**WEEK 5: THE SELECTION STRUCTURE**

**Objectives**

Following completion of this topic, students should be able to:

• Understand selection structures

• Understand if, if-else and switch statements

**TASK 0. CONCEPTS: SELECTION STRUCTURE**

*NOTE: This task is to be completed in your own time* ***before*** *the start of your lab*

*This task is a pen and paper task – I.e. no computer is required.*

a. What is the difference between nested and cascaded if statements? Give examples

to illustrate this difference.

b. When can we use a conditional ‘?:’ operator? Give an example to justify your

argument.

c. What is a “dangling else” problem? Give an example.

**TASK 1. UNDERSTANDING A SIMPLE IF-STATEMENT AND SIMPLE OUTPUT**

**FORMATTING**

It is a very common programming practice to only execute a certain code segment if a

particular condition has been met. For example, only print a persons age if the age is

greater than 0. C++ provides the if statement to accommodate such situations. The

syntax is:

if (condition)

statement;

where condition is a logical expression and statement is any executable

statement.

• First, the condition is evaluated to determine its logical value, either true

or false*.*

• If the condition evaluates to true, the statement will be executed then

program flow will fall to the next statement immediately following the if

statement.

• If the condition evaluates to false*,* program flow will fall to the next

statement immediately following the if statement.

• If more than one statement is to be executed, they must be contained within

braces {…}.

• *Style note: All code contained within a if statement* ***must*** *be indented.*

Do the following:

a. Create a file named ‘*task1.cpp’* and add an appropriate file header, followed by

the preprocessor directive(s) that allow the program to send output to the

computer screen.

b. Create the main function declaring an integer variable named iNum and a float

variable named avg. *Note: Don’t forget to initialise variables*.

**c.** Add appropriate code to get user input and store this value in the variable, iNum**.**

d. Include in the body of main the following lines of code.

if (iNum % 2 == 0)

cout << iNum << ". . .\n";

replacing the space denote by ‘**. . .**’ with an appropriate string (word(s))

describing the variable iNum.

**e.** Compile the program to create an executable named **‘***task1’***.**

f. You should have received a compile *warning*. What is the warning and what does

it mean?

The warning should have to deal with the initialized float variable that was not utilized in the course of the program

g. Modify the code so that no compilation warnings or errors are produced.

• Even though this was a warning and not an error and the compilation will be

successful, it is imperative to remove all warnings as well as errors before

submitting assessable work.

• A warning is an indication that your code may have a *logical* error. Heed the

warning!

h. How did you remove the warning?

By deleting the initialized float variable and recompiling, the error disappeared

i. What is the output produced by the program if the following numbers are input:

i. 26 … Is Even

ii. 21 … Is Odd

j. Add another if statement to deal with odd numbers.

k. Compile and run your program to verify that it does work appropriately with both

odd and even numbers.

l. Test your program with several numbers that are negative, zero and positive to

ensure that it does work correctly. Which numbers have you tested your program

with?

1 2 3 4 5 6 21 26

**TASK 2. IF-ELSE CHAIN**

In solving many real world problems, different actions must be taken depending on the

value of the data. For example, calculating an area only if the measurements are positive,

performing a division only if the divisor is non-zero, etc. The if-else statement in C++

is used to implement such a decision structure. The most common way to use the if-else

statement is:

if (condition)

statementTrue;

else

statementFalse;

Where condition is a logical expression and statementTrue and statementFalse

are executable statements.

• If condition evaluates to true, statementTrue is executed. Program flow

then falls to the next statement immediately following the if-else statement.

• If condition evaluates to false, statementFalse is executed. Program flow

then falls to the next statement immediately following the if-else statement.

• If more than one statement is to be executed, they must be contained within

braces {…}.

• *Style note: All code contained within a if statement* ***must*** *be indented.*

This can be extended to deal with more complicated decision structures; the cascaded

if-else chain. Following, is the syntax:

if (condition1)

statement1;

else if (condition2)

statement2;

else

statement3;

Where condition1 and condition2 are logical expressions and statement1,

statement2 and statement3 are any executable statements.

• If condition1 is evaluated to true, statement1 will be executed. Program

flow then falls to the next statement immediately following the if-else chain.

• If condition1 is false, condition2 is evaluated and if it is true,

statement2 will be executed. Program flow then falls to the next statement

immediately following the if-else chain.

• If condition1 is false **and** condition2 is false, statement3 will be

executed. Program flow then falls to the next statement immediately following

the if-else chain.

• And so on.

Do the following:

a. Using *nedit*, create a source file named ‘*task2.cpp’* in this week’s directory.

b. Write a program which gets a character from user input and displays:

• *“*Individual is married.” if the user enters an ‘*m’.*

• *“*Individual is single*.”* if the user enters an ‘*s’*.

• *“*Individual is divorced.” if the user enters a ‘*d’*.

• *“*Individual is widowed.” if the user enters a ‘*w’*.

• *“*An invalid code was entered.” if the user enters anything else.

c. Have you used the if-else structure correctly? Compile your program using the

following command:

[g++ -Wall task2.cpp –o status]

If the program does not compile, remember to read the error message carefully to

determine the cause of the error and go to the appropriate line of code to fix it.

d. Run the program and test it for **all** possible values to ensure that it does work

appropriately.

**TASK 3.MORE ADVANCED SELECTION STRUCTURES.**

The following program will ask for input of an integer value, which represents a

month in the year, and will display the number of days in that particular month.

a. Implement the following program as ‘*task3.cpp’* in this week’s directory.

#include <iostream>

using namespace std;

int main()

{

int numOfDays = 0, month = 0;

cout << "Enter month: ";

cin >> month;

if (month == 1 || month == 3 || month == 5 ||

month == 7 || month == 8 || month == 10 ||

month == 12)

numOfDays = 31;

else if (month == 4 || month == 6 || month == 9 ||

month == 11)

numOfDays = 30;

else // February

numOfDays = 28;

cout << "There are " << numOfDays

<< " days in this month\n";

return 0;

}

b. What operation does the logical operator ‘||’ perform?

It performs the logical or operation. Takes 2 arguments and if either of them are true it returns true.

c. What are the other logical operators and what operations do they perform?

Beyond or there are the and and no operators. And is && and or is !. And takes 2 arguments and returns true if both are true. Not takes one argument and returns the opposite truth value of that argument.

d. Compile the above program using the following command:

[g++ -Wall task3.cpp –o month]

e. Execute your program, giving the following input, and record the output:

i. 1 (to represent January) Output:

31

ii. 4 (to represent April) Output:

30

iii. 12 (to represent December) Output:

31

iv. 2 (to represent February) Output:

28

**TASK 4. SWITCH STATEMENT**

The switch statement provides an alternative to the if-else chain for cases that

compare the value of an integer expression to a *specific* value. The switch statement

uses four keywords: switch, case, default and break. The switch

statement syntax is:

**switch**(expression)

{

**case** value1:

statement(s);

**break**;

**case** value2:

statement(s);

**break**;

. . .

. . .

**case** valueN:

statement(s);

**break**;

**default**:

statement(s);

**break**;

}

where expression evaluates to an integral value and value1, value2, …, and

valueN are also integrals.

a. Explain how the four keywords switch, case, default and break work

in the switch statement.

Switch is a selector that takes a single argument parameter. Within the scope of the selector the switch function checks the value of the argument against the value of each keyword case followed by a colon. Arguments that match the case have the following block executed, non matching cases skip their code block and go to check the next case. Once one case is done the switch function immediately tries the next case unless you have a break keyword in your statement. If none of the cases match the argument or no switch statement is performed, the default keyword will execute a block of code that follows it.

b. Rewrite the program ‘*task2.cpp’* using a switch statement, and save it as

‘*task24.cpp’*.

c. Compile and run the program. Compare the output to that of ‘*task2’* to ensure that

it is correct.

***Note: Do the following now if there appears to be ample time, otherwise do it later.***

d. Rewrite the program ‘*task3.cpp’* using a switch statement, and save it as

‘*task34.cpp’*.

e. Compile and run the program, ensuring that you test it thoroughly. Compare the

output to that of ‘*task3’* to ensure that it is correct.

f. What is the main benefit of using switch statements instead of if statements?

It allows you to execute following cases after a condition has been reached

g. When is it **not** appropriate to use switch statements in place of if statements?

It is not appropriate when there are a small number of cases or if it can be summed up more easily using an if else selector.