switch case

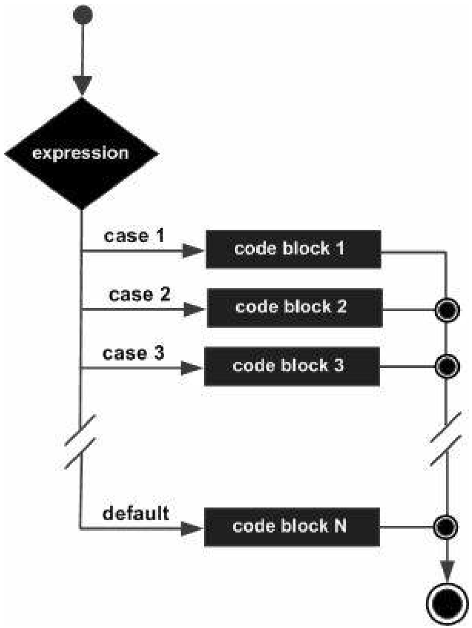
Comp Sci 1570 Introduction to C++

Missouri S&T | Computer Science

Outline

Outline

switch case


expression
case 1
code block 1
case 2
code block 2
case 3
code block 3
default
code block N

Outline

Syntax

```
switch (expression)
{
  case constant1:
     group-of-statements-1;
     break;
  case constant2:
     group-of-statements-2;
     break;
  .
  .
  .
  default:
     default-group-of-statements
}
```

Outline

Basics

```
switch (expression){
  case constant1:
     group-of-statements-1;
     break;
  case constant2:
     group-of-statements-2;
     break;
  default:
     default-group-of-statements
}
```

- switch evaluates expression and checks if it is equivalent to constant1; if it is, it executes statements-1 until it finds the break statement.
- When it finds this break statement, the program jumps to the end of the entire switch statement (the closing brace).
- If expression was not equal to constant1, it is then checked against constant2.
- If it is equal to this, it executes group-of-statements-2 until a break is found, when it jumps to the end of the switch.
- If the value of expression did not match any of the previously specified constants (there may be any number of these), the program executes the statements included after the default: label, if it exists (since it is optional).

Outline

How it works

- The value of control_var is compared to constant1.
- If the values are equal, every statement after that is executed until a break is encountered, at which point control exits the switch-case statement.
- If they don't match, then C++ makes comparison to the value in the next case.
- This continues until a match is found, or until the default is encountered or until the end of the switch-case statement.

Outline

Rules

- Typically this expression is just a single variable, but it can be something more complex like $nX + 2$ or $nX - nY$.
- Expression must have an integral or enumerated type, or be of a class type in which the class has a single conversion function to an integral or enumerated type (that is, char, short, int, long, long long, or enum). Floating point variables and other non-integral types may not be used here.
- Any number of case statements within a switch. Each case is followed by the value to be compared to and a colon.
- Constant for a case must be the same data type as the variable in the switch, and it must be a constant or a literal, and known at compile time.
- When the variable being switched on is equal to a case, the statements following that case will execute until a break statement is reached.
- When a break statement is reached, the switch terminates, and the flow of control jumps to the next line following the switch statement.
- Not every case needs to contain a break. If no break appears, the flow of control will fall through to subsequent cases until a break is reached.
- A switch statement can have an optional default case, which must appear at the end of the switch. 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.

Outline

Nested switch case

```
switch(ch1)
{
    case 'A':
        cout << "A from Outer switch";
        switch(ch2)
        {
            case 'A':
                cout << "A from Inner switch";
                break;
            case 'B': // ...
        }
        break;
    case 'B': // ...
}
```