Lecture 12.1

Topics

1. Extended Conditional Structure – `switch` Statement

---

1. Extended Conditional Structure – `switch` Statement

Besides the extended **`if-else-if-else`** statements, C has one structure that can handle multiple options. This structure is called a **`switch`** statement, which is a composite statement used to make a selection among many options.

1.1 `switch` Syntax and Flowchart

Its syntax is given as follows,

```
switch (testExpression) {
  case constantValue1 :
    statement1
    break;
  case constantValue2 :
    statement2
    break;

  . . .

  case constantValueN :
    statementN
    break;
  default :
    statementDefault
}
```

where

(1) **`testExpression`** must produce an integral value. It is commonly given as a unary expression in the form of an identifier.

(2) **`constantValue1, constantValue2, …, constantValueN`** represent all possible values matching with the above integral value (i.e., **testExpression** or its result).

The **`switch`** statement will have the following characteristics:

a. The test expression after the **`switch`** keyword must be an integral type.

b. The expression after the **`case`** keyword must be a constant expression. The expression together with the **`case`** keyword is called a **`case-label`** statement. Note that a constant expression is an expression that is evaluated at compiled time, not run time.

c. No two **`case labels`** can have the same value.

d. Two or more **`case labels`** can be associated with the same statement(s).

e. The **`default`** label is not required. If there is no match, then the control jumps outside of the **`switch`** statement.

f. There can be at most one `default` label. It can be placed anywhere, but it is mostly placed last in the structure.

A general flowchart is given in **Figures 1 & 2** as follows,

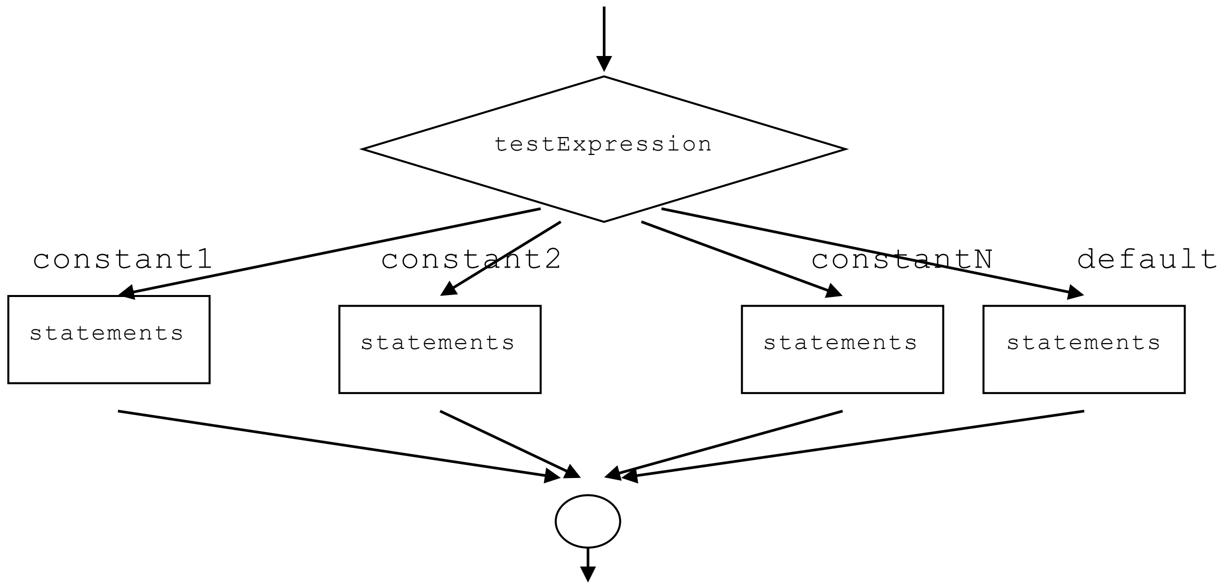

Figure 1 A general `switch` structure

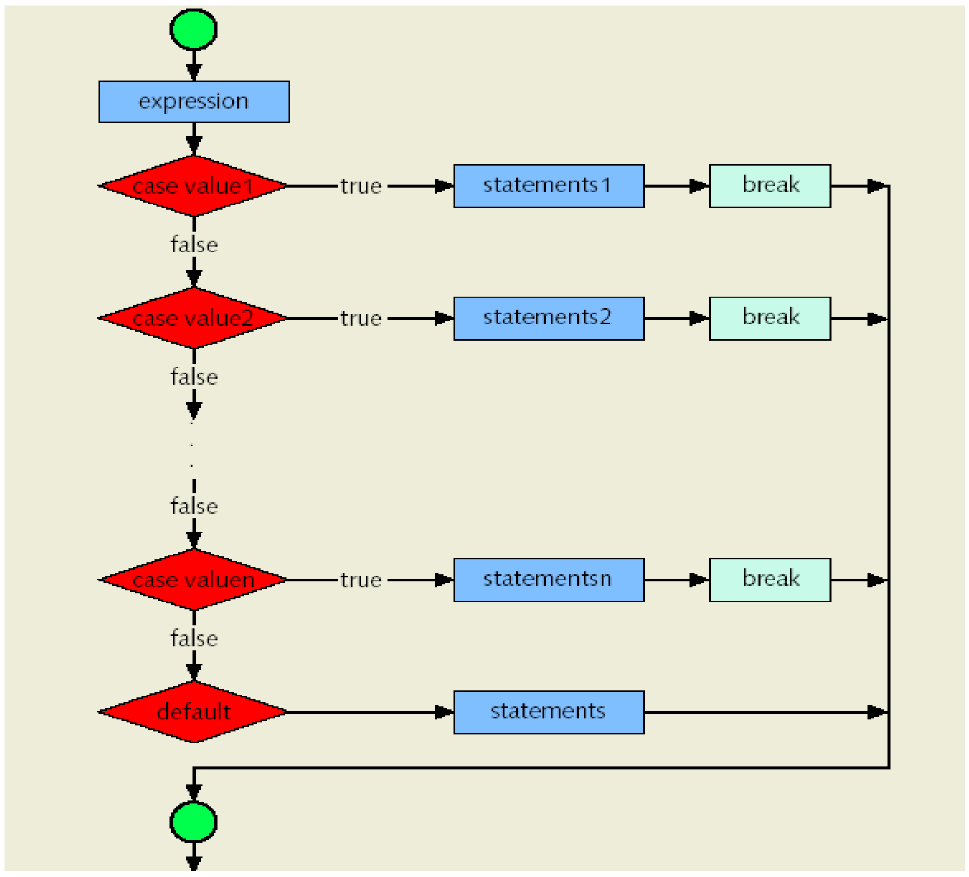

Figure 2 A `switch` structure

An example with a function **`printDaySwitch()`** can be written as follows,

```
void printDaySwitch(int iDay) {

  switch (iDay) {
    case 1:
      cout << "\nIt is Sunday!" << endl;
      break;
    case 2:
      cout << "\nIt is Monday!" << endl;
      break;
    case 3:
      cout <<  "\nIt is Tuesday!" << endl;
      break;
    case 4:
      cout <<  "\nIt is Wednesday!" << endl;
      break;
    case 5:
      cout <<  "\nIt is Thursday!" << endl;
      break;
    case 6:
      cout <<  "\nIt is Friday!" << endl;
      break;
    case 7:
      cout <<  "\nIt is Saturday!" << endl;
      break;
    default:
      cout <<  "\nIt is an INVALID selection!" << endl;
  }

  return;
}
```

1.2 Example – Menu setup

Recall that a menu program will provide the user with options and selections. The execution will continue after an option is selected and entered to the program.

Again, let us consider a menu of four basic arithmetic operations:

```
MENU --

(1) Add
(2) Subtract
(3) Multiply
(4) Divide
(5) Quit
```

The discussion will present a menu using `do-while` and `switch` structures.