements

Decision Making in C

- Flow chart of an if statement
  - If the **conditions** (i.e. boolean expression) evaluates to **true**, then the **block of code** inside the if statement will be executed.
  - If the **conditions** (i.e. boolean expression) evaluates to **false**, then the **other statements** after the end of the if statement (after the closing curly brace) will be executed.



Decision Making in C

- Example of an if statement
  - The following example is in order to understand the **if statement** available in C:

```c
#include <stdio.h>

main()
{
    /* local variable definition */
    int a = 10;

    /* check the boolean condition using if statement */
    if (a < 20)
    {
        /* if condition is true then print the following */
        printf("a is less than 20\n");
    }

    printf("value of a is : %d\n", a);

    return 0;

}
```

```
C:\Users\SBLEE\source\re
a is less than 20
value of a is : 10
```

Decision Making in C

- Example of an if statement
  - The following example is in order to understand the **if statement** available in C:

```c
#include <stdio.h>

main()
{
    /* local variable definition */
    int a = 10;

    /* check the boolean condition using if statement */
    if (a < 20)
    {
        /* if condition is true then print the following */
        printf("a is less than 20\n");
    }

    printf("value of a is : %d\n", a);

    return 0;

}
```

```
C:\Users\SBLEE\source\re
a is less than 20
value of a is : 10
```

Change

```c
#include <stdio.h>

main()
{
    /* local variable definition */
    int a = 30;

    /* check the boolean condition using if statement */
    if (a < 20)
    {
        /* if condition is true then print the following */
        printf("a is less than 20\n");
    }

    printf("value of a is : %d\n", a);

    return 0;

}
```

```
C:\Users\SBLEE\source
value of a is : 30
```

Decision Making in C

- Example of an if statement
  - The following example is in order to understand the **if statement** available in C:

```c
#include <stdio.h>

main()
{
    /* local variable definition */
    int a = 30;

    /* check the boolean condition using if statement */
    if (a < 20)
    {
        /* if condition is true then print the following */
        printf("a is less than 20\n");
    }

    printf("value of a is : %d\n", a);

    return 0;

}
```

```
C:\Users\SBLEE\source
value of a is : 30
```

Decision Making in C

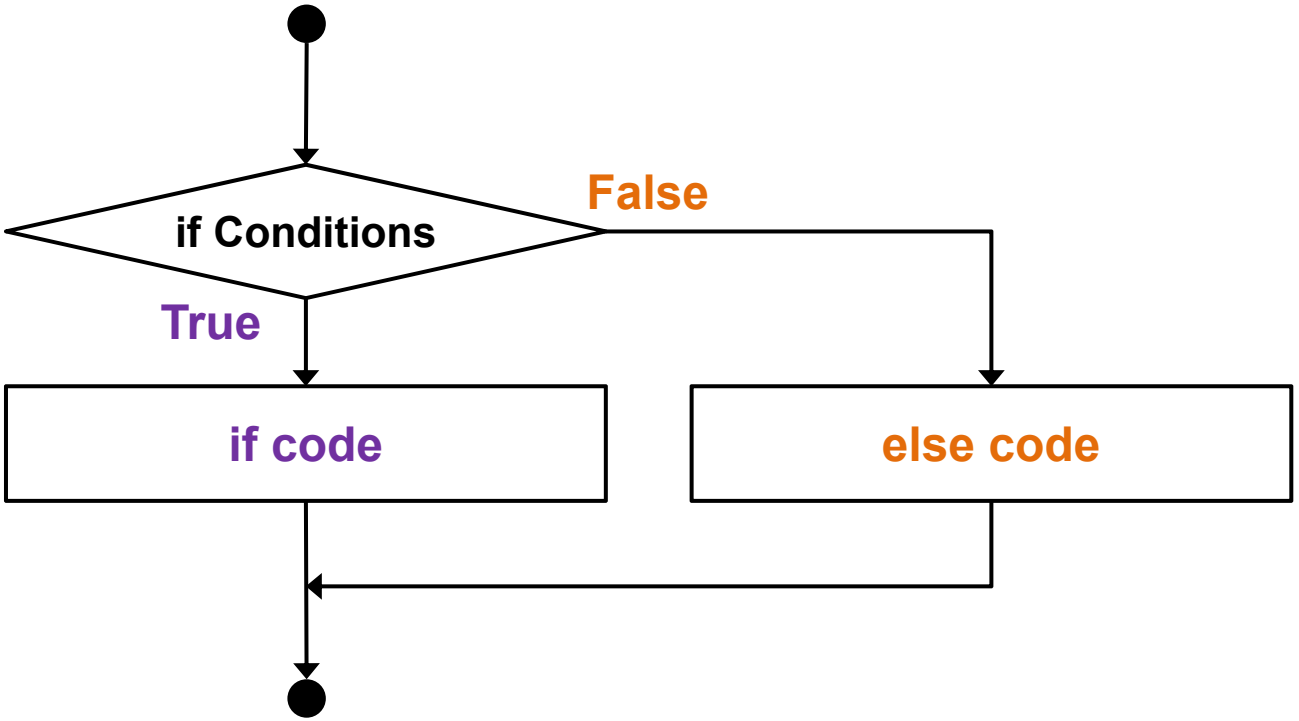
- if… else statement
  - An if statement can be followed by an optional else statement, which executes when the Boolean expression is false.
- Syntax of an if… else statement
  - The syntax of an **if… else** statement in C programming language is:

```
if(boolean_expression)
{
   /* statement(s) will execute if the boolean expression is true */
}
else
{
  /* statement(s) will execute if the boolean expression is false */
}
```

  - If the **boolean expression (or conditions)** evaluates to true, then the if block of code will be executed, otherwise else block of code will be executed.

Decision Making in C

- Flow chart of an if… else statement
  - If the **conditions** (i.e. boolean expression) evaluates to **true**, then the if code will be executed, otherwise else code will be executed.



Decision Making in C

- Example of an if… else statement
  - The following example is in order to understand the **if… else statement** available in C:

```c
#include <stdio.h>

main()
{
    /* local variable definition */
    int a = 100;

    /* check the boolean condition */
    if (a < 20)
    {
        /* if condition is true then print the following */
        printf("a is less than 20\n");
    }

    else
    {
        /* if condition is false then print the following */
        printf("a is not less than 20\n");
    }

    printf("value of a is : %d\n", a);

    return 0;

}
```

C:\Users\SBLEE\source\repos

```
a is not less than 20
value of a is : 100
```

Decision Making in C

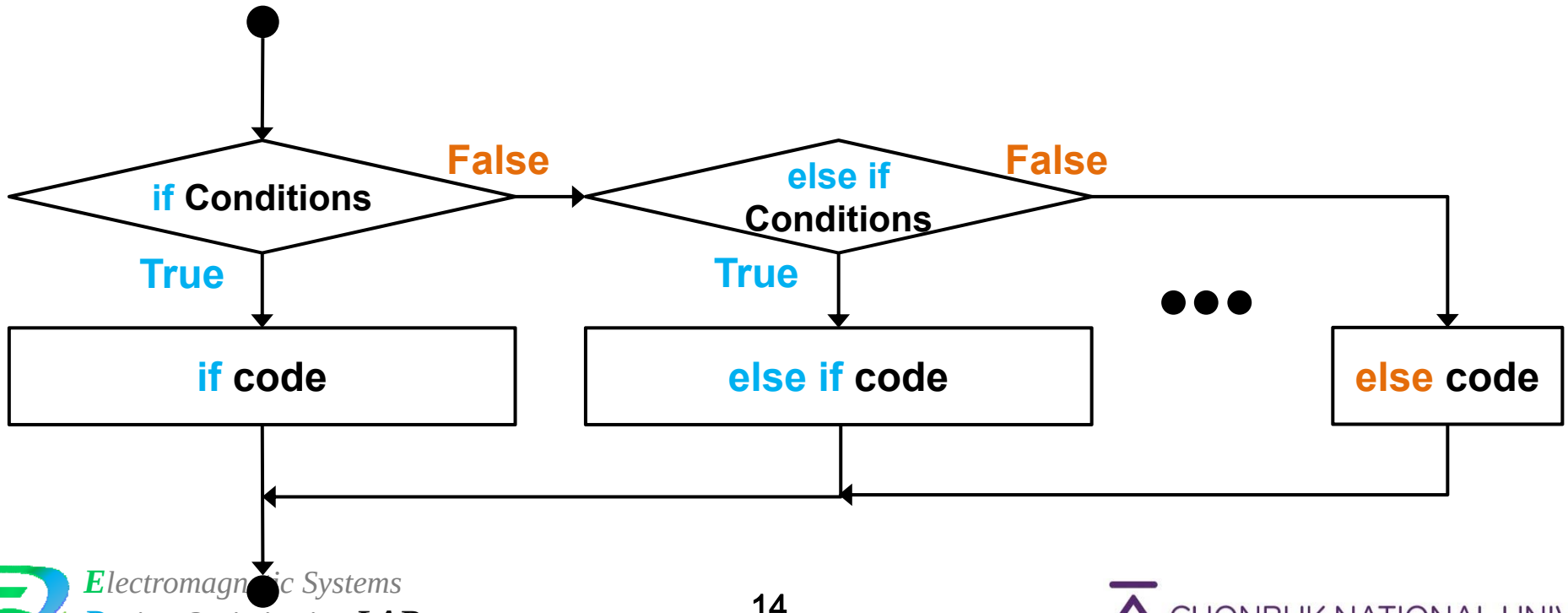
- if… else if… else statement
  - An if statement can be followed by an optional else if… else statement, which is very useful to test various conditions using single if...else if statement.
  - When using if…else if…else statements, there are few points to keep in mind:
    - An **if** can have zero or one **else** and it must come after any **else if**'s.
    - An **if** can have zero to many **else if**'s and they must come before the **else**.
    - Once an **else if** succeeds, the remaining **else if's** or **else's** will **not** be tested.

- Syntax of an if… else if… else statement

```
if(boolean_expression 1)
{
   /* Executes when the boolean expression 1 is true */
}
else if( boolean_expression 2)
{
   /* Executes when the boolean expression 2 is true */
}
else if( boolean_expression 3)
{
   /* Executes when the boolean expression 3 is true */
}
else
{
   /* executes when the none of the above condition is true */
}
```

Decision Making in C

- Flow chart of an if… else if… else statement
  - If the **if conditions** (i.e. boolean expression) evaluates to true, then the if code will be executed. Otherwise, else if conditions will be evaluated.
  - If the **else if conditions** evaluates to true, then the else if code will be executed. Otherwise, next else if conditions will be evaluated.
  - If **all** of the conditions evaluates to false, then the else code will be executed.



Decision Making in C

- Example of an if… else if… else statement
  - The following example is in order to understand the **if… else if… else statement** available in C:

```c
#include <stdio.h>

main()
{
    /* local variable definition */
    int a = 100;

    /* check the boolean condition */
    if (a == 10)
    {
        /* if condition is true then print the following */
        printf("Value of a is 10\n");
    }
    else if (a == 20)
    {
        /* if else if condition is true */
        printf("Value of a is 20\n");
    }
    else if (a == 30)
    {
        /* if else if condition is true */
        printf("Value of a is 30\n");
    }
    else
    {
        /* if none of the conditions is true */
        printf("None of the values is matching\n");
    }

    printf("Exact value of a is: %d\n", a);
    return 0;

}
```

C:\Users\SBLEE\source\repos\Project1\De

```
None of the values is matching
Exact value of a is: 100
```

Decision Making in C

- Nested if Statement
  - It is always legal in C programming to **nest** if-else statements, which means you can use one **if** or **else if** statement **inside** another **if** or **else if** statement(s).
- Syntax of a **nested if** statement
  - The syntax for a **nested if** statement is as follows:

```
if( boolean_expression 1)
{
   /* Executes when the boolean expression 1 is true */
   if(boolean_expression 2)
   {
      /* Executes when the boolean expression 2 is true */
   }
}
```

Decision Making in C

- Example of a Nested if statement
  - The following example is in order to understand the **nested if** statement available in C:

```c
#include <stdio.h>

main()
{
    /* local variable definition */
    int a = 100;
    int b = 200;

    /* check the boolean condition */
    if (a == 100)
    {
        /* if condition is true then check the following */
        if (b == 200)
        {
            /* if condition is true then print the following
            printf("Value of a is 100 and b is 200\n");
        }
    }
    printf("Exact value of a is : %d\n", a);
    printf("Exact value of b is : %d\n", b);

    return 0;

}
```

```
C:\Users\SBLEE\source\repos\Project1\De
Value of a is 100 and b is 200
Exact value of a is : 100
Exact value of b is : 200
```

Decision Making in C

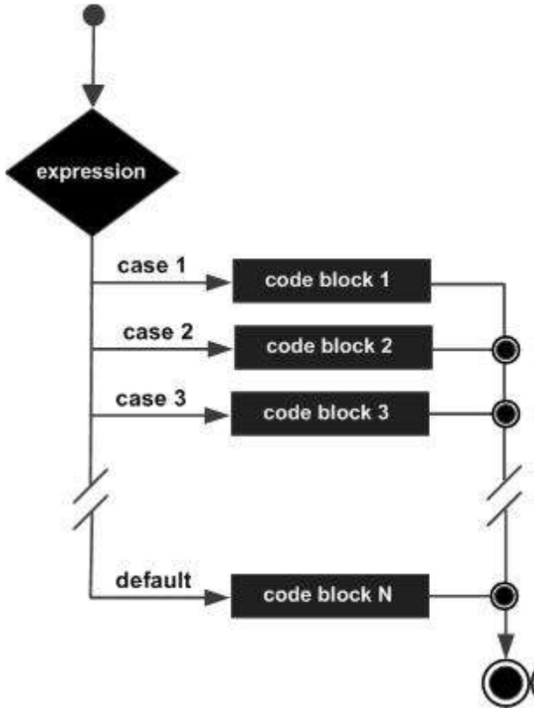
- switch Statement
  - A **switch** statement allows a variable to be tested for equality against a list of values.
  - Each value is called a **case**, and the variable being switched on is checked for each switch case.

Decision Making in C

- switch Statement
  - The following rules apply to a **switch** statement:
    - The **expression used in a switch statement** must have an **integral or enumerated type**, or be of a class type in which the class has a single conversion function to an integral or enumerated type.
    - You can have any number of case statements within a switch. Each case is followed by the value to be compared to and a colon.
    - The **constant-expression** for a case must be the same data type as the variable in the switch, and it must be a constant or a literal.
    - When the variable being switched on is equal to a case, the statements following that case will execute until a **break statement** is reached.
    - When a **break** statement is reached, the switch terminates, and the flow of control jumps to the next line following the switch statement.
    - Not every case needs to contain a **break**. If **no break** appears, the flow of control will fall through to subsequent cases until a break is reached.
    - A switch statement **can** have an optional **default** case, which must appear at the end of the switch. The default case can be used for performing a task when none of the cases is true. No break is needed in the default case.

Decision Making in C

- Flow chart and example of the switch statement
  - The switch statement is used to select one case among several cases.
  - Switch statement is composed of the switch condition, each case, statements, and break statement.



Project1

```c
#include <stdio.h>

main()
{
    /* local variable definition */
    char grade = 'B';

    /* switch statement declaration */
    switch (grade)
    {
        case 'A':
            printf("Excellent!\n");
            break;

        case 'B':
            printf("Good!\n");
            break;

        case 'C':
            printf("Well done\n");
            break;

        case 'D':
            printf("You passed\n");
            break;

        case 'F':
            printf("Better try again\n");
            break;

        default:
            printf("Invalid grade\n");
    }

    printf("Your grade is %c\n", grade);

    return 0;

}
```

C:\Users\SBLEE\sou

```
Good!
Your grade is B
```

Thank You