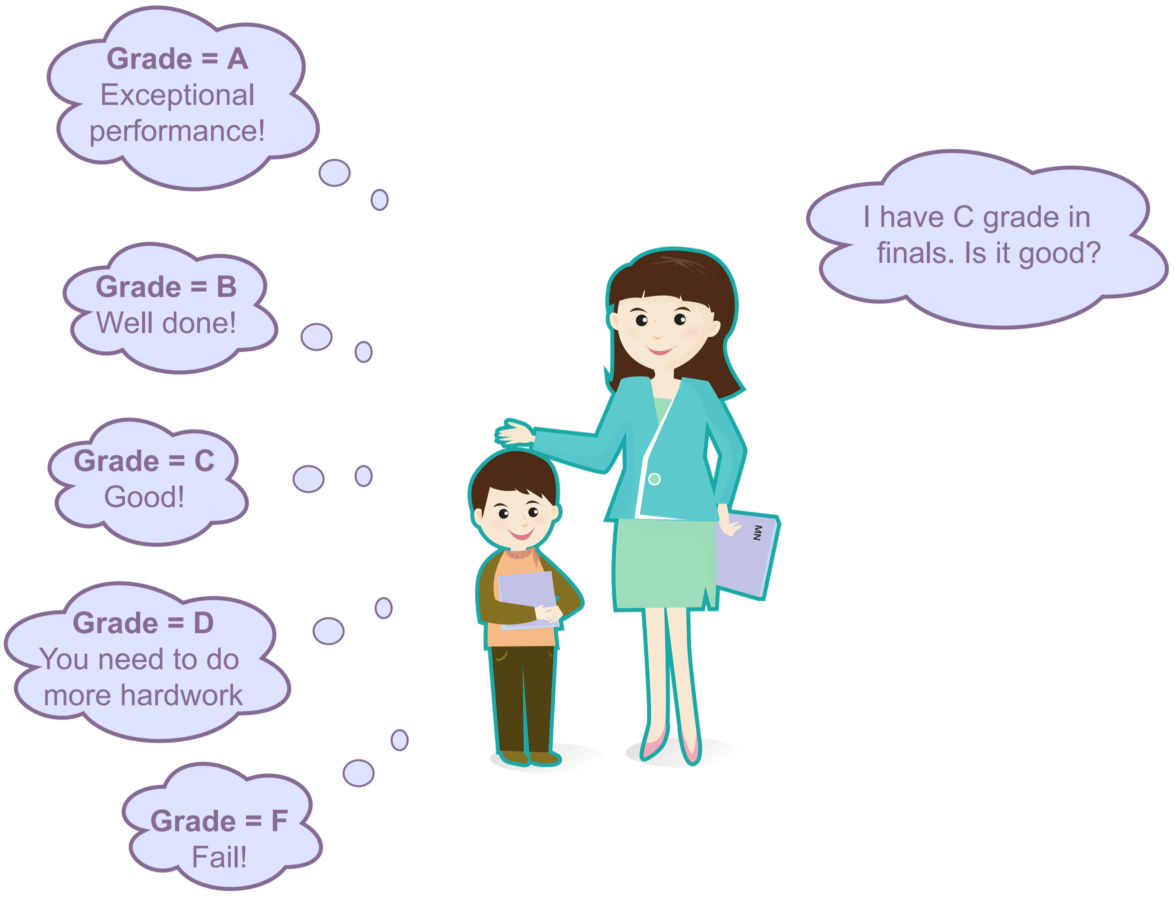
switch Statement

In this lesson, you will learn about the switch statement in C++.

We'll cover the following

Introduction

Suppose your teacher is writing remarks on your report card based on your grade.


Grade = A Exceptional performance!
Grade = B Well done!
Grade = C Good!
Grade = D You need to do more hardwork
Grade = F Fail!
I have C grade in finals. Is it good?

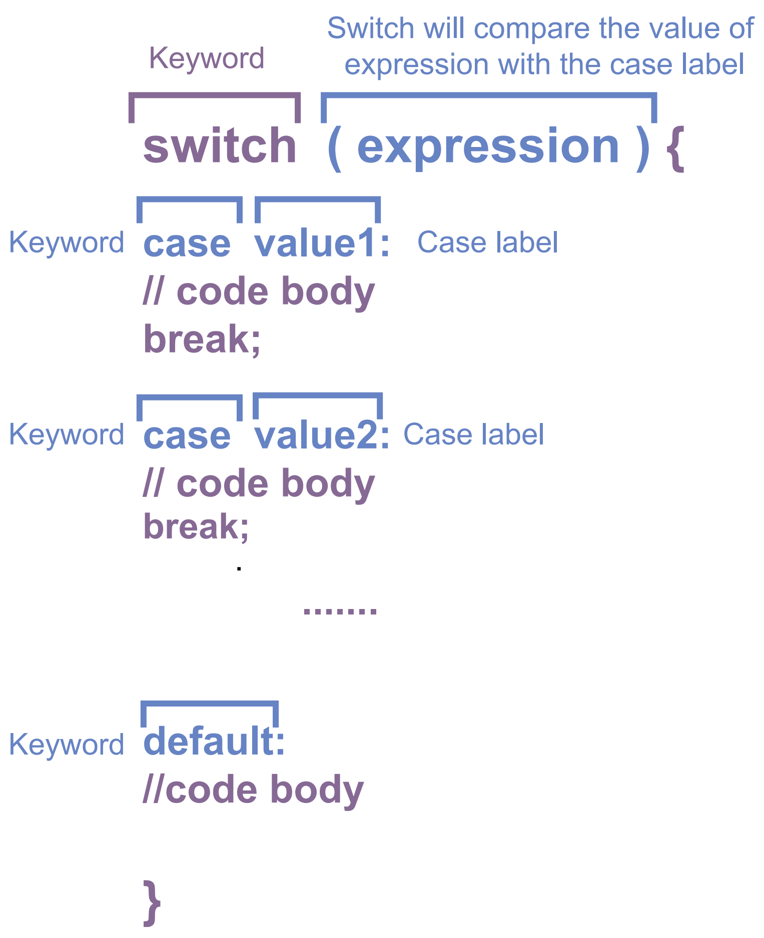
We can use the `else-if` statement here, but the number of choices is extensive. Therefore, `else-if` makes our code slow and complicated. Here, the `switch` statement comes in. Whenever we have to check the value of a single variable against an extensive number of choices, it is better to use the `switch` statement.

---

*The **switch** statement evaluates the given expression and then compares its value with each case label. If the value of a case label equals the value of the expression, the statement(s) specific to that case is executed.*

Syntax

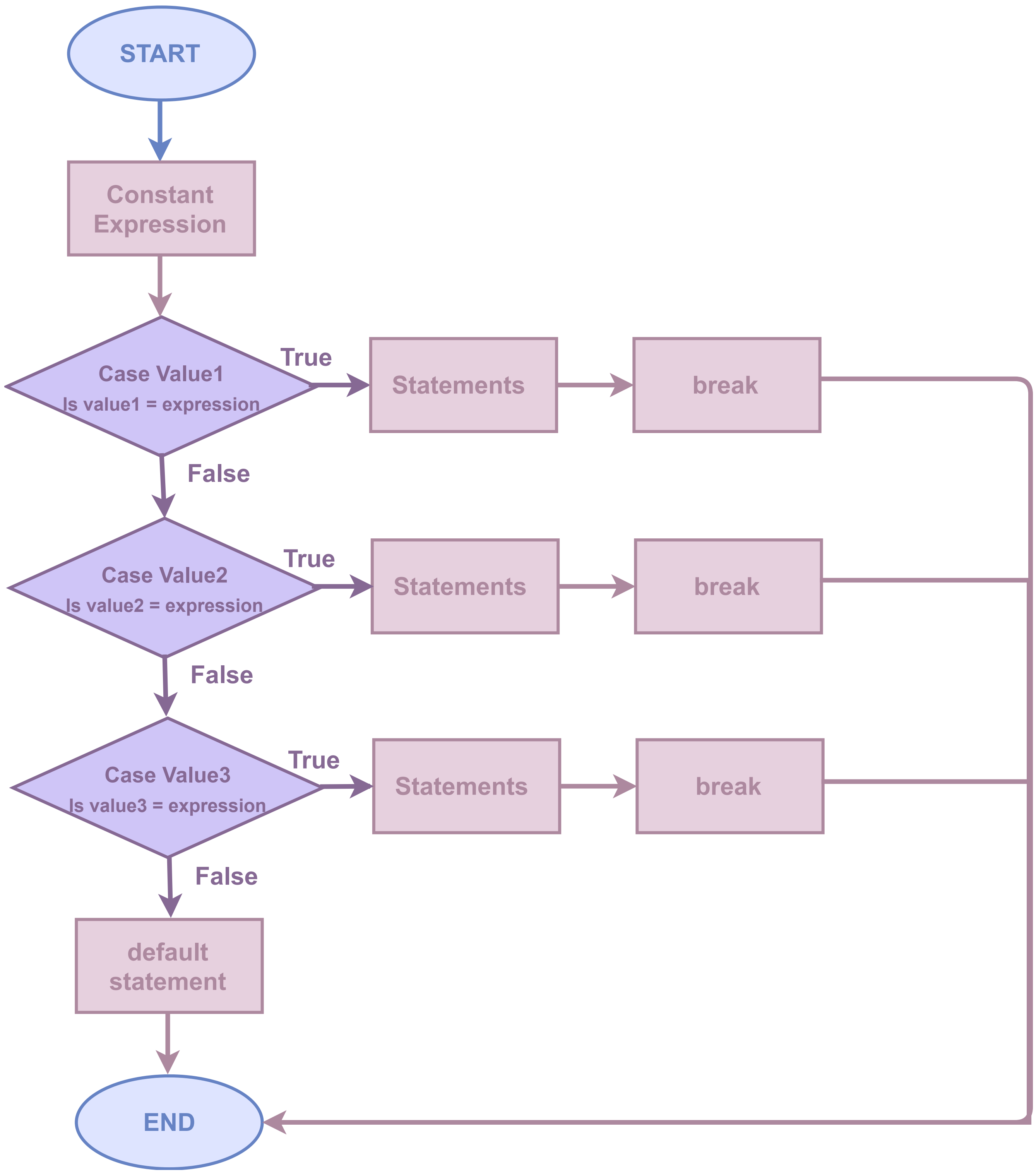
The basic syntax of the `switch` statement is given below:



📝 Switch expression and case label only accept variables of `int` or `char` data types.

Flowchart

The flowchart given below explains the workings of the `switch` statement:



In the figure above:

- The `switch` statement compares the value of the expression with the label of the `case`.
- If the value of the expression equals the `case` label, then the statements following this `case` are executed until it encounters a `break` statement.

- When the compiler encounters a `break` statement, it transfers the control to the line after the switch block.
- If the value of the expression does not match any of the `case` labels, then the `default` case is executed.

> 📝 If we don’t add a `break` statement to a case, the code specific to all the proceeding cases is also executed.

> 📝 The `default` case is optional in the `switch` statement.

Example program for specific values

Let’s convert the example given above into a C++ program.

Run the code below and see how the `switch` statement works!

```
#include <iostream>

using namespace std;

int main() {
  // Initialize variable grade
  char grade = 'C';
  // switch statement
  switch (grade) {
    // first case
    case 'A':
      cout << "Exceptional performance!";
      break;
    // second case
    case 'B':
      cout << "Well done!";
      break;
    // third case
    case 'C':
      cout << "Good!";
      break;
    // fourth case
    case 'D':
      cout << "You need to do more hardwork!";
      break;
    // fifth case
    case 'F':
      cout << "Fail";
      break;
    // default case
    default:
      cout << "Invalid input";
  }
  return 0;
}
```

Explanation

Line No. 7: Sets the value of `grade` to `C`

Line No. 9: The `switch` statement compares the value of the `grade` with the case labels.

Line No. 11: The value of `grade` is not equal to the value of the `case` label that is `A`. Therefore, statements following this case are not executed.

Line No. 15: The value of `grade` is not equal to the value of the `case` label that is `B`. Therefore, statements following this case are not executed.

Line No. 19: The value of `grade` is equal to the value of the `case` label that is `C`. Therefore, statements following this case are executed.

Line No. 20: It prints `Good!` to the console.

Line No. 21: Encountering the `break` statement makes the compiler exit the `switch` block and continue execution from thereon.

> 📝 Comment lines No. 21, 25, and 29 in the above program. Then, run the program!
> You will see that if we don't use the `break` statement, all the cases after the correct case will be executed.

Example program for ranges of values

Consider the example given in the previous lesson. We can use the `switch` statement to test the range of values, but it is not a good way.

Press the **RUN** button and see the output!

```
#include <iostream>

using namespace std;

int main() {
  // Initialize variable money
  int money = 6;
  switch (money) {

    // first case
  case 20 ... 100:
    cout << "You can gift a watch" << endl;
    break;
    // compares value of case label from 10 to 19 with the value of money
  case 10 ... 19:
    cout << "You can gift a comic book " << endl;
    break;
    // compares value of case label from 9 to 5 with the value of money
  case 5 ... 9:
    cout << "You can gift a chocolate " << endl;
    break;
    // default case
  default:
    cout << "You can gift a pen " << endl;
  }
  return 0;
}
```

Explanation

In the above code, it seems that the `switch` statement is working in the same way as the `else-if` statement. However, there is a difference! Try to run the above code for `money = 101`.

With the `switch` statement, the output is `"You can gift a pen"`. Whereas, with `else-if`, the output is `"You can gift a watch"`.

In a switch statement, you have to define both the upper and lower range of values. The upper range of `money` is unknown; therefore, the `switch` statement is not a good option for testing ranges of values. If you want to test ranges, use the `else-if` statement.

Quiz

Q

If `percentage = 85`, then what is the output of the following code?

```
int percentage;

cout << "Grade : ";
```

```
switch (percentage) {
  case 90 ... 100:

    cout << "A";
    break;
  case 70 ... 89:
    cout << "B";
    break;
  case 50 ... 69:
    cout << "C";
    break;
  default:
    cout << "D";
}
```

Retake Quiz

Let’s discuss the conditional operator in the upcoming lesson.

See you there!