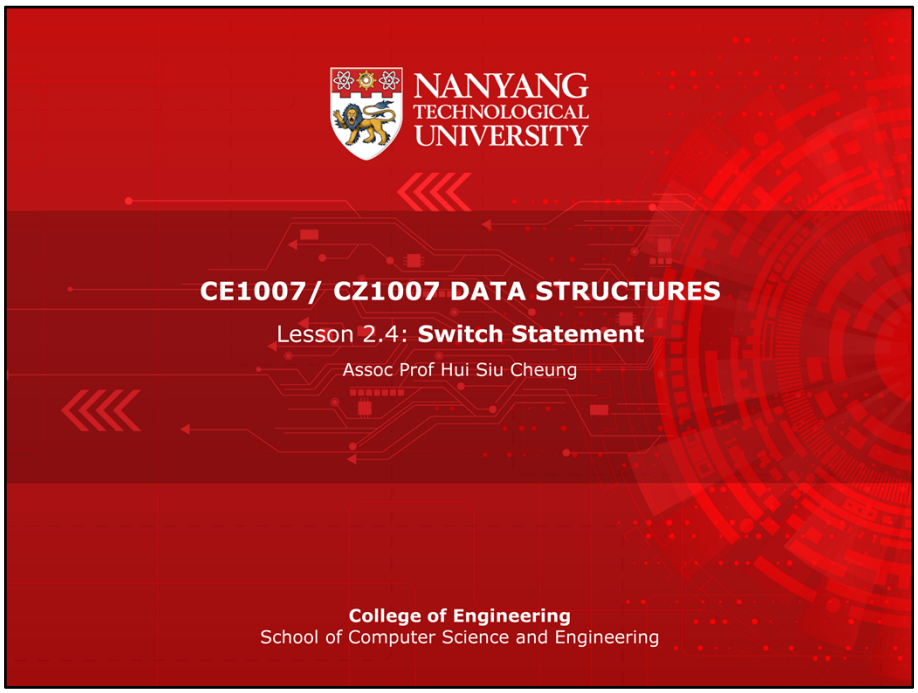
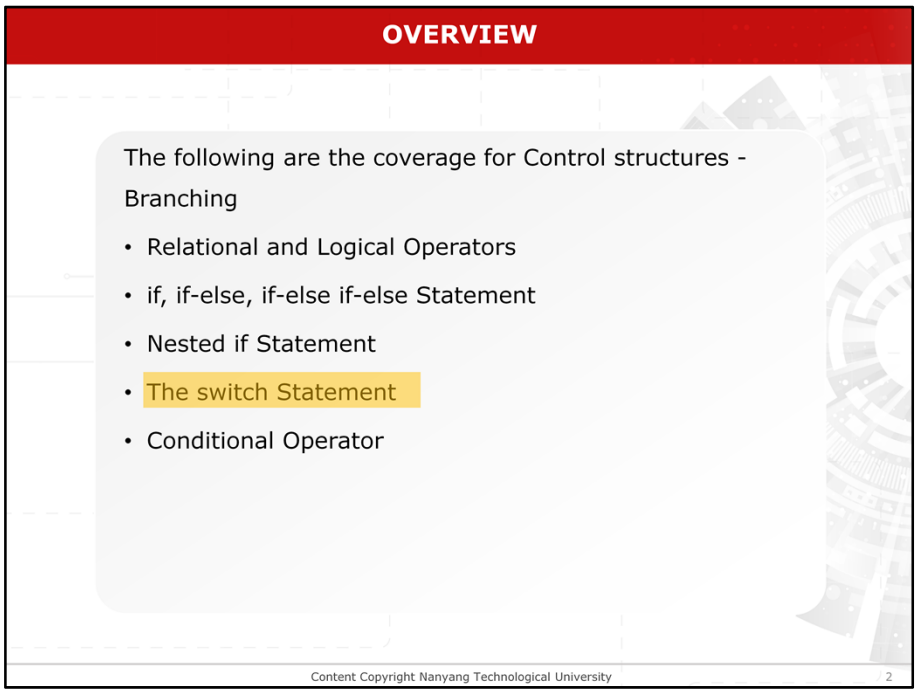
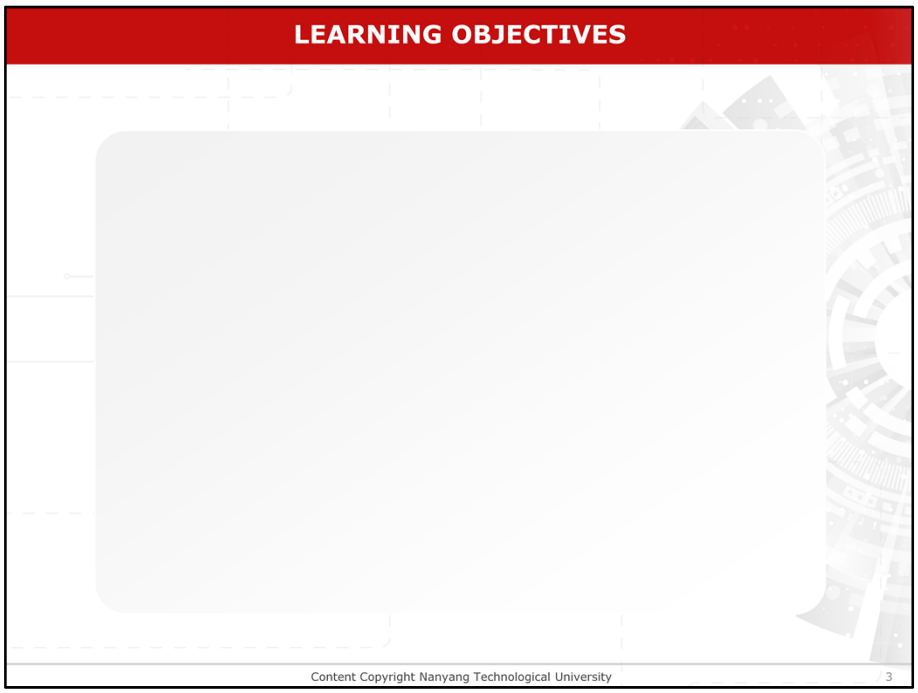


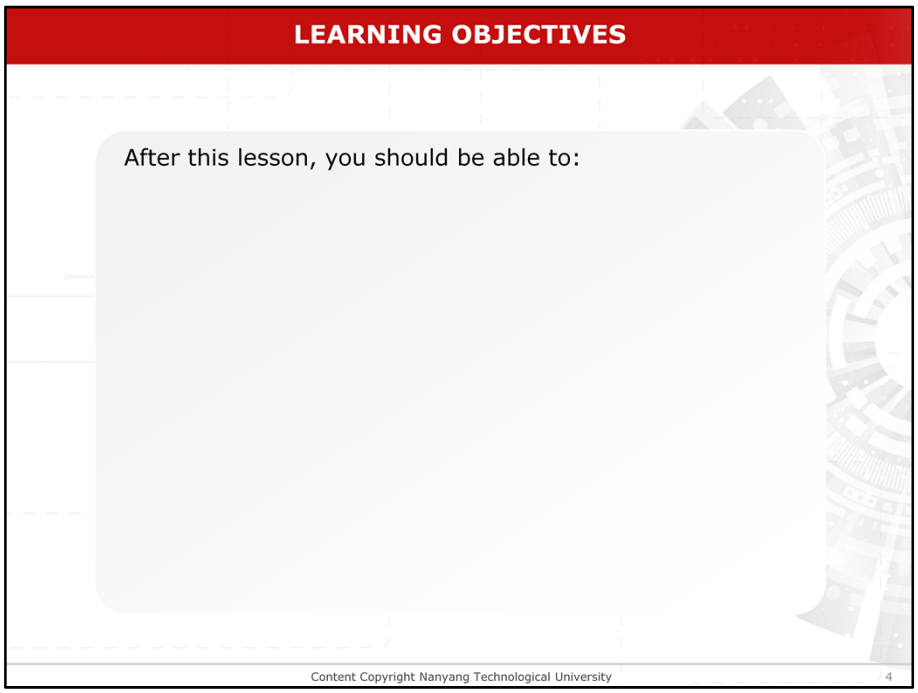
This lesson is on control structure branching



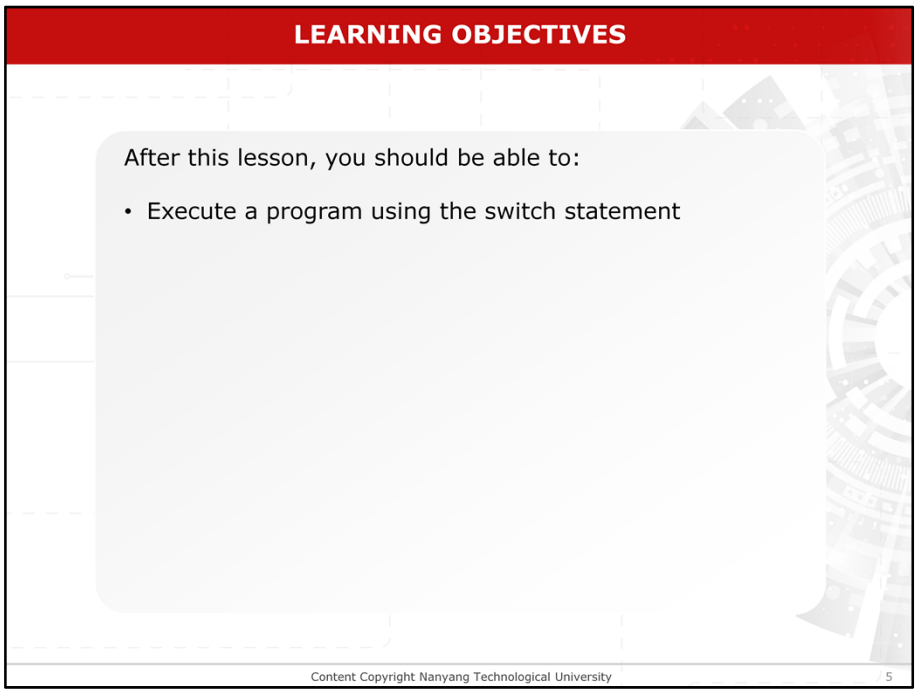
There are 5 main sections to cover for Control structures (branching). This video lesson focuses on the fourth part: The switch statements



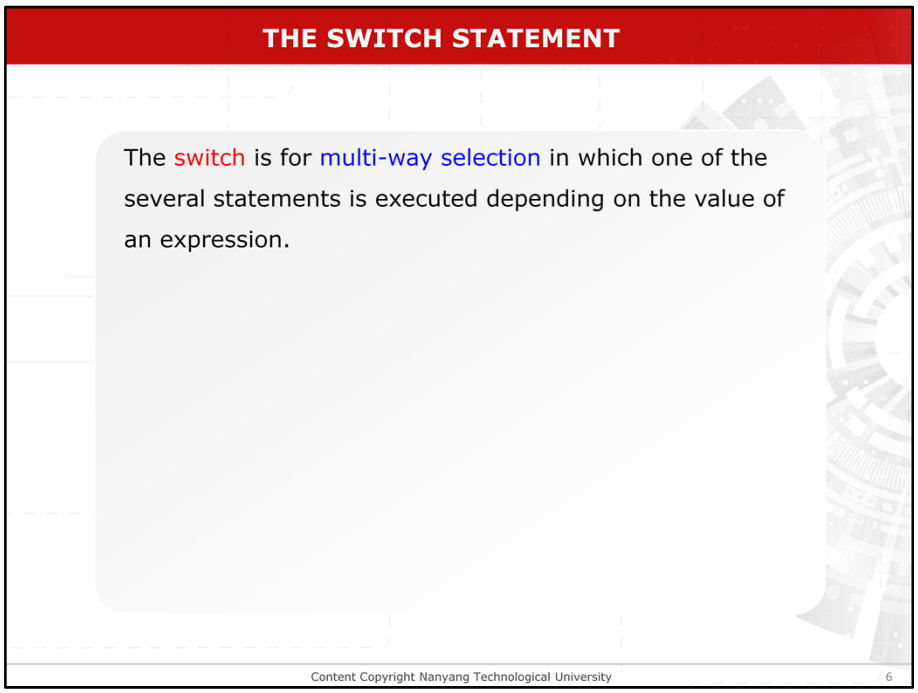
Learning objectives



After this lesson, you should be able to:

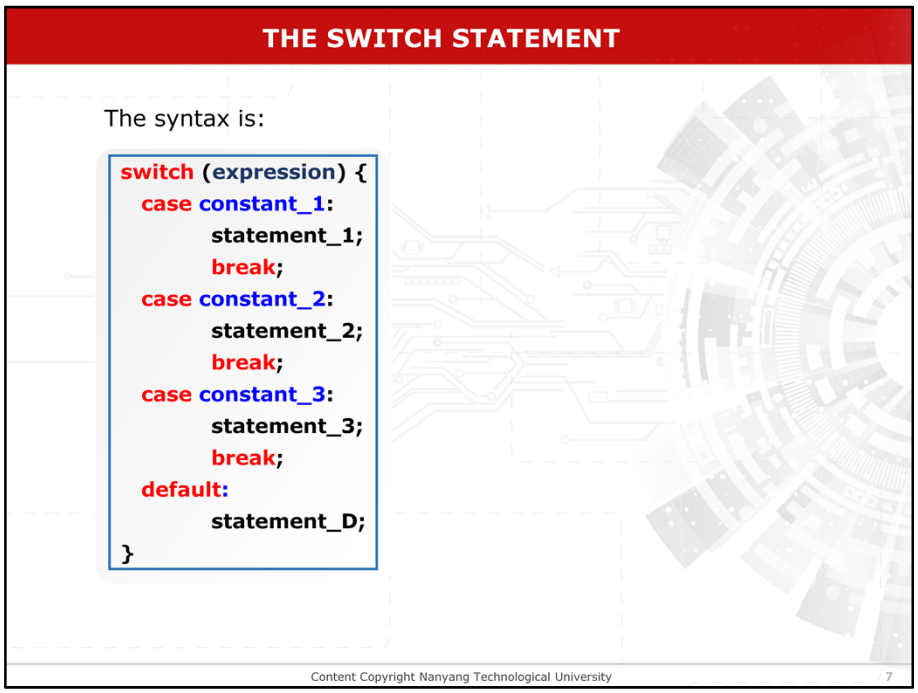


Execute a program using Switch statement



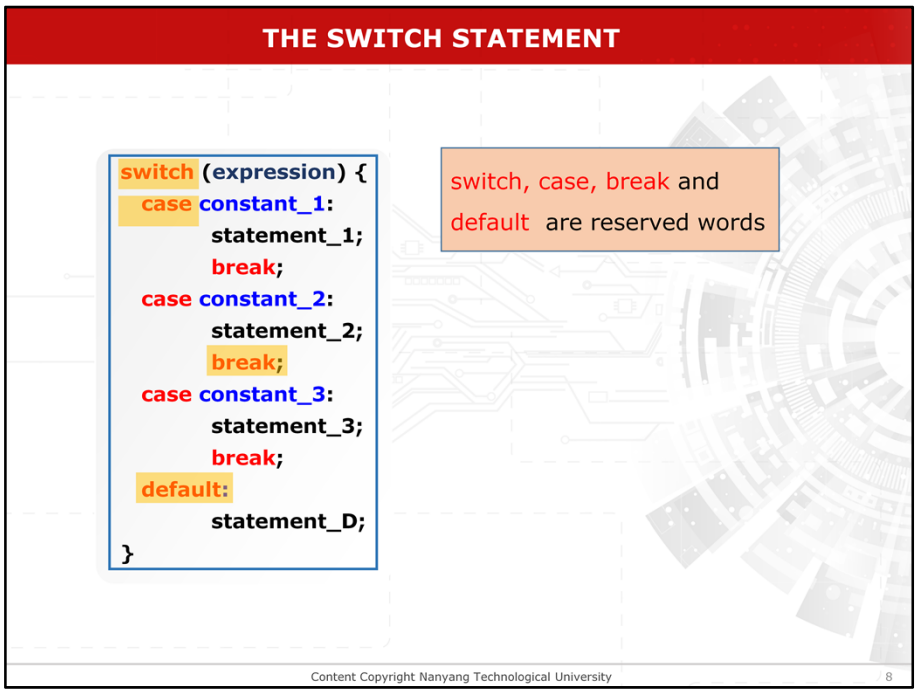
The switch Statement

The **switch** statement provides a multi-way decision structure in which one of the several statements is executed depending on the value of an expression.

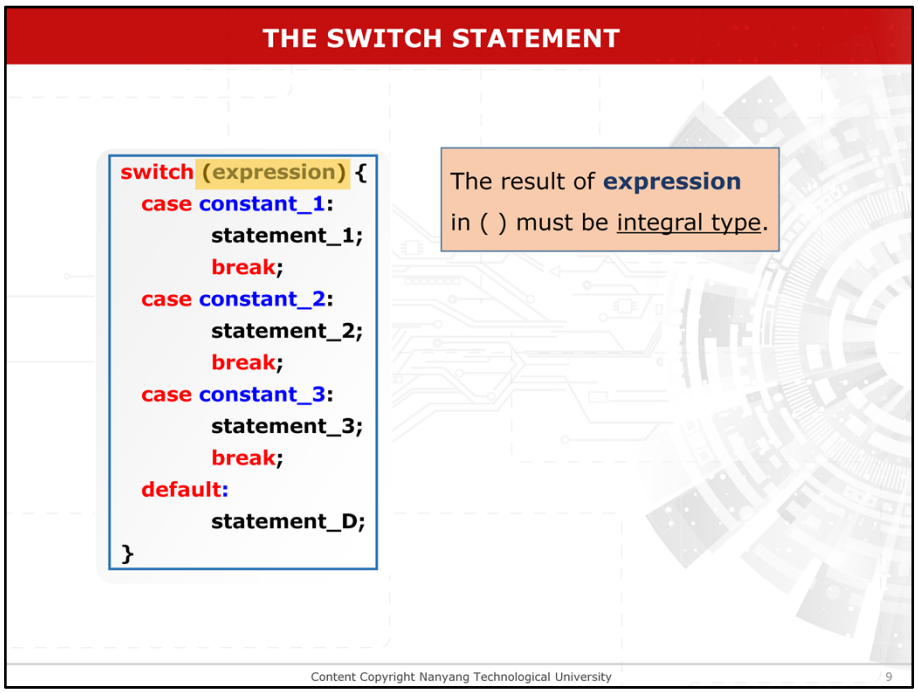


The switch Statement

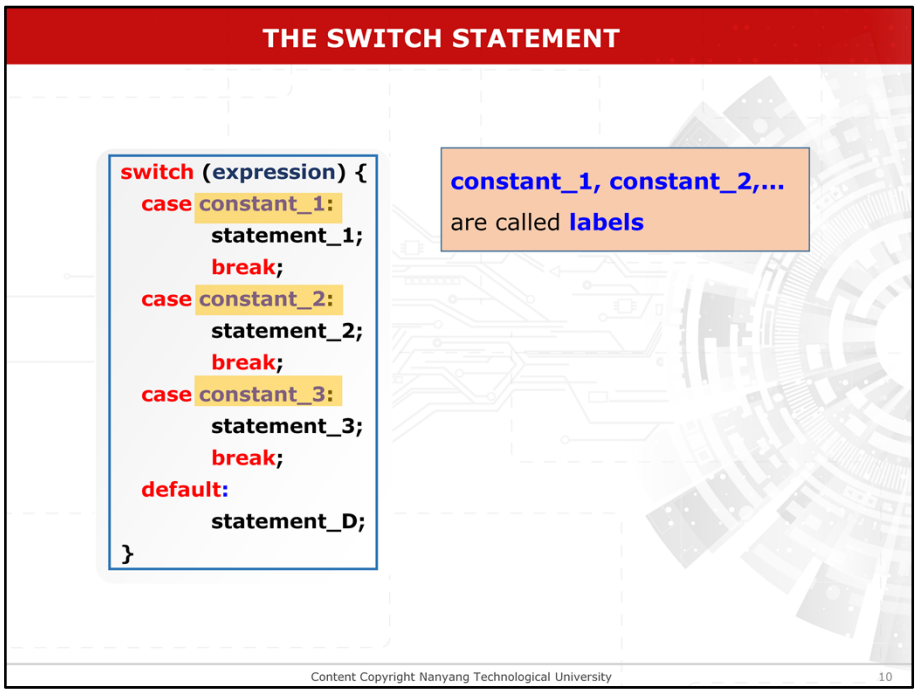
The syntax of a **switch** statement is shown here.



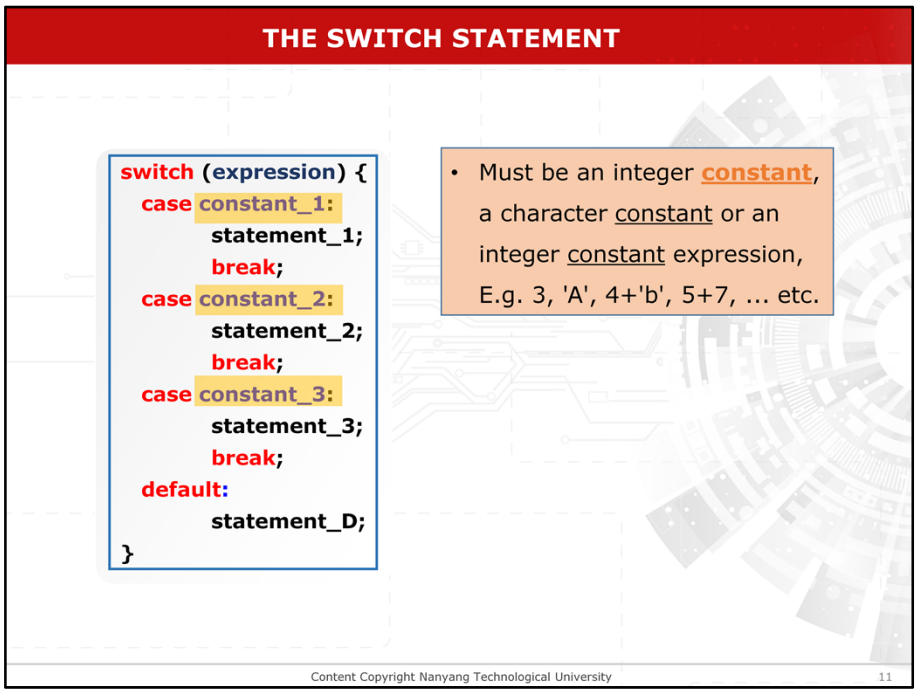
where **switch**, **case**, **break** and **default** are reserved keywords.



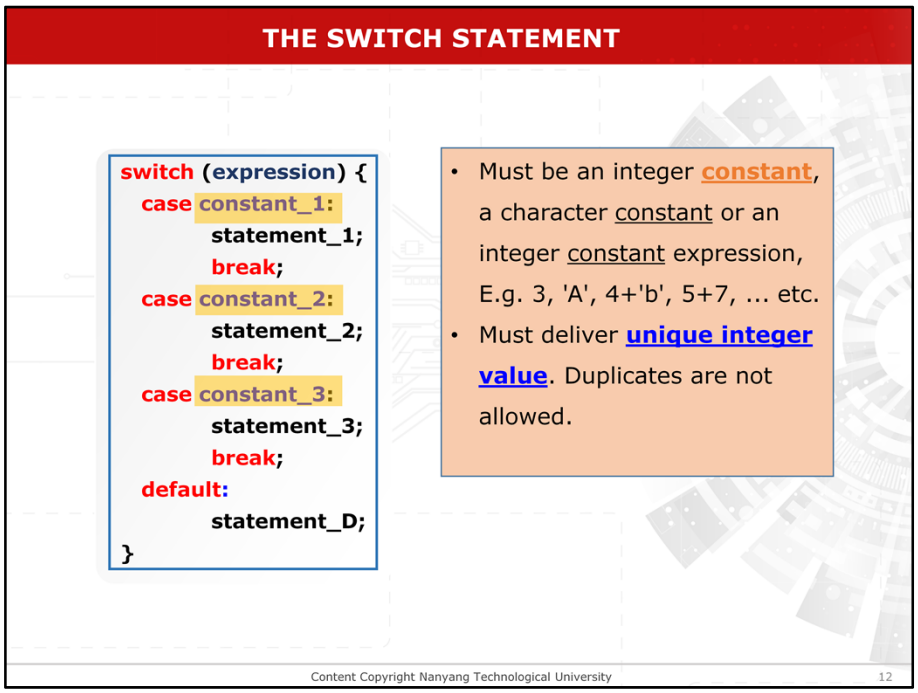
The result of ***expression*** in bracket must be **integral type**.



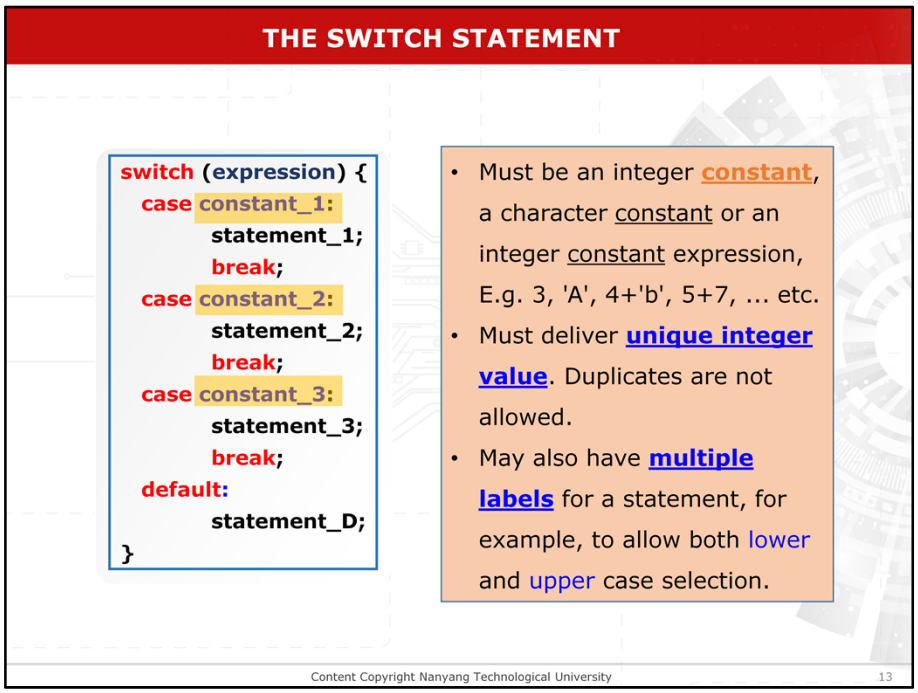
constant_1, **constant_2**, etc. are called *labels*.



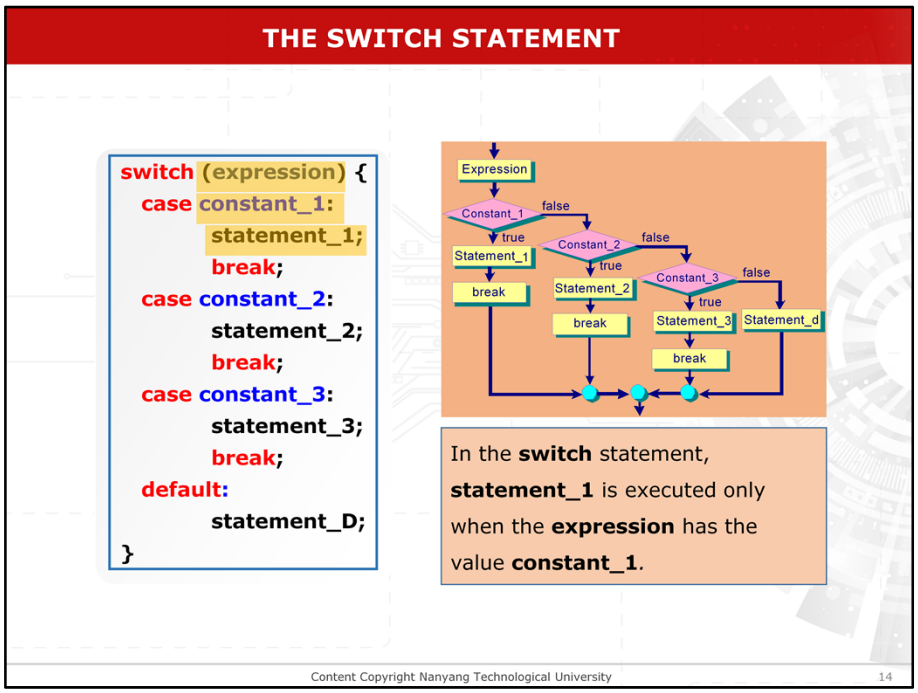
Each must be an integer constant, a character constant or an integer constant expression, e.g. 3, 'A', 4+'b', 5+7, etc.



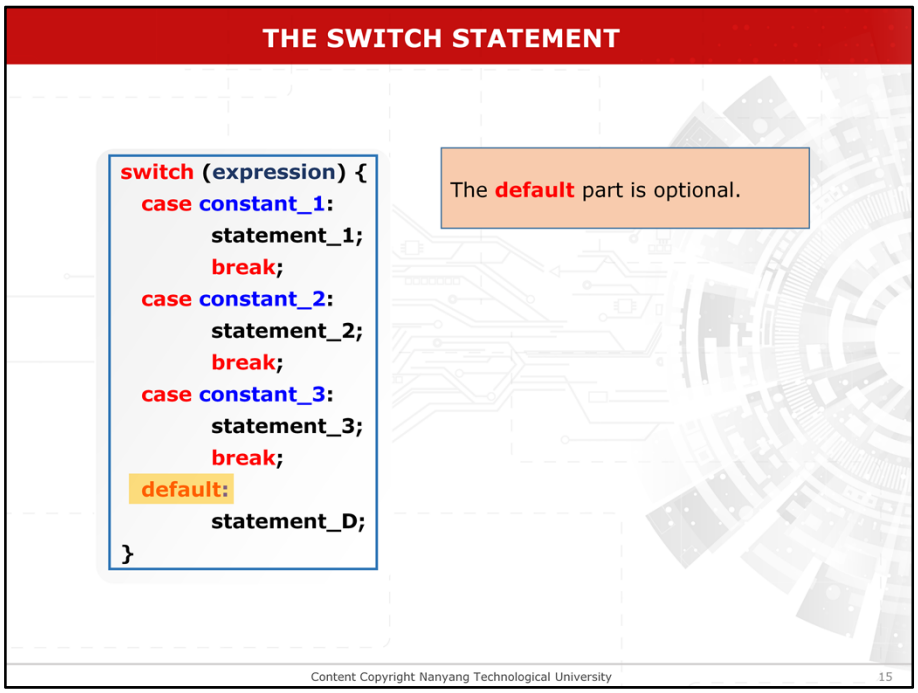
Each of the labels must deliver a unique integer value. Duplicates are not allowed.



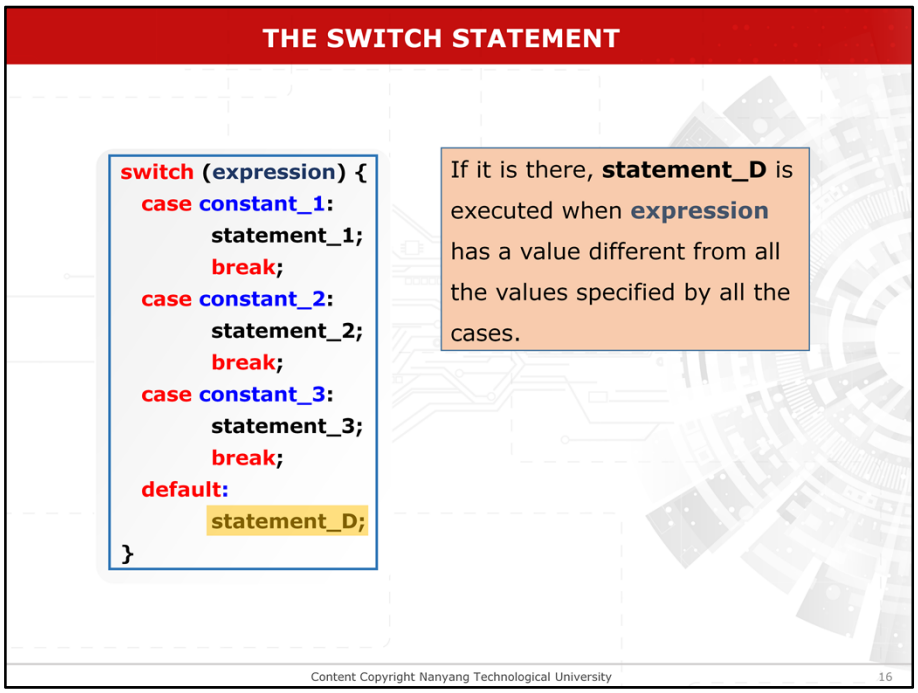
Each of the labels may also have **multiple labels** for a statement, for example, to allow both lower and upper case selection.



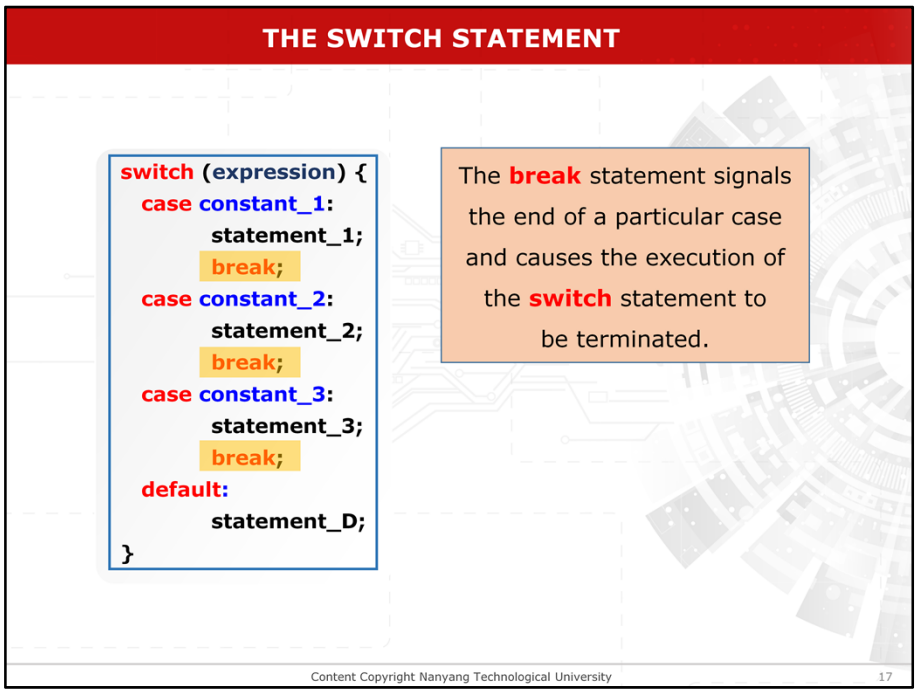
In the **switch** statement, **statement_1** is executed only when the **expression** has the value **constant_1.**



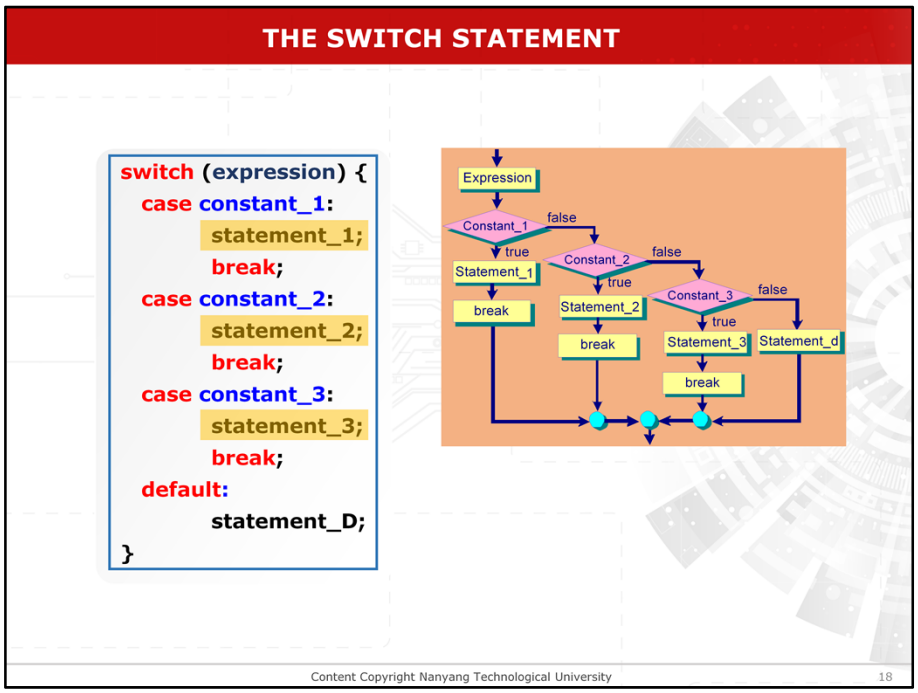
The **default** part is optional.



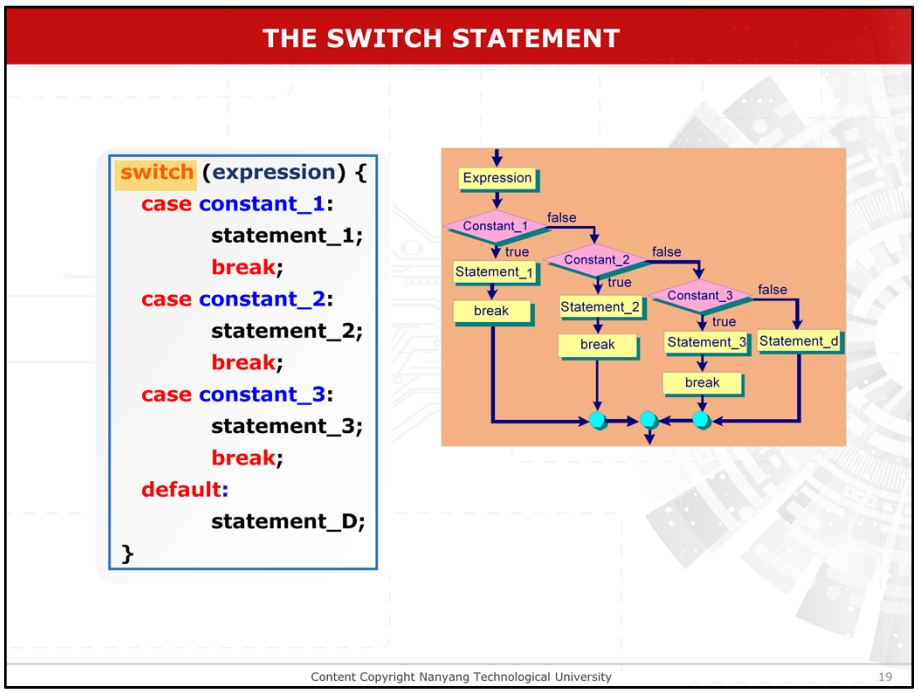
If it is there, **statement_D** is executed when **expression** has a value different from all the values specified by all the cases.



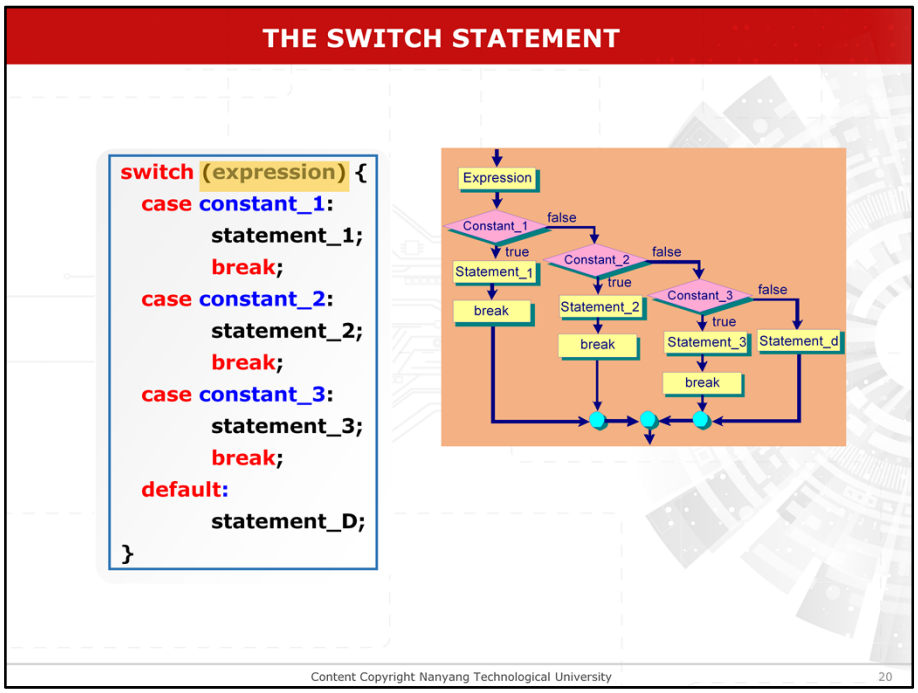
The **break** statement signals the end of a particular case and causes the execution of the **switch** statement to be terminated.



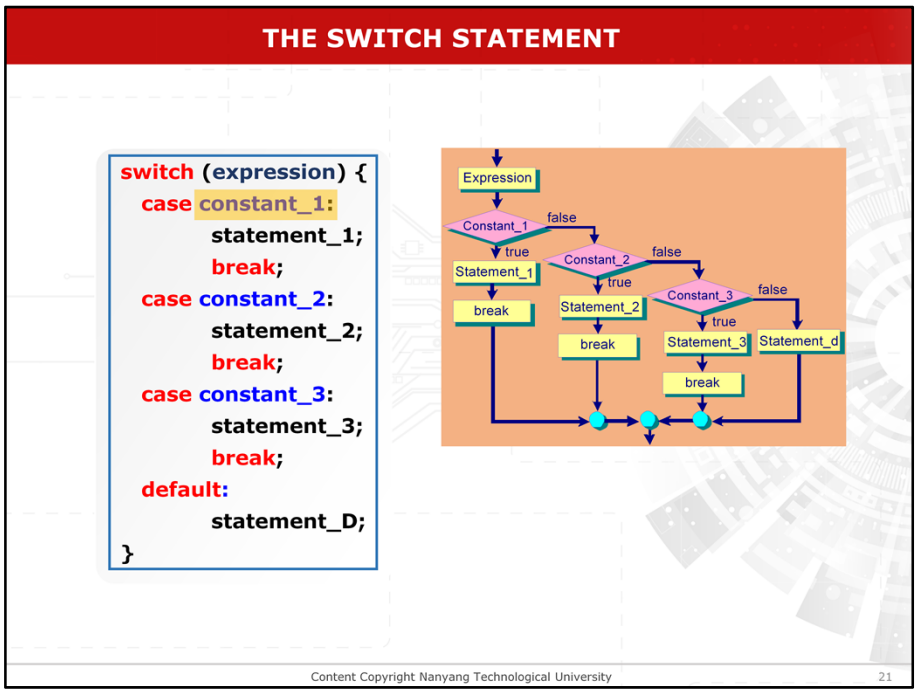
Each of the **statement_I** may be a single statement terminated by a semicolon or a compound statement enclosed by **braces{}.**



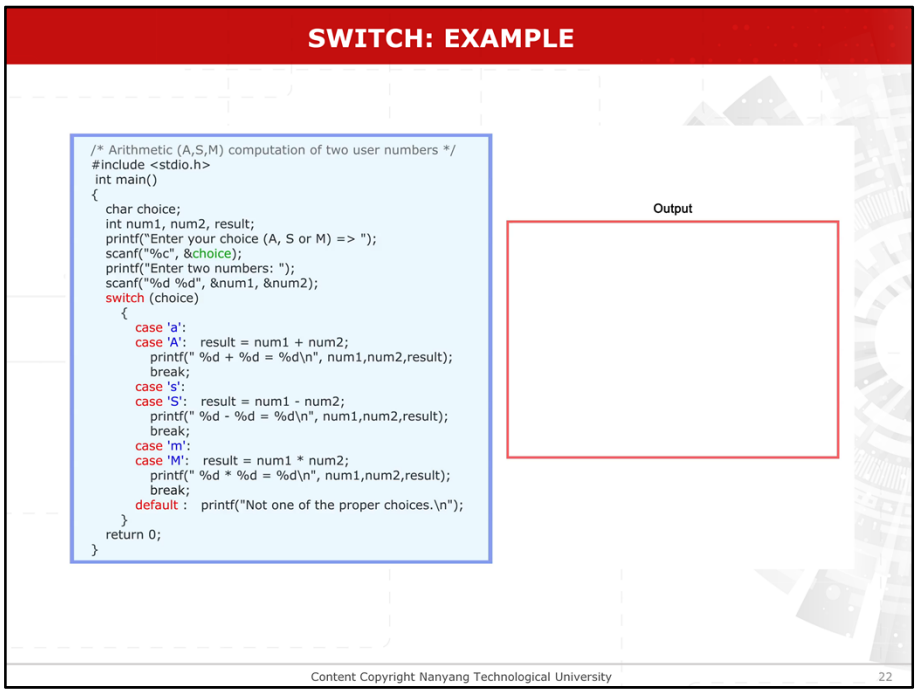
The **switch** statement is a better construct than multiple **if-else** statements. However, there are several restrictions in the use of the **switch** statement.



The **expression** of the **switch** statement must return a result of *integer* or *character* data type.

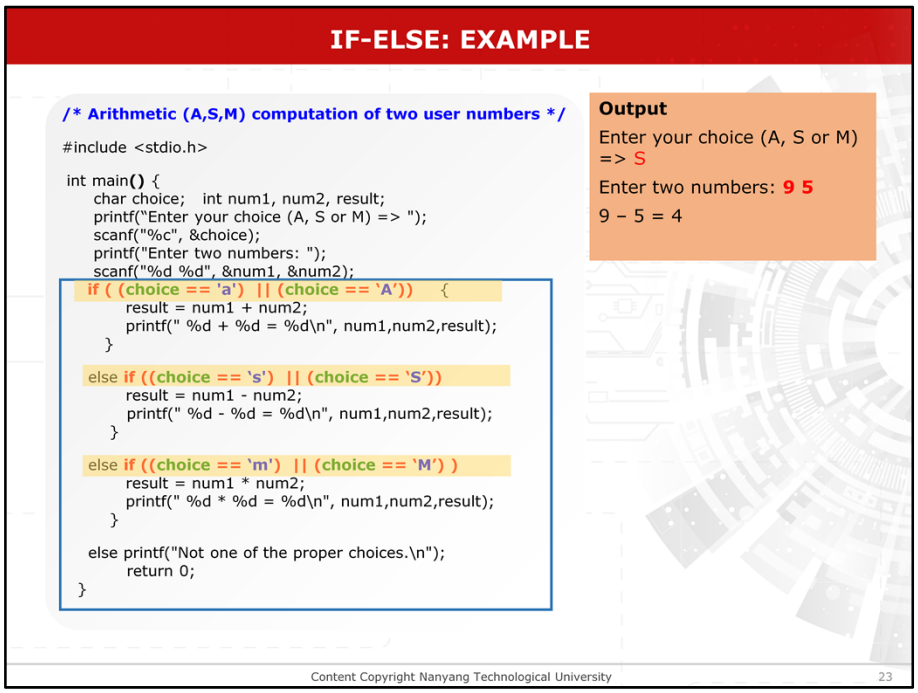


The value in the **constant_I** part must be an integer constant or character constant. Moreover, it does not support a range of values to be specified.



The switch Statement: Example

The **switch** statement is quite commonly used in menu-driven applications. In this program, it uses the **switch** statement for menu-driven selection.



The if-else-if-else statement: Example

The same program on supporting arithmetic operation can also be implemented using the if-else-if-else statements. Generally, the switch statements can be replaced by if-else-if-else statements

IF-ELSE: EXAMPLE

```
/* Arithmetic (A,S,M) computation of two user numbers */

#include <stdio.h>

int main() {
    char choice;   int num1, num2, result;
    printf("Enter your choice (A, S or M) => ");
    scanf("%c", &choice);
    printf("Enter two numbers: ");
    scanf("%d %d", &num1, &num2);
    if ( (choice == 'a') || (choice == 'A'))    {
        result = num1 + num2;
        printf(" %d + %d = %d\n", num1,num2,result);
      }

    else if ((choice == 's') || (choice == 'S'))
        result = num1 - num2;
        printf(" %d - %d = %d\n", num1,num2,result);
      }

    else if ((choice == 'm') || (choice == 'M') )
        result = num1 * num2;
        printf(" %d * %d = %d\n", num1,num2,result);
      }

    else printf("Not one of the proper choices.\n");
        return 0;
}
```

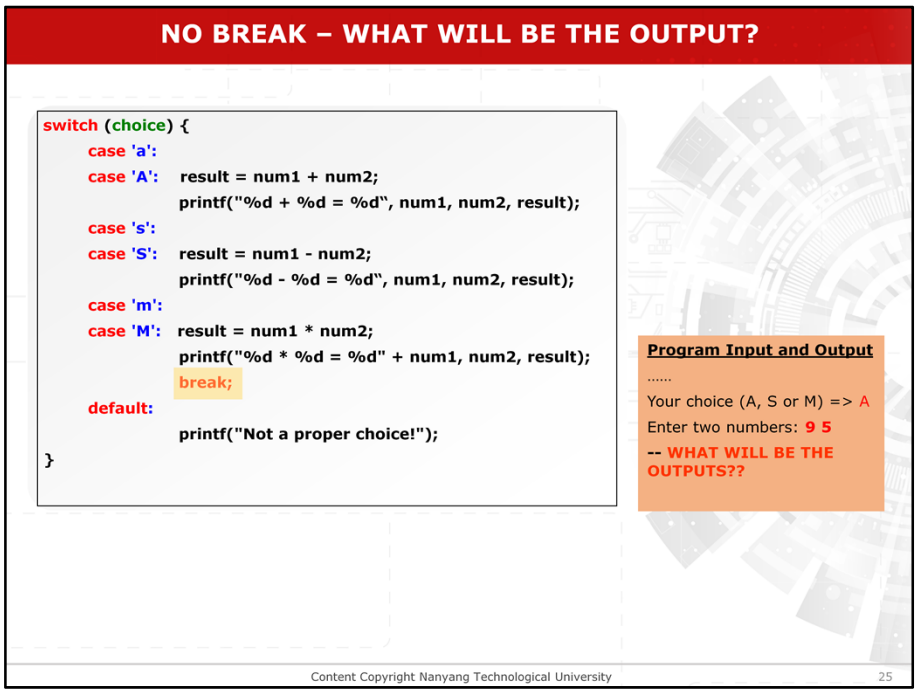
Output

Enter your choice (A, S or M) => S

Enter two numbers: **9 5**

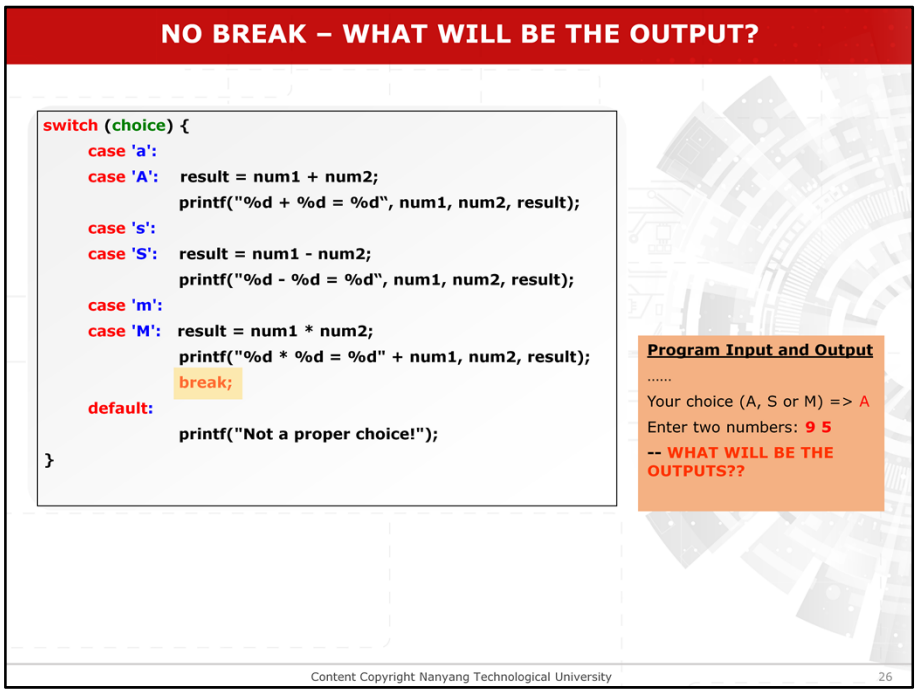
9 – 5 = 4

However, as labels in the switch construct must be constant values (i.e. integer, character or expression), we will not be able to convert certain if-else statements into switch statements.

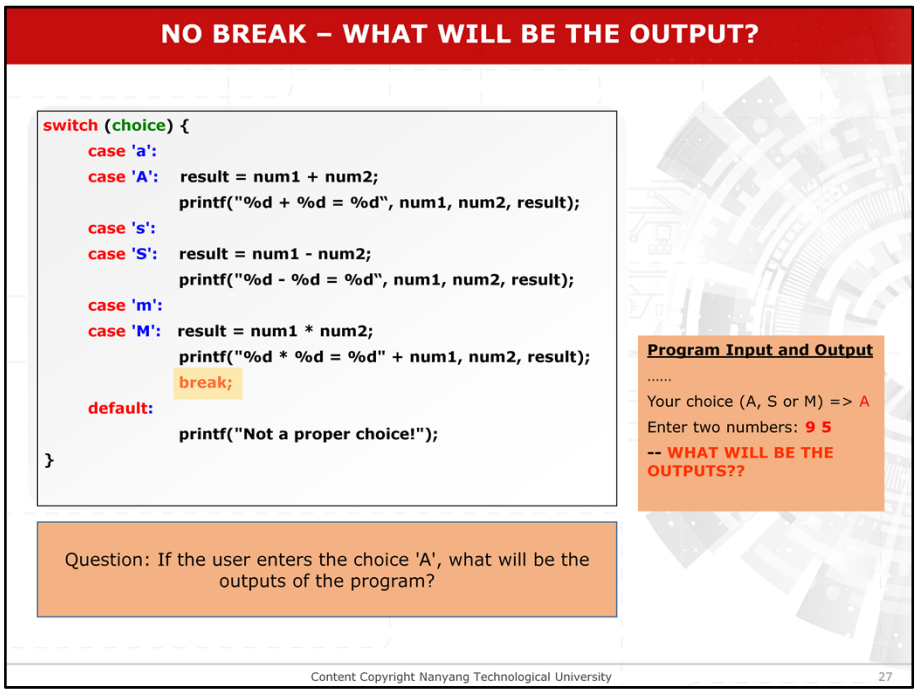


The switch Statement – Omitting break

In the **switch** statement, the **break** statement is placed at the end of each **case**. What will happen if we omit the **break** statement?



For example, if the **switch** statement is modified as follows:
In this example, the **break** statement is omitted for the cases on addition and subtraction.



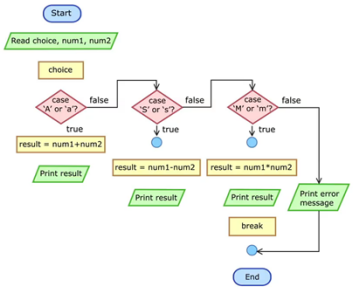
If the user enters the choice 'A', what will be the outputs of the program?

NO BREAK - FALL THROUGH

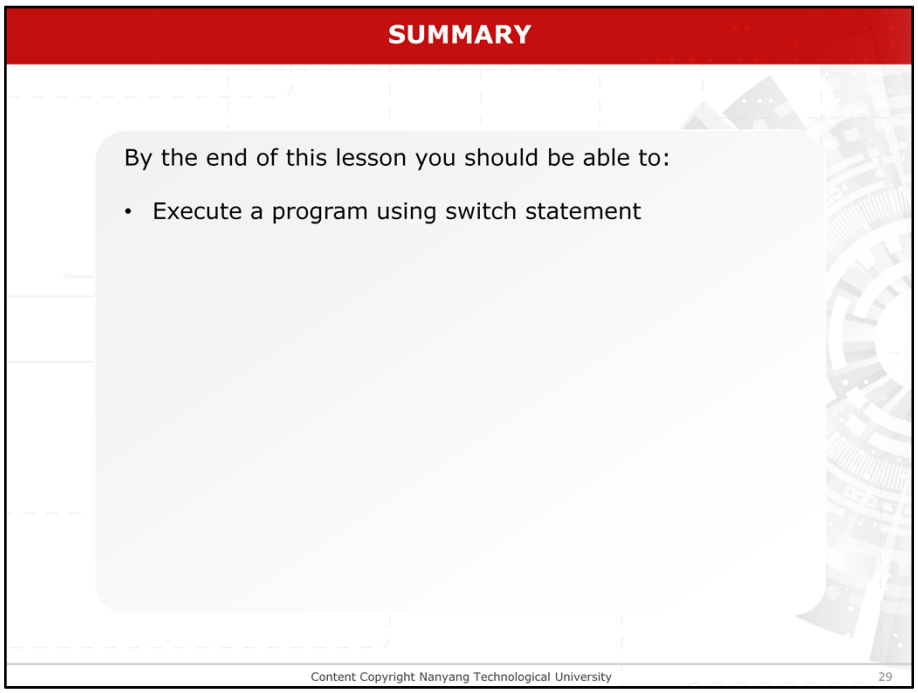
```
switch (choice) {

        case 'a':
        case 'A':   result = num1 + num2;
          printf("%d + %d = %d", num1, num2, result);
        case 's':
        case 'S':   result = num1 - num2;
          printf("%d - %d = %d", num1, num2, result);
        case 'm':
        case 'M':  result = num1 * num2;
          printf("%d * %d = %d", num1, num2, result);
          break;
        default:
          printf("Not a proper choice!");

  }
```



The **break** statement is used to end each **case** constant block statement. If we do not put the **break** statement in the **switch** statement, execution will continue with the statements for the subsequent **case** labels until a **break** statement or the end of the **switch** statement is reached. Therefore, if the break statement is omitted, the input and output from the program will become like the one shown. Therefore, if the break statement is omitted, the input and output from the program will become like the one shown.



By the end of the lesson, you should be able to do the listed.