# Comparing if, switch and nested if

In the last lesson you learned how to write programs using if, else, and switch. In this lesson will compare these different methods. You will learn to identify different construct from a flowchart.

We will start with a program that inputs a month and prints the season. In the previous lesson this was done using a switch construct:

```
 0 // Input a month, print the season
 1 #include <iostream.h>
 2 int main()
 3 { int month;
 4   cout<<"Enter the number of a month:";
 5   cin>>month;
 6   switch (month)
 7   { case 1: case 2: case 12:  cout<<"Winter\n"; break;
 8     case 3: case 4: case 5:   cout<<"Spring\n"; break;
 9     case 6: case 7: case 8:   cout<<"Summer\n"; break;
10     case 9: case 10: case 11: cout<<"Fall\n";   break;
11     default: cout<<month<<" is not a valid month\n";
12   } // switch month
13   return 0;
14 } // main
```

The switch statement is often an easy way to write a solution, but it is never necessary. Any program that could be written using switch can be written using if and else. The program below is very straightforward. Sometimes the best solution is not the most elegant. There are big advantages to making a program very clear.

```
 0 // Input a month, print the season using if
 1 #include <iostream.h>
 2 int main()
 3 { int month;
 4   cout<<"Enter the number of a month:";
 5   cin>>month;
 6   if (month==1) cout<<"Winter\n";
 7   if (month==2) cout<<"Winter\n";
 8   if (month==3) cout<<"Spring\n";
 9   if (month==4) cout<<"Spring\n";
10   if (month==5) cout<<"Spring\n";
11   if (month==6) cout<<"Summer\n";
12   if (month==7) cout<<"Summer\n";
13   if (month==8) cout<<"Summer\n";
14   if (month==9) cout<<"Fall\n";
15   if (month==10) cout<<"Fall\n";
16   if (month==11) cout<<"Fall\n";
17   if (month==12) cout<<"Winter\n";
```

```
18   // an else here would mean if the month is not 12
19   // we can not use else!!!
20   if (month<1 || month >12)
21      cout<<month<<" is not a valid month\n";
22   return 0;
23 } // main
```

Make sure that you understand why line 20 cannot use else. Put in an else and run the program. Does it work? What happens?

The solution below uses || (the or operator) to simplify the code:

```
 0 // Input a month, print the season with if and ||
 1 #include <iostream.h>
 2 int main()
 3 { int month;
 4   cout<<"Enter the number of a month:";
 5   cin>>month;
 6   if (month==1 || month==2 || month==12) cout<<"Winter\n";
 7   if (month==3 || month==4 || month==5) cout<<"Spring\n";
 8   if (month==6 || month==7 || month==8) cout<<"Summer\n";
 9   if (month==9 || month==10 || month==11) cout<<"Fall\n";
10   // an else here would mean if the month is not 12
11   // we can not use else!!!
12   if (month<1 || month >12)
13      cout<<month<<" is not a valid month\n";
14   return 0;
15 } // main
```

The next modification uses a nested if else. Notice that we can now use the else at the end:

```
 0 // Input a month, print the season
 1 #include <iostream.h>
 2 int main()
 3 { int month;
 4   cout<<"Enter the number of a month:";
 5   cin>>month;
 6   if (month==1 || month==2 || month==12)
 7     cout<<"Winter\n";
 8   else if (month==3 || month==4 || month==5)
 9     cout<<"Spring\n";
10   else if (month==6 || month==7 || month==8)
11     cout<<"Summer\n";
12   else if (month==9 || month==10 || month==11)
13     cout<<"Fall\n";
14   else
15     cout<<month<<" is not a valid month\n";
16   return 0;
17 } // main
```

It should be clear why the else could not be used for the invalid moth in the first example, but it is OK now. Experiment with the else until you understand the concept.

The solution below takes advantage of the fact that there are ranges of values:

```
 0 // Input a month, print the season using && (and)
 1 #include <iostream.h>
 2 int main()
 3 { int month;
 4   cout<<"Enter the number of a month:";
 5   cin>>month;
 6   if (month==1 || month ==2 || month==12)
 7     cout<<"Winter\n";
 8   else if (month>=3 && month <=5)
 9     cout<<"Spring\n";
10   else if (month >=6 && month<=8)
11     cout<<"Summer\n";
12   else if (month>=9 && month<=11)
13     cout<<"Fall\n";
14   else
15      cout<<month<<" is not a valid month\n";
16   return 0;
17 } // main
```

Flowcharts do not have a special symbol for switch statements. When you look at a flowchart, look for several decision blocks that all compare a variable to a constant. When you see this pattern, you ***may*** be able to use switch in your code. In programming there are usually several ways of solving a problem. The most important consideration is whether it works or not. If it works, the second consideration is whether it is easy to understand. If you have to come back in a year and modify it, will you be able to?

## Problem

A company pays employees on different schedules. All employees are paid an annual salary. Type 0 employees get paid every week. Type 1 employees are paid twice a month. Type 2 employees are paid once a month. Calculate the base pay for each employee.

To develop an algorithm for this problem you first decide what the output is to be. Next you figure out what data you need to produce the output. The user is asked to enter raw data. Raw data can be input by the user without doing any calculations. Information is calculated from the raw data. Usually it will take several steps to get from the raw data to the information to be output. You may calculate some values that are not output. Printing out these intermediate values while you are developing the program can be a tremendous help in debugging your program.

For this problem we can see that to calculate the base pay, we will need the pay type (0, 1, or 2: everything else is invalid) and the annual salary.

Is the calculation exactly the same for everyone? No. Is the calculation dependent upon a single variable? Yes. This makes us think that perhaps a switch block might be a good way to write the solution.

```
 0 // Calculate base pay
 1 #include <iostream.h>
 2 int main()
 3 { int payType; //0 weekly; 1 2x per month; 2 monthly
 4   double salary; //annual salary
 5   cout<<"Enter pay type: ";
 6   cin>>payType;
 7   cout<<"Enter the annual salary: $";
 8   //we print $ so the user doesn't
 9   cin>>salary;
10   switch (payType)
11   { case 0: cout<<"Base pay = "<<salary/52.0<<"\n"; break;
12     case 1: cout<<"Base pay = "<<salary/24.0<<"\n"; break;
13     case 2: cout<<"Base pay = "<<salary/12.0<<"\n"; break;
14     default: cout<<"Pay type must be 0, 1, or 2\n"; break;
15   } //switch paytype
16   return 0;
17 } // main
```

Run the program. The program produces the correct output, but there is a small problem.

When the user enters a pay type such as 5, the program asks the salary before it tells the user that the pay type is invalid. Actually, the user should not have had to enter the salary if it wouldn't be used.

```
 0 // Calculate base pay
 1 #include <iostream.h>
 2 int main()
 3 { int payType; //0 weekly; 1 2x per month; 2 monthly
 4   double salary; //annual salary
 5   cout<<"Enter pay type: ";
 6   cin>>payType;
 7   if (payType <0 || payType >2)
 8      cout<<"Pay type must be 0, 1, or 2\n";
 9   else
10   {  cout<<"Enter the annual salary: $";
11      cin>>salary;
12      switch (payType)
13      { case 0: //paid weekly
14          cout<<"Base pay = "<<salary/52.0<<"\n"; break;
15        case 1: //paid twice a month
16          cout<<"Base pay = "<<salary/24.0<<"\n"; break;
17        case 2: //paid monthly
18         cout<<"Base pay = "<<salary/12.0<<"\n"; break;
19      } //switch paytype
20   } //else
21   return 0;
22 } // main
```

This program has a switch block nested **inside** the if else block. Notice the level of indentation. Every { has the matching } directly underneath it and every } uses a comment to explain its purpose. The indentation is an important part of writing good code. Without the indentation and the comments, the code is much harder to understand. Code that is indented correctly and commented is called self-documented code. It **should** be a goal of every good programmer! (Of course it still has to produce the correct results.)

**Exercise**: Can you write the code above using a nested if else instead of switch? Try to avoid duplicating the statements to print the prompt and read in the salary. If we want to change the prompt, it is easier if it only appears once.