

This lesson is on control structure branching

OVERVIEW

The following are the coverage for Control structures -

Branching

- Relational and Logical Operators
- if, if-else, if-else if-else Statement
- Nested if Statement
- **The switch Statement**
- Conditional Operator

Content Copyright Nanyang Technological University

2

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



Learning objectives

LEARNING OBJECTIVES

After this lesson, you should be able to:

Content Copyright Nanyang Technological University 4

After this lesson, you should be able to:

LEARNING OBJECTIVES

After this lesson, you should be able to:

- Execute a program using the switch statement

Content Copyright Nanyang Technological University 5

Execute a program using Switch statement

THE SWITCH STATEMENT

The **switch** is for **multi-way selection** in which one of the several statements is executed depending on the value of an expression.

Content Copyright Nanyang Technological University

6

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 is:

```
switch (expression) {  
    case constant_1:  
        statement_1;  
        break;  
    case constant_2:  
        statement_2;  
        break;  
    case constant_3:  
        statement_3;  
        break;  
    default:  
        statement_D;  
}
```

Content Copyright Nanyang Technological University 7



The switch Statement

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

THE SWITCH STATEMENT

switch (expression) {
 case constant_1:
 statement_1;
 break;
 case constant_2:
 statement_2;
 break;
 case constant_3:
 statement_3;
 break;
 default:
 statement_D;
}

switch, case, break and default are reserved words

Content Copyright Nanyang Technological University 8

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

THE SWITCH STATEMENT

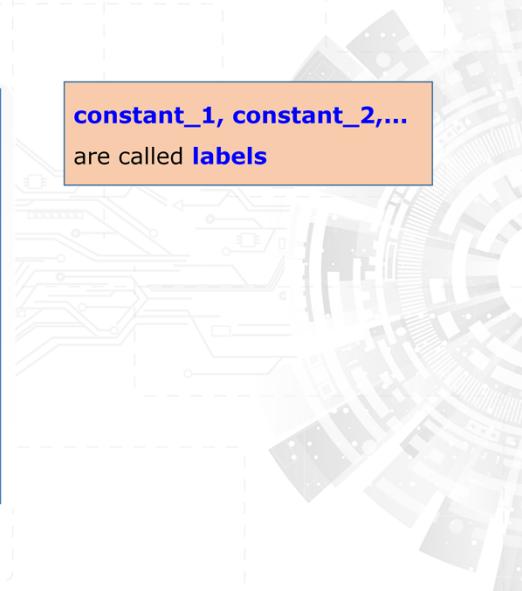
```
switch(expression) {  
    case constant_1:  
        statement_1;  
        break;  
    case constant_2:  
        statement_2;  
        break;  
    case constant_3:  
        statement_3;  
        break;  
    default:  
        statement_D;  
}
```

The result of **expression** in () must be integral type.

Content Copyright Nanyang Technological University 9

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

THE SWITCH STATEMENT



```
switch (expression) {  
    case constant_1:  
        statement_1;  
        break;  
    case constant_2:  
        statement_2;  
        break;  
    case constant_3:  
        statement_3;  
        break;  
    default:  
        statement_D;  
}
```

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

Content Copyright Nanyang Technological University 10

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

THE SWITCH STATEMENT

```
switch (expression) {  
    case constant_1:  
        statement_1;  
        break;  
    case constant_2:  
        statement_2;  
        break;  
    case constant_3:  
        statement_3;  
        break;  
    default:  
        statement_D;  
}
```

- Must be an integer constant, a character constant or an integer constant expression, E.g. 3, 'A', 4+'b', 5+7, ... etc.

Content Copyright Nanyang Technological University 11

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

THE SWITCH STATEMENT

```
switch (expression) {  
    case constant_1:  
        statement_1;  
        break;  
    case constant_2:  
        statement_2;  
        break;  
    case constant_3:  
        statement_3;  
        break;  
    default:  
        statement_D;  
}
```

- Must be an integer constant, a character constant or an integer constant expression, E.g. 3, 'A', 4+'b', 5+7, ... etc.
- Must deliver unique integer value. Duplicates are not allowed.

Content Copyright Nanyang Technological University 12

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

THE SWITCH STATEMENT

```
switch (expression) {  
    case constant_1:  
        statement_1;  
        break;  
    case constant_2:  
        statement_2;  
        break;  
    case constant_3:  
        statement_3;  
        break;  
    default:  
        statement_D;  
}
```

- Must be an integer constant, a character constant or an integer constant expression, E.g. 3, 'A', 4+'b', 5+7, ... etc.
- Must deliver unique integer value. Duplicates are not allowed.
- May also have multiple labels for a statement, for example, to allow both lower and upper case selection.

Content Copyright Nanyang Technological University 13

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

THE SWITCH STATEMENT

```
switch (expression) {
    case constant_1:
        statement_1;
        break;
    case constant_2:
        statement_2;
        break;
    case constant_3:
        statement_3;
        break;
    default:
        statement_D;
}
```

```

graph TD
    Expression --> C1{Constant_1}
    Expression --> C2{Constant_2}
    Expression --> C3{Constant_3}
    
    C1 -- true --> S1[Statement_1]
    C1 -- false --> End1(( ))
    S1 --> Break1[break]
    Break1 --> End1
    
    C2 -- true --> S2[Statement_2]
    C2 -- false --> End2(( ))
    S2 --> Break2[break]
    Break2 --> End2
    
    C3 -- true --> S3[Statement_3]
    C3 -- false --> End3(( ))
    S3 --> Break3[break]
    Break3 --> End3
    
    S1 --> Break1
    S2 --> Break2
    S3 --> Break3
    
    S1 --> S2
    S2 --> S3
    S3 --> S4[Statement_D]
    S4 --> Break4[break]
    Break4 --> End4(( ))
  
```

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

Content Copyright Nanyang Technological University

14

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

THE SWITCH STATEMENT

```
switch (expression) {  
    case constant_1:  
        statement_1;  
        break;  
    case constant_2:  
        statement_2;  
        break;  
    case constant_3:  
        statement_3;  
        break;  
    default:  
        statement_D;  
}
```

The **default** part is optional.

Content Copyright Nanyang Technological University 15

The **default** part is optional.

THE SWITCH STATEMENT

```
switch (expression) {  
    case constant_1:  
        statement_1;  
        break;  
    case constant_2:  
        statement_2;  
        break;  
    case constant_3:  
        statement_3;  
        break;  
    default:  
        statement_D;  
}
```

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

Content Copyright Nanyang Technological University 16

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

THE SWITCH STATEMENT

```
switch (expression) {  
    case constant_1:  
        statement_1;  
        break;  
    case constant_2:  
        statement_2;  
        break;  
    case constant_3:  
        statement_3;  
        break;  
    default:  
        statement_D;  
}
```

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

Content Copyright Nanyang Technological University 17

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

THE SWITCH STATEMENT

```
switch (expression) {
    case constant_1:
        statement_1;
        break;
    case constant_2:
        statement_2;
        break;
    case constant_3:
        statement_3;
        break;
    default:
        statement_D;
}
```

```

graph TD
    Expression --> C1{Constant_1}
    Expression --> C2{Constant_2}
    Expression --> C3{Constant_3}
    Expression --> Default

    C1 -- true --> S1[Statement_1]
    C1 -- false --> C2
    S1 -- break --> Join(( ))
    C2 -- true --> S2[Statement_2]
    C2 -- false --> C3
    S2 -- break --> Join
    C3 -- true --> S3[Statement_3]
    C3 -- false --> Default
    S3 -- break --> Join
    Default --> Join

    Join --> Statement_D[Statement_D]
    Statement_D -- break --> Join
  
```

Content Copyright Nanyang Technological University

18

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

THE SWITCH STATEMENT

```
switch (expression) {  
    case constant_1:  
        statement_1;  
        break;  
    case constant_2:  
        statement_2;  
        break;  
    case constant_3:  
        statement_3;  
        break;  
    default:  
        statement_D;  
}
```

The flowchart illustrates the execution of a switch statement. It starts with an **Expression** node at the top. An arrow labeled **true** points from the Expression node to a **Constant_1** node. From the Constant_1 node, an arrow labeled **true** leads to a **Statement_1** node, which then leads to a **break** node. If the path through Constant_1 leads to a **false** outcome, it proceeds to the Constant_2 node. This pattern continues for Constant_3 and Statement_3. Finally, if all constants lead to **false**, the flow reaches the **Statement_D** node, which also leads to a **break**. All **break** nodes have arrows pointing to a common exit point at the bottom.

Content Copyright Nanyang Technological University

19

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 SWITCH STATEMENT

```
switch (expression) {
    case constant_1:
        statement_1;
        break;
    case constant_2:
        statement_2;
        break;
    case constant_3:
        statement_3;
        break;
    default:
        statement_D;
}
```

```

graph TD
    Expression --> C1{Constant_1}
    Expression --> C2{Constant_2}
    Expression --> C3{Constant_3}
    
    C1 -- true --> S1[Statement_1]
    C1 -- false --> C2
    
    S1 --> Break1[break]
    Break1 --> Join1(( ))
    
    C2 -- true --> S2[Statement_2]
    C2 -- false --> C3
    
    S2 --> Break2[break]
    Break2 --> Join2(( ))
    
    C3 -- true --> S3[Statement_3]
    C3 -- false --> SD[Statement_D]
    
    S3 --> Break3[break]
    SD --> Break4[break]
    
    Break3 --> Join3(( ))
    Break4 --> Join3
    
    Join3 --> Exit(( ))
  
```

Content Copyright Nanyang Technological University

20

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

THE SWITCH STATEMENT

```
switch (expression) {  
    case constant_1:  
        statement_1;  
        break;  
    case constant_2:  
        statement_2;  
        break;  
    case constant_3:  
        statement_3;  
        break;  
    default:  
        statement_D;  
}
```

```
graph TD  
    Expression --> C1  
    C1 -- true --> S1  
    C1 -- false --> C2  
    C2 -- true --> S2  
    C2 -- false --> C3  
    C3 -- true --> S3  
    C3 -- false --> Sd  
    S1 -- break --> Join1  
    S2 -- break --> Join1  
    S3 -- break --> Join1  
    Sd -- break --> Join1  
    Join1 --> End
```

The flowchart illustrates the execution of a switch statement. It starts with an expression being evaluated. If the result is true, it executes Statement_1 and then reaches a break point. If the result is false, it moves to the next case (Constant_2). This process repeats for each case (Statement_2, Statement_3, etc.) and the default branch (Statement_D). Each branch ends with a break statement, which leads to a common exit point.

Content Copyright Nanyang Technological University

21

The value in the **constant_1** part must be an integer constant or character constant. Moreover, it does not support a range of values to be specified.

SWITCH: 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);
    switch (choice)
    {
        case 'a':
        case 'A': result = num1 + num2;
                    printf(" %d + %d = %d\n", num1,num2,result);
                    break;
        case 's':
        case 'S': result = num1 - num2;
                    printf(" %d - %d = %d\n", num1,num2,result);
                    break;
        case 'm':
        case 'M': result = num1 * num2;
                    printf(" %d * %d = %d\n", num1,num2,result);
                    break;
        default:   printf("Not one of the proper choices.\n");
    }
    return 0;
}

```

Content Copyright Nanyang Technological University 22

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.

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

Content Copyright Nanyang Technological University

23

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

Content Copyright Nanyang Technological University

24

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.

NO BREAK – WHAT WILL BE THE OUTPUT?

```

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!");
}

```

Program Input and Output

.....

Your choice (A, S or M) => A

Enter two numbers: 9 5

-- WHAT WILL BE THE
OUTPUTS??

Content Copyright Nanyang Technological University

25

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?

NO BREAK – WHAT WILL BE THE OUTPUT?

```

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!");
}

```

Program Input and Output

.....

Your choice (A, S or M) => A

Enter two numbers: 9 5

-- WHAT WILL BE THE
OUTPUTS??

Content Copyright Nanyang Technological University

26

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.

NO BREAK – WHAT WILL BE THE OUTPUT?

```
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!");  
}
```

.....
Your choice (A, S or M) => A
Enter two numbers: 9 5
**-- WHAT WILL BE THE
OUTPUTS??**

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

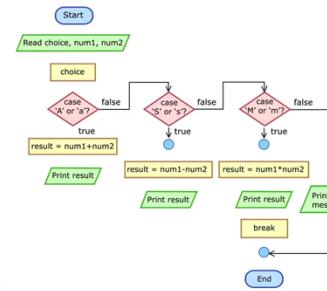
Content Copyright Nanyang Technological University

27

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!");  
}
```



Content Copyright Nanyang Technological University

28

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.

SUMMARY

By the end of this lesson you should be able to:

- Execute a program using switch statement

Content Copyright Nanyang Technological University

29

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