Switch Case statement in C++ with example

Switch case statement is used when we have multiple conditions and we need to perform different action based on the condition. When we have multiple conditions and we need to execute a block of statements when a particular condition is satisfied. In such case either we can use lengthy if else or switch case. The problem with lengthy if..else-if is that it becomes complex when we have several conditions. The switch case is a clean and efficient method of handling such scenarios.

The **syntax of Switch case** statement:

```
switch (variable or an integer expression)
{
     case constant:
     //C++ code
     ;
     case constant:
     //C++ code
     ;
     default:
     //C++ code
     ;
}
```

Switch Case statement is mostly used with break statement even though the break statement is optional. We will first see an example without break statement and then we will discuss switch case with break

Example of Switch Case

```
#include <iostream>
using namespace std;
int main(){
   int num=5;
   switch(num+2) {
      case 1:
        cout<<"Case1: Value is: "<<num<<endl;
      case 2:
        cout<<"Case2: Value is: "<<num<<endl;
      case 3:
        cout<<"Case3: Value is: "<<num<<endl;
      default:
        cout<<"Default: Value is: "<<num<<endl;
   }
   return 0;
}
```

Output:

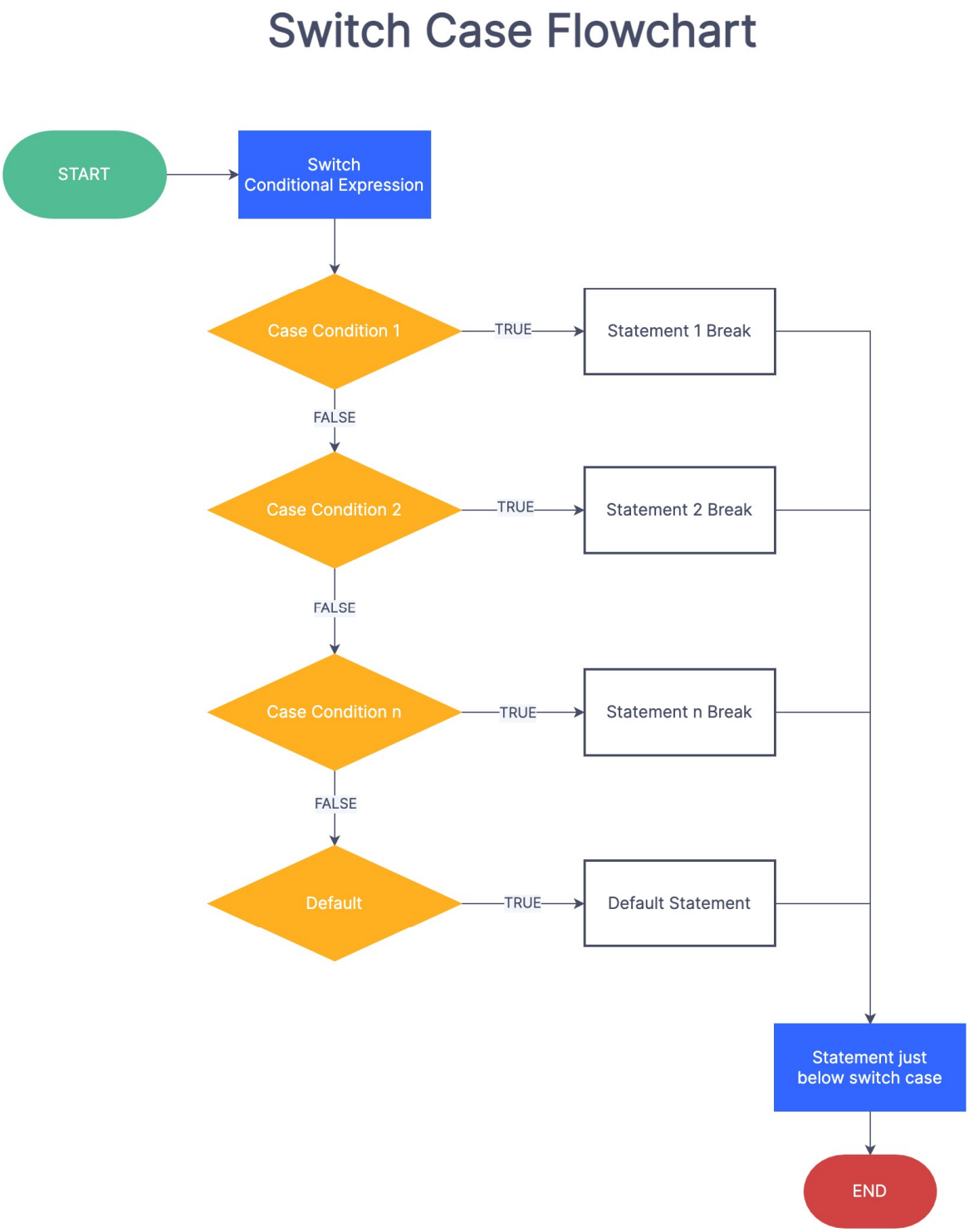
```
Default: Value is: 5
```

Explanation: In switch I gave an expression, you can give variable as well. I gave the expression num+2, where num value is 5 and after addition the expression resulted 7. Since there is no case defined with value 4 the default case got executed.

Switch Case Flow Diagram

It evaluates the value of expression or variable (based on whatever is given inside switch braces), then based on the outcome it executes the

corresponding case.



Break statement in Switch Case

Before we discuss about break statement, Let's see what happens when we don't use break statement in switch case. See the example below:

```
#include <iostream>
using namespace std;
int main(){
   int i=2;
   switch(i) {
      case 1: cout<<"Case1 "<<endl;
      case 2: cout<<"Case2 "<<endl;
      case 3: cout<<"Case3 "<<endl;
      case 4: cout<<"Case4 "<<endl;
      default: cout<<"Default "<<endl;
   }
   return 0;
}
```

Output:

```
Case2
Case3
Case4
Default
```

In the above program, we have the variable i inside switch braces, which means whatever the value of variable i is, the corresponding case block gets executed. We have passed integer value 2 to the switch, so the control switched to the case 2, however we don't have break statement after the case 2 that caused the flow to continue to the subsequent cases till the end. However this is not what we wanted, we wanted to execute the right case block and ignore rest blocks. The solution to this issue is to use the break statement in after every case block.

Break statements are used when you want your program-flow to come out of the switch body. Whenever a break statement is encountered in the switch body, the execution flow would directly come out of the switch, ignoring rest of the cases. This is why you must end each case block with the break statement.

Let's take the same example but this time with break statement.

```
#include <iostream>
using namespace std;
int main(){
   int i=2;
   switch(i) {
      case 1:
        cout<<"Case1 "<<endl;
        break;
      case 2:
        cout<<"Case2 "<<endl;
        break;
      case 3:
        cout<<"Case3 "<<endl;
```

```
        break;
      case 4:
        cout<<"Case4 "<<endl;
        break;
      default:
        cout<<"Default "<<endl;
    }
    return 0;
}
```

Output:

```
Case2
```

Now you can see that only case 2 got executed, rest of the subsequent cases were ignored.

Why didn't I use break statement after default?
The control would itself come out of the switch after default so I didn't use break statement after it, however if you want you can use it, there is no harm in doing that.

Important Notes

1) Case doesn't always need to have order 1, 2, 3 and so on. It can have any integer value after case keyword. Also, case doesn't need to be in an ascending order always, you can specify them in any order based on the requirement.

2) You can also use characters in switch case. for example –

```
#include <iostream>
using namespace std;
int main(){
   char ch='b';
   switch(ch) {
      case 'd': cout<<"Case1 ";
      break;
      case 'b': cout<<"Case2 ";
      break;
      case 'x': cout<<"Case3 ";
      break;
      case 'y': cout<<"Case4 ";
      break;
      default: cout<<"Default ";
   }
   return 0;
}
```

3) Nesting of switch statements are allowed, which means you can have switch statements inside another switch. However nested switch statements should be avoided as it makes program more complex and less readable.