

Chapter - 8

More Control

Statements

for Statement

General form:

```
for (initial-statement; condition;  
    iteration-statement)  
    body-statement;
```

Is equivalent to:

```
initial-statement;  
while (condition) {  
    body-statement;  
    iteration-statement;  
}
```

for Example

```
}
```

```
}
```

Note that `counter` goes from 0 to 4. Normally you count five items as 1,2,3,4,5. You will get along much better in C++ if you change your thinking to zero-based counting and count five items as 0,1,2,3,4.

for vs. while

```
int main() {  
    // ...  
    count = 0;  
    while (count < 5) {  
        // ...  
        ++count;  
    }  
    std::cout << "The grand total is " << total << '\n';  
}  
  
int main() {  
    // ...  
    for (count = 0; count < 5; ++count) {  
        // ...  
    }  
    std::cout << "The grand total is " << total << '\n';  
}
```

Question: What Does this Program Print?

```
/*
```

```
*
```

```
*/
```

Question: Why Does this Program Print the Wrong Answer?

Program (cont.)

```
{
```

```
    ++three_count;
```

```
    ++seven_count;
```

```
}
```

switch Statement

General form:

switch

case *constant1*:
statement

break;

case *constant2*:
statement

default :
statement

break;

case *constant3*:
statement

break;

From the *calc* program

}

As a **switch** Statement

```
main() {
```

```
    break;
```

As a switch (cont.)

```
switch (oper_char) {
    case '+':
        result += value;
        break;
    case '-':
        result -= value;
        break;
    case '*':
        result *= value;
        break;
    case '/':
        if (value == 0) {
            std::cout << "Error:Divide by zero\n";
            std::cout << "    operation ignored\n";
        } else
            result /= value;
        break;
    default:
        std::cout << "Unknown op. " <<oper_char << '\n';
        break;
}
```

Ending breaks

A **break** is not required at the end of a case switch:

```
// a not so good example of programming
switch (control) {
    case 0:
        std::cout << "Reset\n";
    case 1:
        std::cout << "Initializing\n";
        break;
    case 2:
        std::cout << "Working\n";
}
```

Did the programmer intend to fall through for case 0 or did he forget the break statement?

A Better Switch

```
// a better example of programming
switch (control) {
    case 0:
        std::cout << "Reset\n";
        // Fall through
    case 1:
        std::cout << "Initializing\n";
        break;
    case 2:
        std::cout << "Working\n";
}
```

Let's Add a New Case at the End

```
// We have a little problem
switch (control) {
    case 0:
        std::cout << "Reset\n";
        // Fall through
    case 1:
        std::cout << "Initializing\n";
        break;
    case 2:
        std::cout << "Working\n";
    case 3:
        std::cout << "Closing down\n";
}
```

We have a problem.

Our Problem is Fixed.

```
std::cout << "Reset\n";
```

```
std::cout << "Initializing\n";  
break;
```

```
std::cout << "Working\n";  
break;
```

```
break;
```

```
}
```

But what happens if control is '5'. The switch does nothing. Did the programmer intend for this to happen or is it just an accident.

Final switch

```
std::cout << "Reset\n";
```

```
std::cout << "Initializing\n";  
break;
```

```
std::cout << "Working\n";  
break;
```

```
break;  
default:
```

```
break;
```

```
}
```

A “default” is required even if it is only:

```
default:
```

```
break;
```


switch, break, and continue

```
while (1) {  
    std::cout << "Enter operator and number: ";  
    std::cin >> oper_char >> value;  
    if (oper_char == 'Q') break;  
  
    switch (oper_char) {  
        case '+':  
            result += value;  
            break;  
        // .....  
        case 'h':  
            // ... help stuff ..  
            continue;  
    }  
    std::cout << "Result: " << result << '\n';  
}  
return (0); // End of program
```

**break outside
switch**

**continue
(switch irrelevant)**

**break inside
switch**