CS 101: Computer Programming and Utilization

Jul-Nov 2017

Umesh Bellur (cs101@cse.iitb.ac.in)

6: Conditional Execution
About These Slides

• Based on Chapter 6 of the book
  *An Introduction to Programming Through C++*
  by Abhiram Ranade *(Tata McGraw Hill, 2014)*

• Original slides by Abhiram Ranade
  – First update by Varsha Apte
  – Second update by Uday Khedker
Let Us Calculate Income Tax

Write a program to read income and print income tax, using following rules

• If income ≤ 1,80,000, then tax = 0

• If income is between 180,000 and 500,000 then tax = 10% of (income - 180,000)

• If income is between 500,000 and 800,000, then tax = 32,000 + 20% of (income – 500,000)

• If income > 800,000, then tax = 92,000 + 30% of (income – 800,000)

Cannot write tax calculation program using what we have learnt so far
An Even Simpler Problem

• Using the rules given earlier, read in the income of an individual and print a message indicating whether or not the individual owes tax

• Even this simpler problem cannot be done using what we have learned so far

• For completeness, we need
  – Sequence of statements
default textual appearance
  – Repetition of statements
repeat statement
  – Selection of statements
new statement needed: if statement
Outline

• Basic `if` statement
• `if-else` statement
• Most general `if` statement form
• `switch` statement
• Computing Logical expressions
Basic IF Statement

Form:
if (condition) consequent

condition: boolean expression
boolean : Should evaluate to true or false
consequent: C++ statement, e.g. assignment

If condition evaluates to true, then the consequent is executed.
If condition evaluates to false, then consequent is ignored.
Conditions

- Simple condition: \texttt{exp1 relop exp2}
  - \texttt{relop}: relational operator: \texttt{<, \leq, \equiv, >, \geq, \neq}
    - less than, less than or equal, equal, greater than, greater than or equal, not equal
- Condition is considered true if \texttt{exp1} relates to \texttt{exp2} as per the specified relational operator \texttt{relop}
A Better Program for our Simple Problem

```c
main_program {
    float income, tax;
    cin >> income;
    if (income <= 180000)
        cout << "No tax owed." << endl;
    else
        cout << "You owe tax.\n" << endl;
}
// Only one condition check
// Thus more efficient than previous
Program for the Simple Problem

```cpp
main_program {
    float income, tax;
    cin >> income;
    if (income <= 180000)
        cout << "No tax owed" << endl;
    if (income > 180000)
        cout << "You owe tax" << endl;
}
// Always checks both conditions
// If the first condition is true,
// then you know second must be false,
// and vice versa. Cannot be avoided
// using just the basic if statement
```
Flowchart

- Pictorial representation of a program
- Statements put inside boxes
- If box C will possibly be executed after box B, then put an arrow from B to C
- Specially convenient for showing conditional execution, because there can be more than one next statements
- Diamond shaped boxes are used for condition checks
Flowchart of the IF Statement

Previous Statement

Condition

Consequent

True

False

New Statement
A More General Form of the IF Statement

if (condition) consequent else alternate

The condition is first evaluated
If it is true, then consequent is executed
If the condition is false, then alternate is executed
Flowchart of the IF-ELSE statement

Previous Statement

Condition

True
Consequent

False
Alternate

New Statement
Most General Form of the IF-ELSE Statement

\[
\text{if (condition}_1) \text{ consequent}_1 \\
\text{else if (condition}_2) \text{ consequent}_2 \\
\ldots \\
\text{else if (condition}_n) \text{ consequent}_n \\
\text{else alternate}
\]

Evaluate conditions in order
Some condition true: execute the corresponding consequent. Do not evaluate subsequent conditions
All conditions false: execute alternate
Flowchart of the General IF-ELSE Statement (with 3 conditions)

Previous Statement

<table>
<thead>
<tr>
<th>True</th>
<th>False</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Condition</td>
<td></td>
</tr>
<tr>
<td></td>
<td>True</td>
</tr>
<tr>
<td></td>
<td>False</td>
</tr>
<tr>
<td>Condition 2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>True</td>
</tr>
<tr>
<td></td>
<td>False</td>
</tr>
<tr>
<td>Condition 3</td>
<td></td>
</tr>
<tr>
<td></td>
<td>True</td>
</tr>
<tr>
<td></td>
<td>False</td>
</tr>
</tbody>
</table>

Consequent 1
Consequent 2
Consequent 3
Alternate

New Statement
main_program {
    float tax, income;
    cin >> income;
    if (income <= 180000) tax = 0;
    else if (income <= 500000) 
        tax = (income – 180000) * 0.1;
    else if (income <= 800000) 
        tax = (income – 500000) * 0.2 + 32000;
    else tax = (income – 800000) * 0.3 + 92000;
    cout << tax << endl;
}
Tax Calculation Flowchart

Read Income

True

Income <= 180000

tax = 0;

False

Income > 180000

True

Income <= 500000

tax = (income - 180000) * 0.1;

False

Income > 500000

True

Income <= 800000

tax = 32000 + (income - 320000) * 0.2;

False

Income > 800000

tax = 92000 + (income - 800000) * 0.3;

Print Tax
More General Conditions

• \text{condition1} \land \text{condition2} : \text{true only if both true}
  
  Boolean \text{ AND}

• \text{condition1} \lor \text{condition2} : \text{true only if at least one is true}
  
  Boolean \text{ OR}

• \text{! condition} : \text{true if only if condition is false}

• Components of general conditions may themselves be general conditions, e.g.

  \text{!(income < 18000) || (income > 500000)}

• Exercise: write tax calculation program using general conditions wherever needed
Remark

The consequent in an `if` statement can be a `block` containing several statements. If the `condition` is true, all statements in the block are executed, in order.

Likewise the alternate

Example: If income is greater than 800000, then both the statements below get executed

```c++
if (income > 800000) {
    tax = 92000 + (income – 800000)*0.3;
    cout << “In highest tax bracket.”; 
}
\n: Newline character. Another way besides `endl`
Logical Data

• We have seen that we can **evaluate** conditions, combine conditions
• Why not allow storing the results (**true** or **false**) of such computations?
• Indeed, C++ has data type **bool** into which values of conditions can be stored
• The type **bool** is named after George Boole, who formalized the manipulation of logical data
• An **int** variable can have $2^{32}$ values, a **bool** variable can have only two values (**true/false**)
The Data Type \texttt{Bool}

\begin{verbatim}
bool highincome, lowincome;
Declares \texttt{variables} highincome and lowincome of type \texttt{bool}
highincome = (income > 800000);
bool fun = true;
Will set highincome to true if the variable income contains
value larger than 800000
boolean variables which have a value can be used
wherever \texttt{conditions} are expected, e.g.
if (highincome)
    tax = ...
\end{verbatim}
Example: Determining If a Number is Prime

- Program should take as input a number x (an integer > 1)
- Output **Number is prime** if it is, or **number is not prime** if it is not
- Steps:
  - For all numbers 2 to x-1, check whether any one of these is a factor of n
    - These are x-2 checks
  - If none, then number is prime
Let's try using the accumulation idiom with a boolean variable.

Be careful of $=$ vs $==$
Example...Prime

main_program {
    int x; cin >> x; // read x 4534534536
    int i = 2;       // first factor to check;
    bool factorFound = false; // no factor found yet;
    repeat (x-2) {
        factorFound = factorFound || ((x % i) == 0);
    // Remainder is 0 when x is divisible by i
        i++;
    }
    if (factorFound) cout << x << " is not prime"
    << endl;
}
Remarks

• Conditional execution makes life interesting
• Master the 3 forms of if
• Exercise: write the tax calculation program without using the general if and without evaluating conditions unnecessarily. Hint: use blocks
• You can nest if statements inside each other: some pitfalls in this are discussed in the book
SAFE quiz

• What is printed by this code snippet: "int x=3,y=1; {int x=4; {x = x+2;} y=x;} cout << (x+y);}"

• What does this code print? "int i=0,s=0; repeat(3) {if (i%2==0) s += i; else s += 2*i; i++;} cout << s;"

• What does this program print? "unsigned int x,c=0; cin>>x; repeat (32) {if (x%2==1) c++; x = x/2;} cout << c;"

• What does this program print? "unsigned int x,c=0; cin>>x; repeat (32) {if (x%2==1) c++; x = x/2;} cout << c;"