More general form of conditions

Sometimes we might want to do something if two conditions are true, or one of two conditions is true...

Compound conditions:

- "AND" :***condition1 && condition2*** : true only if both true.
- "OR" :***condition1 || condition2*** : true only if at least one is true.
- "NOT": ***! condition*** : true if only if ***condition*** is false.

Components of compound conditions may themselves be compound conditions,

e.g. ***condition1 && (condition2 || condition3)***

Example 1

```
if ((income >= 180000) && (income <= 500000))
   tax = (income – 180000) * 0.1;
```

Example of execution

- ***income = 1000*** : Condition is false, consequent not executed.
- ***income = 200000 :***
  tax = (200000 – 180000) * 0.1 is executed.
- ***income = 600000*** : Nothing happens.

Note

Same condition may be expressed in many ways.

(income >= 180000) is same as ***!(income < 180000)***

(income <= 500000) is same as ***!(income > 500000)***

Previous statement can be written as

```
if (!(income < 180000) && !(income > 500000))
   tax = (income – 180000) * 0.1;
```

Another example

- Consider rectangle lying between lines x=0, x=10, y=50, y=70.
- Let (X,Y) denote the coordinates of a point.
- The point is inside the rectangle if $0 \leq X \leq 10$, and $50 \leq Y \leq 70$
- To check this we will write the condition:

```
(0 >= X && X<= 10 && Y >= 50 && Y<= 70)
```

- Do not write `0 <= X <= 10`

What we discussed

- More general ways of specifying the conditions.
- Note: ! Has higher precedence than && which has higher precedence than ||

!P && Q || R is same as ***((!P) && Q) || R***

Next: A somewhat large example based on what we have learned so far.

🌺

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.
- ***bool*** is named after George Bool, who formalized the manipulation of conditions/logical data.

The data type bool

bool highincome, lowincome;

- Defines variables ***highincome*** and ***lowincome*** of type ***bool***.

highincome = (income > 800000);
bool fun = true;

- Will set ***highincome*** to true if the variable ***income*** contains value larger than 800000.
- ***true*** and ***false*** : boolean constants.
- boolean variables which have a value can be used wherever “conditions” are expected, e.g.

if(highincome) tax = …

Exercise: write a program to test if a given number n is prime.

- How will you do this manually?
  - Eratosthenes’ sieve
  - We are required to “remember” all the primes determined till n.
  - So far we have no way of doing this
- Can we do something less efficient, but without requiring us to remember too many things?
  - Check if any of the numbers from 2 to n-1 divide n.

Solution

```
#include <simplecpp>

main_program{
  int n, divisor=2; cin >> n;
  bool divisorFound = false;  // no divisor found for n so far
  // check if divisor divides n as it varies from 2 to n-1
  // if divisor divides n, set divisorFound = true
  repeat(n-2){
    if(n % divisor == 0) divisorFound = true;
    divisor = divisor + 1;
  }
  if(!divisorFound) cout <<"Prime.\n";
  else cout <<"Composite.\n";
}
```

Exercise

Execute the program mentally for n = 100.

- What answer does it produce?
- Are you happy with how the program executes?

Summary

- Conditional execution makes life interesting.
- 3 forms of if.
  – You can nest if statements inside each other: some pitfalls in this are discussed in the book.
- Compound conditions
- Logical data

Try the exercises at the end of the book.

🌺🌺

A different way to control the turtle

- We will write a program which reads commands from the user, and accordingly controls the turtle.
  - ‘f’ : turtle should execute forward(100).
  - ‘r’ : turtle should turn right(90).
  - ‘l’ : turtle should turn left(90).
  - Stop after 100 commands are executed.

The program

```
main_program{
  char command;
  turtleSim();

  repeat(100){
    cin >> command;
    if (command == 'f') forward(100);
    else if (command == 'r') right(90);
    else if (command == 'l') left(90);
    else cout << "Not a proper command, "
              << command << endl;
  }
}
```

Demo

Exercise

- Write a program that reads a number and prints 1 if the number is a multiple of 5 but not of 3, and otherwise prints 0. Write this in as many different ways as possible.
  - Using only simple conditions, e.g. expression 1 == expression 2, but with if statements nested inside each other.
  - Using a single if-then-else statement with a compound condition.