Chapter 4
Control structure

Thanh-Sach Le

Content

- Application and usage of control structure
- Statements and program
- If statement
- If-else statement
- Nested if-else
- Switch-case statement
- Enumeration and switch-case
- Conclusion

Application and usage of control structure

Application and usage of control structure

- All useful applications in real life use control structure
- **Example 1:** Input a date (including, they day, month and year).
  - A well-made program MUST CHECK if the inputted date is valid or not. One must not assume that the date entered by user is always in our expected domain.
  - To check if the inputted month is valid, it's possible that the following structure was used:

```
if ( (month < 1) hoặc (month > 12) )
     Handle invalid month error
endif
```

Application and usage of control structure

- **Example 2:** Solve a quadratic equation of the form $\mathrm{A}x^2 + Bx + C = 0$
  - The program should allow the user to enter the three coefficients A, B and C of the equation.
    - A and B can be zero or non-zero.
    - The entered equation can be a quadratic equation or linear equation.
  - => The program can be erroneous if we don’t check if A or B is zero or not.
  - => The control structure can be used to perform the check.

Application and usage of control structure

- **Example 3:** Find the tax rate of an individual in accounting
  - The tax table in 2016 is as follow:

Tier	Monthly income	Tax rate	Tax	
			Method 1	Method 2
1	$income \le 5M$	5%	0M + 5% of TI	5% TI
2	$5M < income \le 10M$	10%	0.25M+10% of TI over 5M	10% TI-0.25M
3	$10M < income \le 19M$	15%	0.75M+15% of TI over 10M	15% TI - 0.75M
4	$18M < income \le 32M$	20%	1.95M+20% of TI over 18M	20% TI - 1.65M
5	$32M < income \le 52M$	25%	4.75M+25% of TI over 32M	25% TI - 3.25M
6	$52M < income \le 80M$	30%	9.75M+30% of TI over 52M	30% TI - 5.85M
7	$80M < income$	35%	18.15M+35% of TI over 80M	35% TI - 9.85M

M: million VND, TI: taxed income

Application and usage of control structure

- **Example 4:** Implement the interaction between a user and a software (may have graphics interface or may not)
  - The program must listen to all sort of events occurring in a software
    - With graphics interface: left mouse, right mouse, middle mouse, menu A chosen, menu B chosen, etc.
    - On console: the ID of a task (number, string) entered.
  - The program must execute all different tasks based on what event/ID was chosen by the user.
  - ⇒ We need control structure (preferably switch because there are a lot of cases)

Statements and program

- What is a statement?
  - Is a programming line written by a programming language.
  - In C++, the end of a statement is marked with a semi-colon (;), similar to the usage of the dot symbol in natural language (.).
- Types of statement:
  - Single statement, consists of simple statements:
    - Variable declaration.
    - Assignment statement.
    - Function call.
    - Etc.

Statements and program

- Types of statement:
  - Single statement.
  - Composite statement, a list of statements to be executed together, sandwiched by a pair of curly braces { and }:

```
{
    <statement 1>
    <statement 2>
    //...
}
```

  - The control statements: if, if-else, switch, for, while, do-while, etc. are considered composite statements.

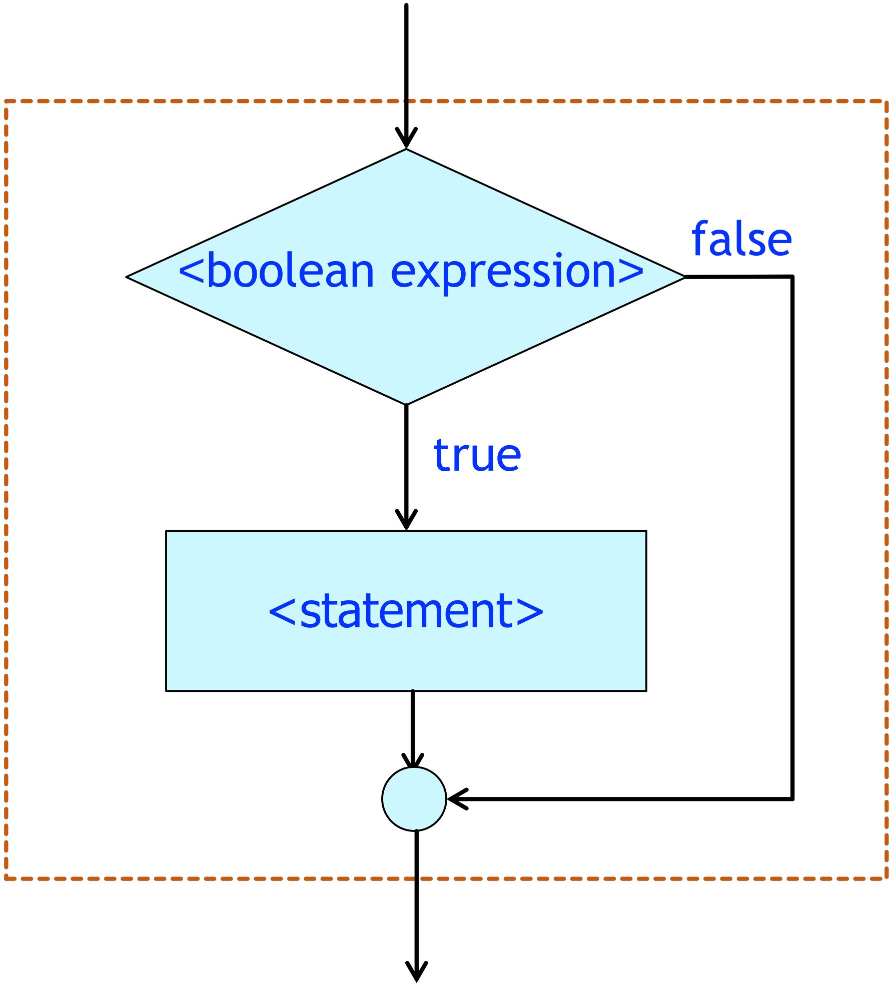
Statements and program

- Types of statement:
  - Single statement.
  - Composite statement.
  - Empty statement:
    - Only has the semi-colon (;) at the end.
    - It is allowed in C++ but not usually used.

Statements and program

- Program
  - A program can be considered an ordered sequence of statements:
    - Each of them can be a single statement or a composite statement (including control statement and loop statement)
  - The computer will execute each statement, one by one from first statement to the last one.
  - This way of controlling a program is called sequential execution.
  - Two another ways of controlling the flow of a program:
    - Branching statements: if, if-else, switch.
    - Loop statements: for, while, do…while.

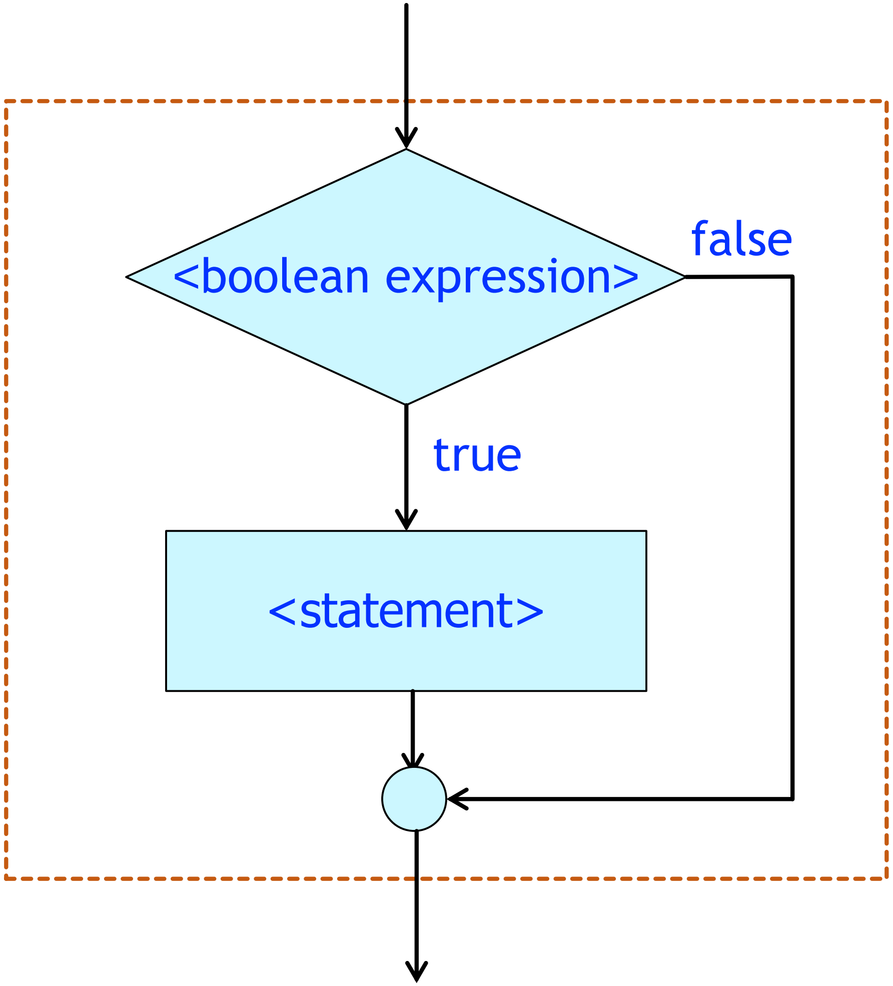
The `if` statement - concept



<conditional statement>: A boolean expression or an expression convertible to a boolean expression

<statement>: a single statement or a composite statement

The if statement - concept



Execution of the **if** statement:

(1) The boolean expression is evaluated

(2) If the evaluation result is true then the program will run <statement>. Otherwise, the program will run the statements after if

The `if` statement - syntax

```
if (<boolean expression>) <statement>
```

```
if (<boolean expression>)
        <statement>
```

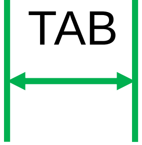
```
if (<boolean expression>) {
    <statement>
    //can add more statements
}
```

```
if (<boolean expression>)
{
    <statement>
    //can add more statements
}
```

The if statement – syntax + coding style

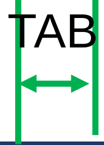
```
if (<boolean expression>) <statement>
```

```
if (<boolean expression>)
        <statement>
```



```
if (<boolean expression>) {
    <statement>
    //can add more statements
}
```

```
if (<boolean expression>)
{
    <statement>
    //can add more statements
}
```



The if statement

- Example: verifying if a date (including month and year) is valid

```
if( (month <= 0) || (month > 12) )
    exit(1);
```

The program will end if the month is invalid

The if statement

- Example: verifying if a date (including month and year) is valid
- Instead of terminating the program, we can also assign the date to be a default date:

```
if( (month <= 0) || (month > 12) ){
    date = 1;
    month = 1;
    year = 1970;
}
```

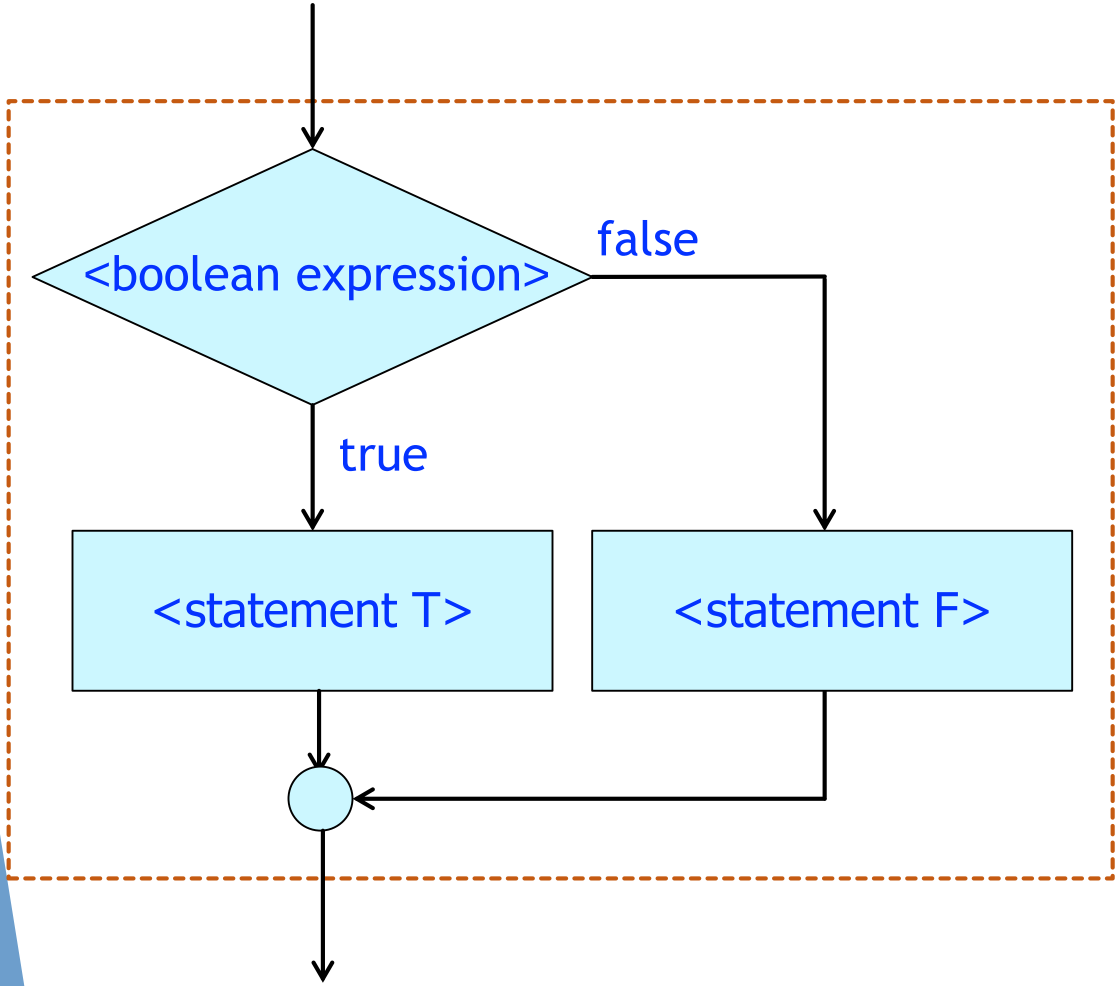
The `if` statement

- Example: verifying if a date (including month and year) is valid
- Instead of terminating the program, we can also assign the date to be a default date:

```
if( (month <= 0) || (month > 12) )
    date = 1;
    month = 1;
    year = 1970;
```

- It is not logically correct to write the above block without the curly braces. The above block means that the month and the year will always be assigned to 1 and 1970 **regardless** of the month being invalid or not.

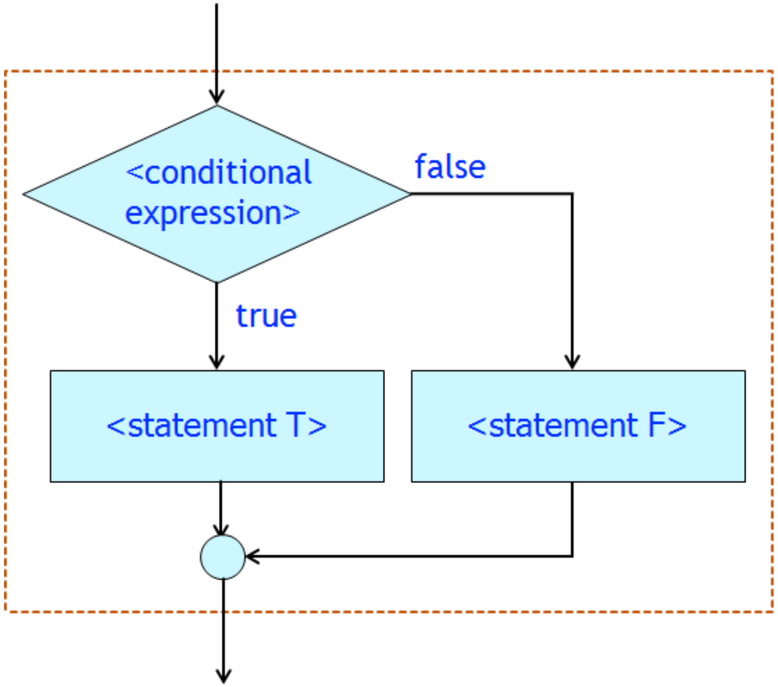
The `if-else` statement - concept



<conditional statement>: A boolean expression or an expression convertible to a boolean expression

<statement T>, <statement F>: a single statement, a composite statement or an empty statement

The `if-else` statement - concept



Execution of the **if-else** statement:

(1) The boolean expression is evaluated

(2) If the evaluation result is true then the program will run <statement T>. Otherwise, the program will run <statement F>. The program will execute the statement after if-else when <statement T> or <statement F> is done.

The if-else statement - syntax

```
if (<boolean expression>)
     <statement T>
else
     <statement F>
```

```
if (<boolean expression>)    <statement T>
else  <statement F>
```

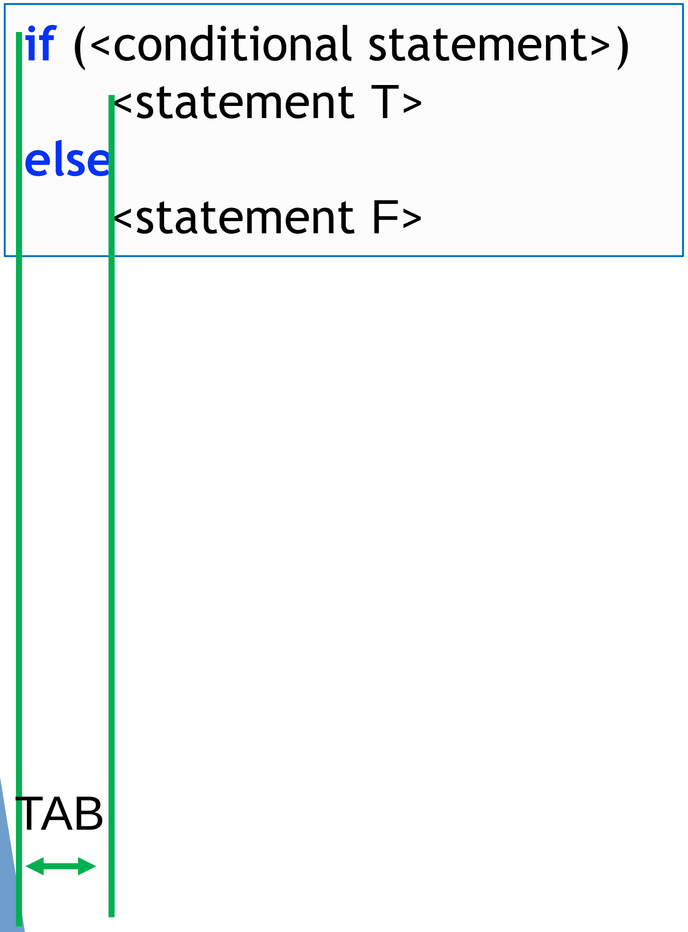
Note: <statement T> and <statement T> must end with ;

The `if-else` statement – syntax + coding style

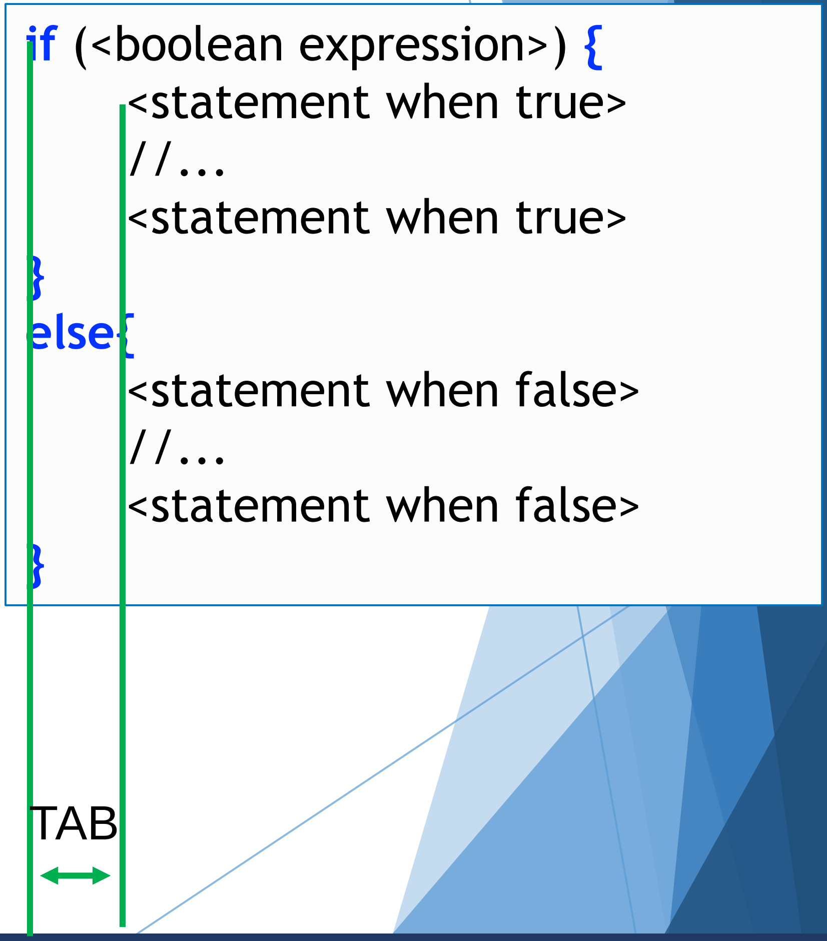
```
if (<conditional statement>)
      <statement T>
else
      <statement F>
```

```
if (<boolean expression>) {
      <statement when true>
      //...
      <statement when true>
}
else{
      <statement when false>
      //...
      <statement when false>
}
```

The `if-else` statement – syntax + coding style



```
if (<conditional statement>)
	<statement T>
else
	<statement F>
```



```
if (<boolean expression>) {
	<statement when true>
	//...
	<statement when true>
}
else{
	<statement when false>
	//...
	<statement when false>
}
```

The if-else statement – syntax + coding style

```
if (<boolean expression>)    <statement T>
else  <statement F>
```

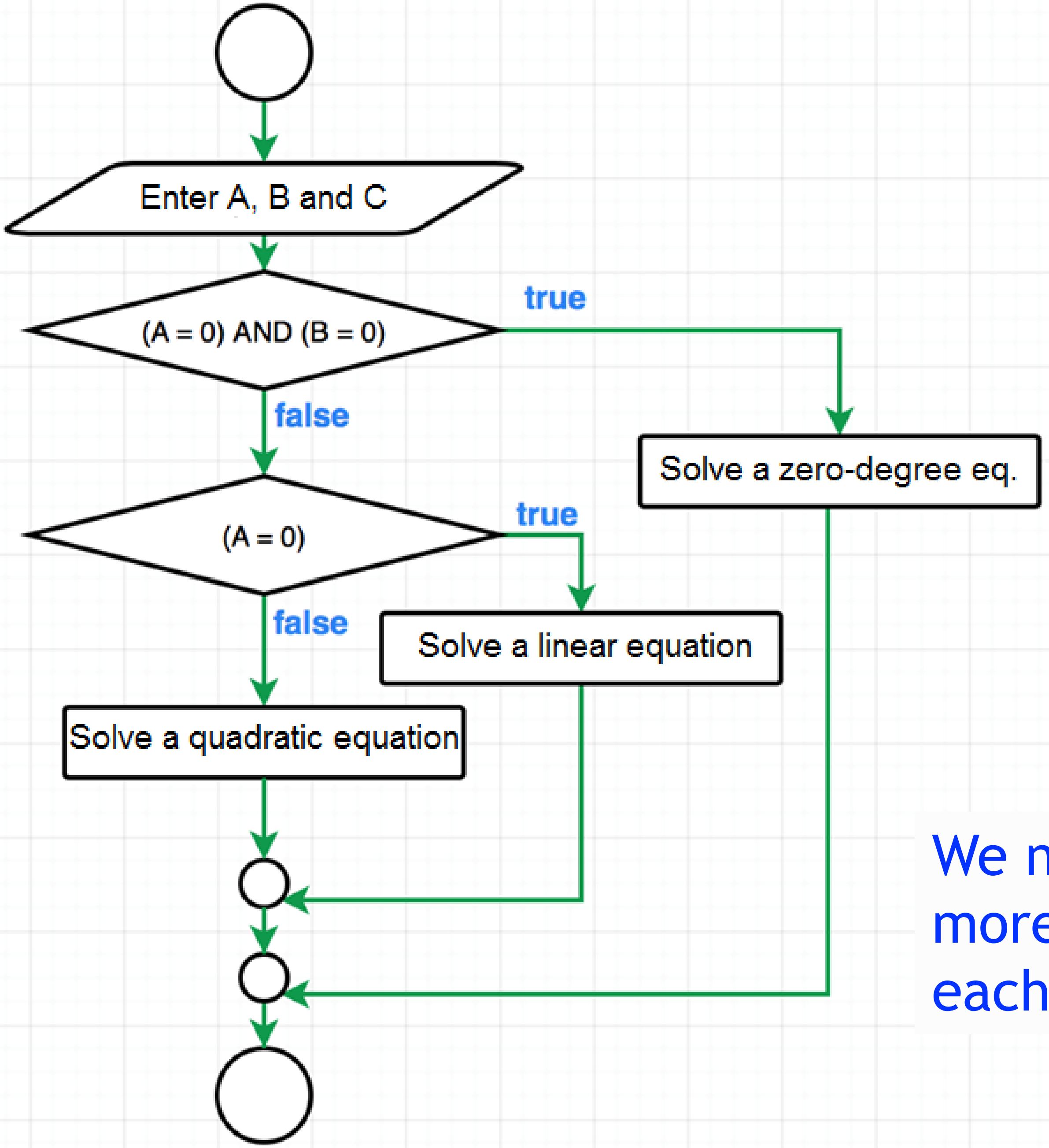
This should only be used if both are simple statements

The `if-else` statement

- Solving quadratic equation
- Analysis:
  - The user needs to enter 3 coefficients: A, B and C
  - Therefore:
    - A, B: can be 0.
    - The equation can become a linear equation or zero-degree equation.
    - There are three cases to check.

The `if-else` statement

- Solving quadratic equation
- Analysis:
  - The three cases are:
    - (1) Zero-degree, A = 0 and B = 0:
      - Use C to check if there are no solution or infinitely many solutions.
    - (2) First-degree (linear), A = 0 but B != 0:
      - Solve a linear equation.
    - (3) Second-degree (quadratic), A != 0:
      - Solve a quadratic equation.



We need to analyze more on how to solve each type of equation

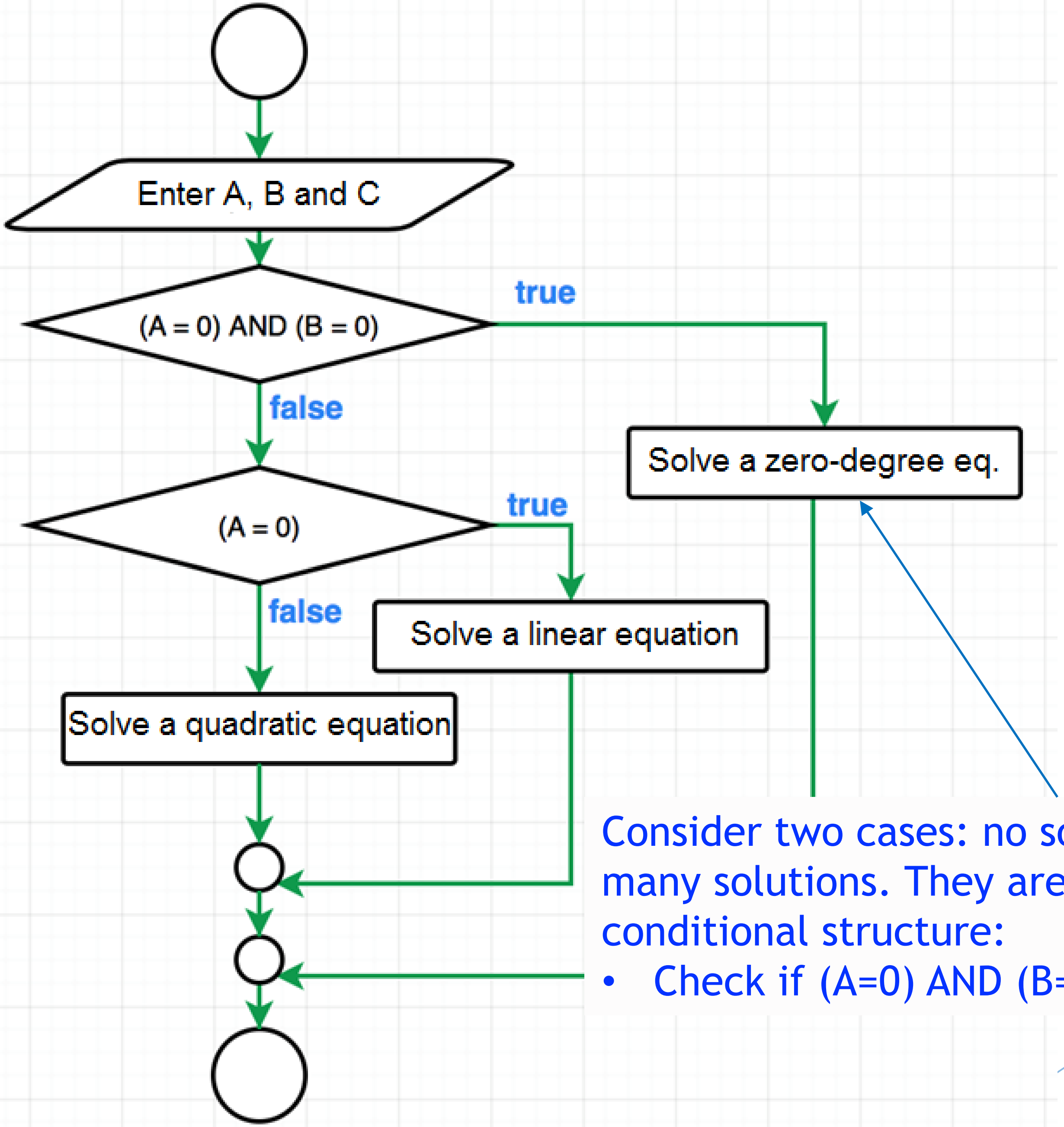
The if-else statement – Example

- Solving quadratic equation

```
#include <iostream>
using namespace std;
int main(){
    float a, b, c, delta;

    cout << “Enter a, b, c: \n";
    cin >> a >> b >> c;

    delta = b*b - 4*a*c;
    if(delta < 0)
        printf(“There is no solution\n");
    else
        printf(“There are at least one solution\n");

    return 0;
}
```

Nested `if-else` statement – Application

- Most of the useful applications in real life does not consist of only simple, separate cases (sequential).
- The applications must also check nested conditions.
- Example: solving quadratic equation:
    - If A and B are zero:
        - The program must then check if C is 0 or not.
        - If C = 0, the equation has infinitely many solutions.
        - If not, the equation has no solution.



Consider two cases: no solution and infinitely many solutions. They are nested in a parent conditional structure:

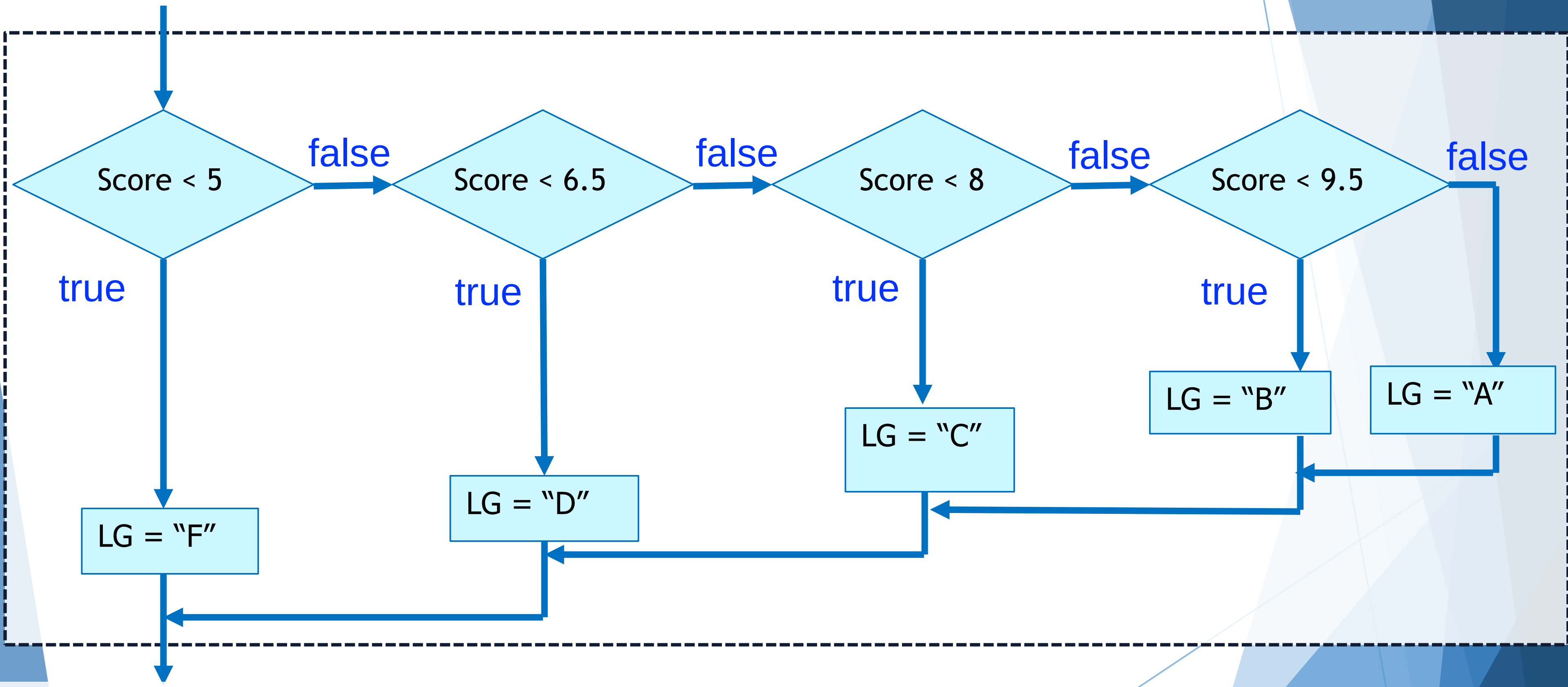
- Check if (A=0) AND (B=0)

Nested `if-else` statement – Application

- **Problem:** calculate the letter grade of each student according to their numerical scores
- There are 5 letter grades:
  - F (fail): grade lies in the [0, 5) interval
  - D: grade lies in the [5, 6.5) interval
  - C: grade lies in the [6.5, 8) interval
  - B: grade lies in the [8, 9.5) interval
  - A: grade lies in the [9.5, 10] interval

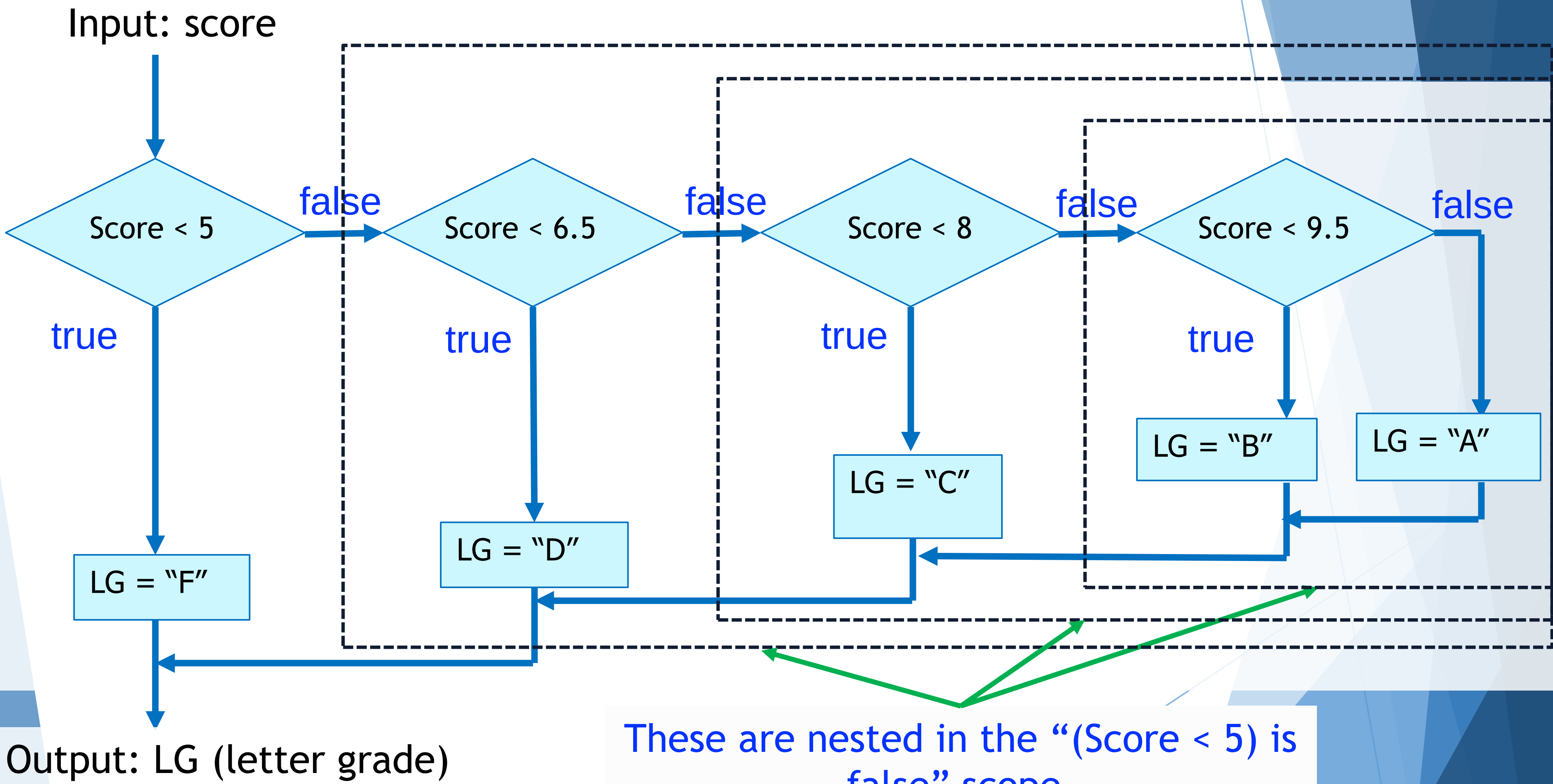
Nested `if-else` statement – Application

Input: score



Output: LG (letter grade)

Nested `if-else` statement – Application



These are nested in the "(Score < 5) is false" scope.

Nested `if-else` statement – Syntax

There are many representations

```
if (<boolean expression 1>)    <statement 1>
else if (<boolean expression 2>)    <statment 2>
else if (<boolean expression 3>)    < statement 3>
else    <statement 4>
```

```
if (<boolean expression 1>)
        <statement 1>
else if (<boolean expression 2>)
        <statement 2>
else if (<boolean expression 3>)
        <statement 3>
else
        <statement 4>
```

Nested `if-else` statement – Syntax

There are many representations

```
if (<boolean expression 1>)    <statement 1>
else if (<boolean expression 2>)    <statment 2>
    else if (<boolean expression 3>)    < statement 3>
        else    <statement 4>
```

```
if (<boolean expression 1>)
    <statement 1>
else if (<boolean expression 2>)
        <statement 2>
    else if (<boolean expression 3>)
            <statement 3>
        else
            <statement 4>
```

Nested if-else statement - Example

Coding style:
Same tab space

```
#include <iostream>
using namespace std;
int main(){
    float score = 8.7f;

    if(score < 5.0f)
        cout << “F";
    else if(score < 6.5f)
        cout << “D";
    else if(score < 8.5f)
        cout << “C";
    else if(score < 9.5f)
        cout << “B";
    else
        cout << “A";

    return 0;
}
```

Nested `if-else` statement - Example

Coding style:
Same tab space
(when pairs of curly braces { } are used)

```
if(score < 5.0f){
    cout << “F";
}
else if(score < 6.5f){
    cout << “D";
}
else if(score < 8.5f){
    cout << “C";
}
else if(score < 9.5f){
    cout << “B";
}
else{
    cout << “A";
}
```

Switch-case statement – Application

- When a program has a list of tasks to be executed according to known cases (events).
    - The number of tasks: countable.
    - The number of events: countable.
- Example: book management.
    - The program provides an interface for the user to
        - Read data from file
        - Enter data to the program
        - Find a book
        - Retrieve a list of borrowers with overdue books
        - Etc.
    - The number of tasks listed is finite.

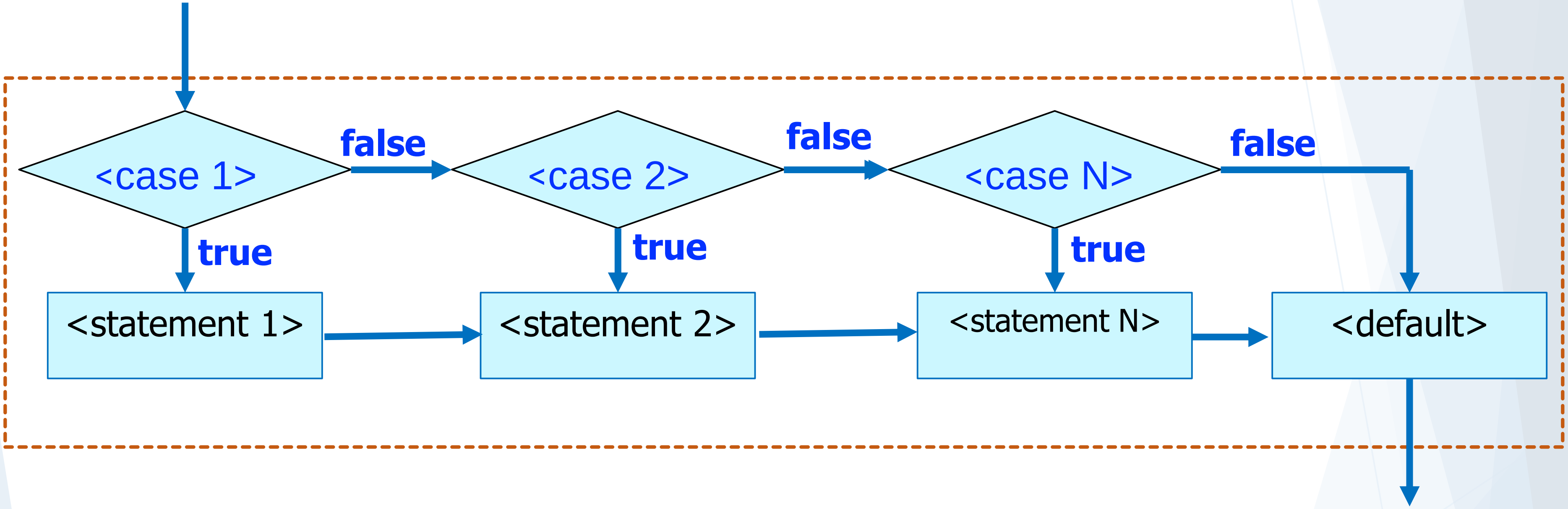
Switch-case statement – Application

- Example: book management.
  - The program can print a list of tasks from which the user choose:
    - Graphics interface: display graphical indicators, buttons, etc. instead of printing on a console.
  - When user chooses a menu:
    - The program performs the task corresponding to the chosen menu.

⇒ Each task is executed depending on a specific event.

⇒ Suitable for switch-case.

Switch-case statement – Concept



Switch-case statement – Concept

- How switch-case works:
  - The program checks which case is triggered among all listed cases: **<case 1>, <case 2>, .., <case N>**
  - If the i-th case is triggered (I = 1 .. N):
    - Execute every statements from i to N (**<statement i>** to **<statement N>**), including **<default>**
    - If the current statement is break; the program will step out of the **switch-case** scope structure and jump into the next statement right after **switch-case**.
  - If none of the case was triggered:
    - The program will execute **<default>** and step out of **switch-case**.

Switch-case statement – Syntax

```
switch (<eventID>){
case <ID 1>: <statement 1>
case <ID 2>: <statement 2>

case <ID N>: <statement N>
default: <default statement>
}
```

switch, case, default: keywords

<**eventID**>:

MUST BE an expression of the following types:

(1) integral or enumeration type, or

(2) a class type contextually implicitly convertible to an integral or enumeration type (through typedef)

<**ID i**>: (i=1,.., N), possible values of eventID

Switch-case statement – Syntax

```
switch (<eventID>){
case <ID 1>:
case <ID 2>:
case <ID 3>: <statement 3>

case <ID N>: <statement N>
default: <default statement>
}
```

If statement 1, 2 and 3 are similar, we can types as above so that <statement 3> can be used for all three cases 1, 2 and 3.

Switch-case statement – Syntax

```
switch (<eventID>){
case <ID 1>: <statement 1> break;
case <ID 2>: <statement 2>

case <ID N>: <statement N>
default: <default statement>
}
```

After <statement 1> is done, the program will step out of the above switch-case, the rest of the statements (2 to N) are skipped.

Switch-case statement – Syntax

```
switch (<eventID>){
case <ID 1>: <statement 1> break;
case <ID 2>: <statement 2> break;

case <ID N>: <statement N> break;
default: <default statement>
}
```

In this switch-case, only one of the statement will be executed.

Switch-case statement – Syntax

```
switch (<eventID>){
case <ID 1>: <statement 1> break;
case <ID 2>: <statement 2> break;

case <ID N>: <statement N> break;
}
```

In this switch-case, only one of the statement will be executed.
No default statement.

Switch-case statement – Example

- Problem:
  - The program receives a choice from user.
    - A choice can be either 1 or 2.
      - The numbers (1 and 2) are not tied to any specific event at the time.
  - Print the choice of the user.
    - In the future, instead of printing the choice, we can just replace the printing with the actual codes of a specific event tied to the corresponding ID number.

Switch-case statement – Example

```
#include <iostream>
using namespace std;
int main(){
    int choice;
    cout << “Enter the choice: \n“;
    cin >> choice;
    switch (choice){
    case 1:
        cout << "Case 1\n"; cout << "Task 1\n");
    case 2:
        cout << "Case 2\n"; cout << "Task 2\n");
    default:
        cout << “Default task\n";
    }
    return 0;
}
```

Switch-case statement – Example

The program might print bot tasks
Because there is no break;

```
#include <iostream>
using namespace std;
int main(){
    int choice;
    cout << “Enter the choice: \n“;
    cin >> choice;
    switch (choice){
    case 1:
        cout << "Case 1\n"; cout << "Task 1\n");
    case 2:
        cout << "Case 2\n"; cout << "Task 2\n");

    default:
        cout << “Default task\n";
    }
    return 0;
}
```

Switch-case statement – Example

break: makes it so that only
The chosen task will be printed

```
#include <iostream>
using namespace std;
int main(){
    int choice;
    cout << “Enter the choice: \n“;
    cin >> choice;
    switch (choice){
    case 1:
        cout << "Case 1\n"; cout << "Task 1\n"); break;
    case 2:
        cout << "Case 2\n"; cout << "Task 2\n"); break;

    default:
        cout << “Default task\n";
    }
    return 0;
}
```

Switch-case statement – Example

- Another example for menu:
  - Enumeration type (enum) can be used.

Enumeration and switch-case

```
#include <iostream>
#include namespace std;
enum choice {Agree, Disagree, Undecided};
int main(){
    enum Luachon luachon;
    cout << "Enter your choice \n";
    cout << "0. Agree \n";
    cout << "1. Disagree \n";
    cout << "2. Haven’t decided \n";
    cin >> choice;

    switch (choice){
    case Agree: cout << "You agreed\n"); break;
    case Disagree: cout << "You disagreed\n"; break;
    case Undecided: cout << "You haven’t decided\n"; break;
    default: cout << "Your choice was invalid\n";
    }
    return 0;
}
```

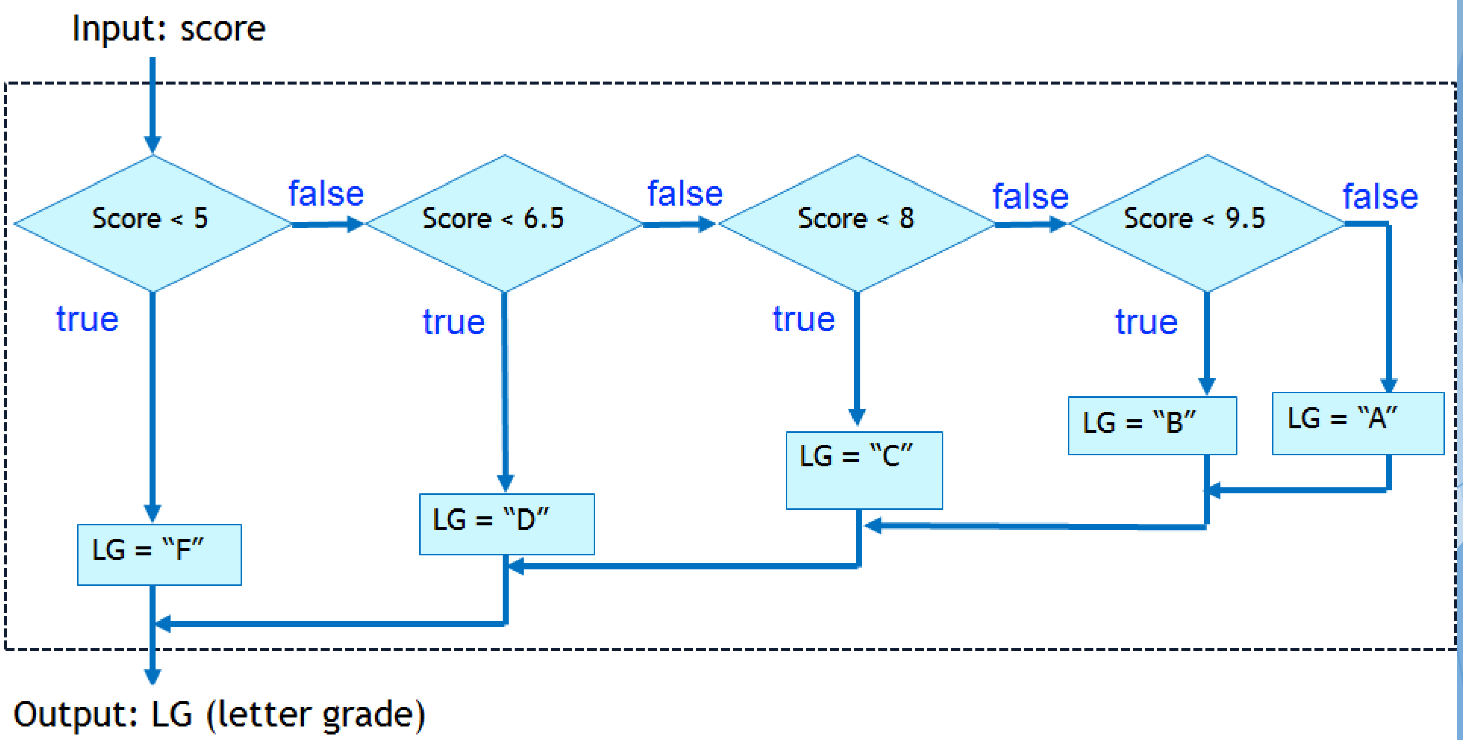
Switch-case statement – Exercise

- Problem: product management
  - The program has the following features:
    - Allow user to enter product
    - Save product
    - Read products from file
    - Etc.
  - The program must print a menu from which the user will choose what to do (choice is entered through a keyboard).
  - For each choice:
    - Print the chosen task
      - Students will have to implement all of these tasks further in the course

Comparing if-else and switch-case

- Why didn’t we use switch-case for the grading problem?

  => Because the scores are not `integral` or `enumeration` data type.



Comparing if-else and switch-case

- `Switch-case` statement can always be replaced as a sequence of `if-else` statements.
- For some situations, `switch-case` is more readable.
- All control structures can be represented by `if-else` and the statement `goto` (with the help of `labels`)

Conclusion

- Students should be able to decompose a problem into smaller ones to solve it:
  - Refer to the quadratic equation examples and the others.
- Understand and is able to apply the control statements provided in C++:
  - The logic behind conditional/branching execution.
  - `if-else` statement and nested `if-else` statements.
  - `switch-case` statement.