

Week 5

Session 5: Control structures using switch statements

Element 3: Apply control structures using 'if' and 'switch' statements

ECT 124: Writing Programs using C++



Performance criteria (PC) for E3



PC1: Write applications using 'if' control structures.

PC2: Write applications with SWITCH/CASE control structures.

In this lesson!



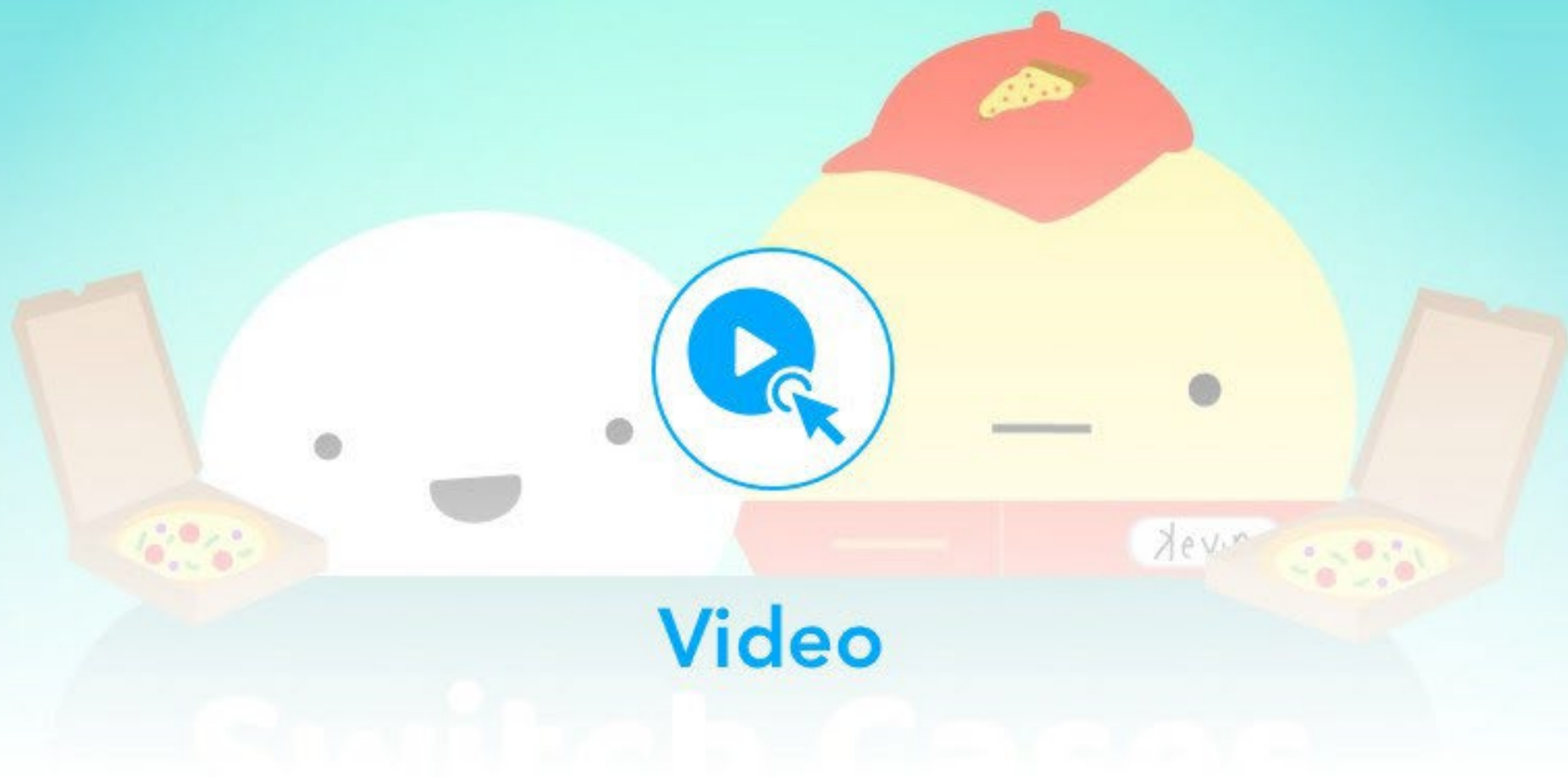
Learning objectives:

By the end of this lesson, the student should be able to:

- ✓ Write applications in C++ using SWITCH/CASE control structures for discrete events.



Class Activity 1



Coding Basics: Switch Statements | Programming for Beginners



Switch statements

- The C++ program has a built-in multiple-branch selection statement, called switch, which successively tests the value of an expression against a list of integer or character constants.
- When a match is found, the statements associated with that constant are executed. The syntax of switch statement is:
- The expression must evaluate to a **character** or **integer value**. Floating-point expressions are not allowed.
- The value of expression is tested, in order, against the values of the constants specified in the case statements.
- When a match is found, the statement sequence associated with that case is executed until the break statement or the end of the switch statement is reached.

```
switch (expression)
{
case constant1:
statement sequence
break;
case constant2:
statement sequence
break;
case constant3:
statement sequence
break;
...
default
statement sequence
}
```




- The **default statement is executed if no matches are found.**
- The default is optional and, if it is not present, no action takes place if all matches fail.
- In C++, a switch can have at least 16,384 case statements. However, in practice, you will want to limit the number of case statements to a smaller amount for efficiency.
- Although case is a label statement, it cannot exist by itself, outside of a switch.
- The break statement is one of C++'s jump statements. When break is encountered in a switch, program execution "jumps" to the line of code following the switch statement.

```
switch (expression)
{
  case constant1:
    statement sequence
    break;
  case constant2:
    statement sequence
    break;
  case constant3:
    statement sequence
    break;
  ...
  default
    statement sequence
}
```




- Technically, the break statements inside the switch statement are optional. They terminate the statement sequence associated with each constant.
- If the break statement is omitted, execution will continue on into the next case's statements until either a break or the end of the switch is reached.
- There are three important things to know about the switch statement:
 - (1) The switch differs from the if statement in that switch can only test for equality, whereas if can evaluate any type of relational or logical expression.
 - (2) No two case constants in the same switch can have identical values.
 - (3) If character constants are used in the switch statement, they are automatically converted to integers.
- The switch statement can be used instead of the if-else if-else statement to implement a sequence of parallel alternatives. It provides an easy way to perform execution to different parts of code based on the value of the expression.

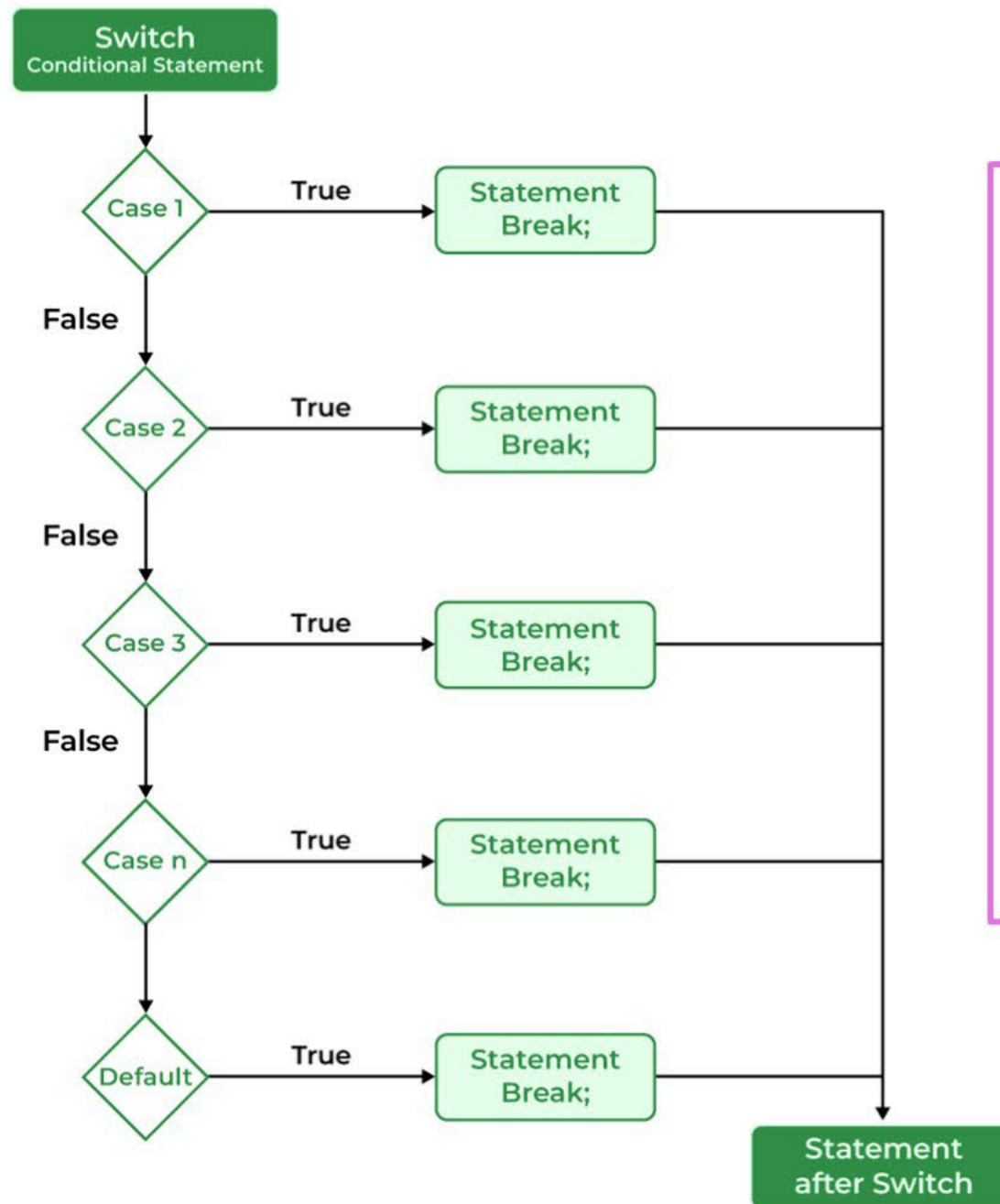


- Following are the main differences between switch statement and if-else if-else statement in C++:

switch statement	if - else if - else statement
It executes the different cases on the basis of the value of the switch variable.	It executes the different blocks based on the condition specified.
It can only evaluate the int or char type expressions.	It can evaluate any type of expression.
Faster and easier to read for a large number of conditions.	It can get messy when there are lots of conditions.



- The flowchart of switch statement is:



The working of the switch statement in C++ program is as follows:

- 1. Step 1:** The switch expression is evaluated.
- 2. Step 2:** The evaluated value is then matched against the present case values.
- 3. Step 3A:** If the matching case value is found, that case block is executed.
- 4. Step 3B:** If the matching code is not found, then the default case block is executed if present.
- 5. Step 4:** Statements after the switch statement is executed.



There are some rules that we need to follow when using switch statements in C++:

1. The case value must be either **int** or **char** type.
 2. There can be any number of cases.
 3. No duplicate case values are allowed.
 4. Each statement of the case can have a break statement. It is optional.
 5. The default Statement is also optional.
- The **break** keyword is used in the switch case to break out of the switch when encountered.
 - It is used at the end of every case block so that when the matching case is executed, the program control comes out of the loop. However, the break keyword is optional. **If omitted, all the cases after the matching case will also be executed.**
 - 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.**

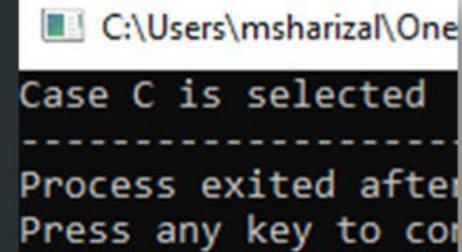


Example 1

Switch statement (selecting ONE of the available cases)

char data type

```
1  #include <iostream>
2  using namespace std;
3
4  int main()
5  {
6      char x = 'C'; // switch variable
7
8      // switch statements
9      switch (x)
10     {
11         case 'A':
12             cout << "Case A is selected";
13             break;
14
15         case 'B':
16             cout << "Case B is selected";
17             break;
18
19         case 'C':
20             cout << "Case C is selected";
21             break;
22
23         default:
24             cout << "Case other than A, B, or C is selected";
25     }
26     return 0;
27 }
```



C:\Users\msharizal\One
Case C is selected

Process exited after
Press any key to con

Statement line 20 is printed out since the case 'C' is declared for variable char x.



Example 2

Switch statement (selecting NOT one of the available cases)

char data type

```
1  #include <iostream>
2  using namespace std;
3
4  int main()
5  {
6      char x = 'Z'; // switch variable
7
8      // switch statements
9      switch (x)
10     {
11         case 'A':
12             cout << "Case A is selected";
13             break;
14
15         case 'B':
16             cout << "Case B is selected";
17             break;
18
19         case 'C':
20             cout << "Case C is selected";
21             break;
22
23         default:
24             cout << "Case other than A, B, or C is selected";
25     }
26     return 0;
27 }
```

```
C:\Users\msharizal\OneDrive - Higher Colleges of Techno
Case other than A, B, or C is selected
-----
Process exited after 0.2266 seconds with
Press any key to continue . . .
```

Statement line 24 is printed out since the case 'Z' is declared for variable char x and not within the provided case lists.

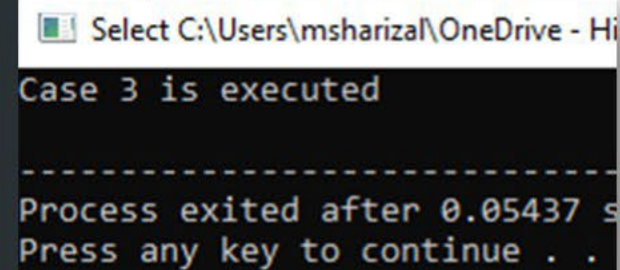


Example 3

Switch statement (selecting ONE of the available cases)

```
1  #include <iostream>
2  using namespace std;
3
4  int main()
5  {
6      int var = 3; // switch variable
7
8      // switch statements
9      switch (var)
10     {
11         case 1:
12             cout << "Case 1 is executed" << endl;
13             break;
14
15         case 2:
16             cout << "Case 2 is executed" << endl ;
17             break;
18
19         case 3:
20             cout << "Case 3 is executed" << endl;
21             break;
22
23         case 4:
24             cout << "Case 4 is executed" << endl;
25             break;
26
27         default:
28             cout << "Choice other than Case 1, Case 2, or Case 3 ";
29     }
30     return 0;
31 }
```

int data type



Select C:\Users\msharizal\OneDrive - Hi...

Case 3 is executed

Process exited after 0.05437 s

Press any key to continue . .

Statement line 20 is printed out since the case '3' is declared for variable int var.



Example 4

Switch statement (selecting one of the available cases but without BREAK)

```
1  #include <iostream>
2  using namespace std;
3
4  int main()
5  {
6      int var = 3; // switch variable
7
8      // switch statements
9      switch (var)
10     {
11         case 1:
12             cout << "Case 1 is executed" << endl;
13
14         case 2:
15             cout << "Case 2 is executed" << endl ;
16
17         case 3:
18             cout << "Case 3 is executed" << endl;
19
20         case 4:
21             cout << "Case 4 is executed" << endl;
22
23         default:
24             cout << "Choice other than Case 1, Case 2, or Case 3 ";
25     }
26     return 0;
27 }
```

int data type

```
C:\Users\msharizal\OneDrive - Higher Colleges of Technology\N
Case 3 is executed
Case 4 is executed
Choice other than Case 1, Case 2, or Case 3
-----
Process exited after 0.0386 seconds with return value 0
Press any key to continue . . .
```

Statement lines 18, 21, and 24 are printed out since the case '3' is declared for variable int var and there is no break statement within the program.



Advantages of switch Statement in C++:

1. Easier to debug and maintain for a large number of conditions.
2. Faster execution speed.
3. Easier to read than if else if.

Disadvantages of switch Statement in C++:

1. Switch case can only evaluate int or char type.
2. No support for logical expressions.
3. Have to keep in mind to add a break in every case.



Class Activity 2

Drag and Drop

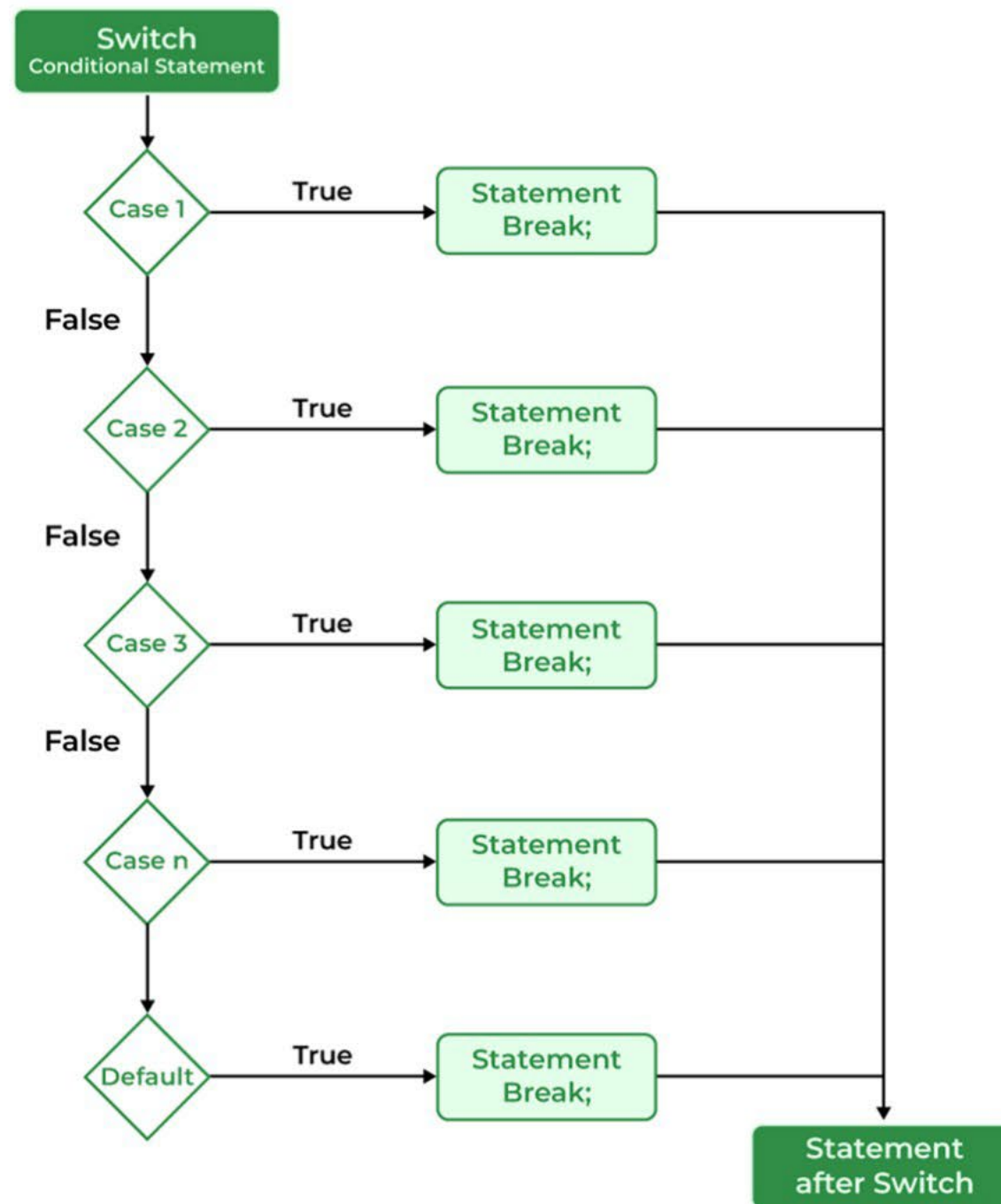
Can be complex
if many
conditions

Drag the descriptors to the correct statement.

^ Instructions	
Switch statement	if-else if-else statement



Summary



Higher
Colleges of
Technology



كليات
التقنية
العليا

Thank You



800 MyHCT (800 69428)



www.hct.ac.ae



HCT_UAE |



hctuae