CS 101: Computer Programming and Utilization

Jul-Nov 2017

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

Chapter 6 : Conditional Execution

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
  - **Repetition** of statements
  - **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

The `IF` Statement

Form:

```
if (condition) consequent
```

condition: boolean expression

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: `exp1 relop exp2`

relop : relational operator: `<, <=, ==, >, >=, !=`

Condition is considered true if exp1 relates to exp2 as per the specified relational operator relop

Program for the Simple Problem

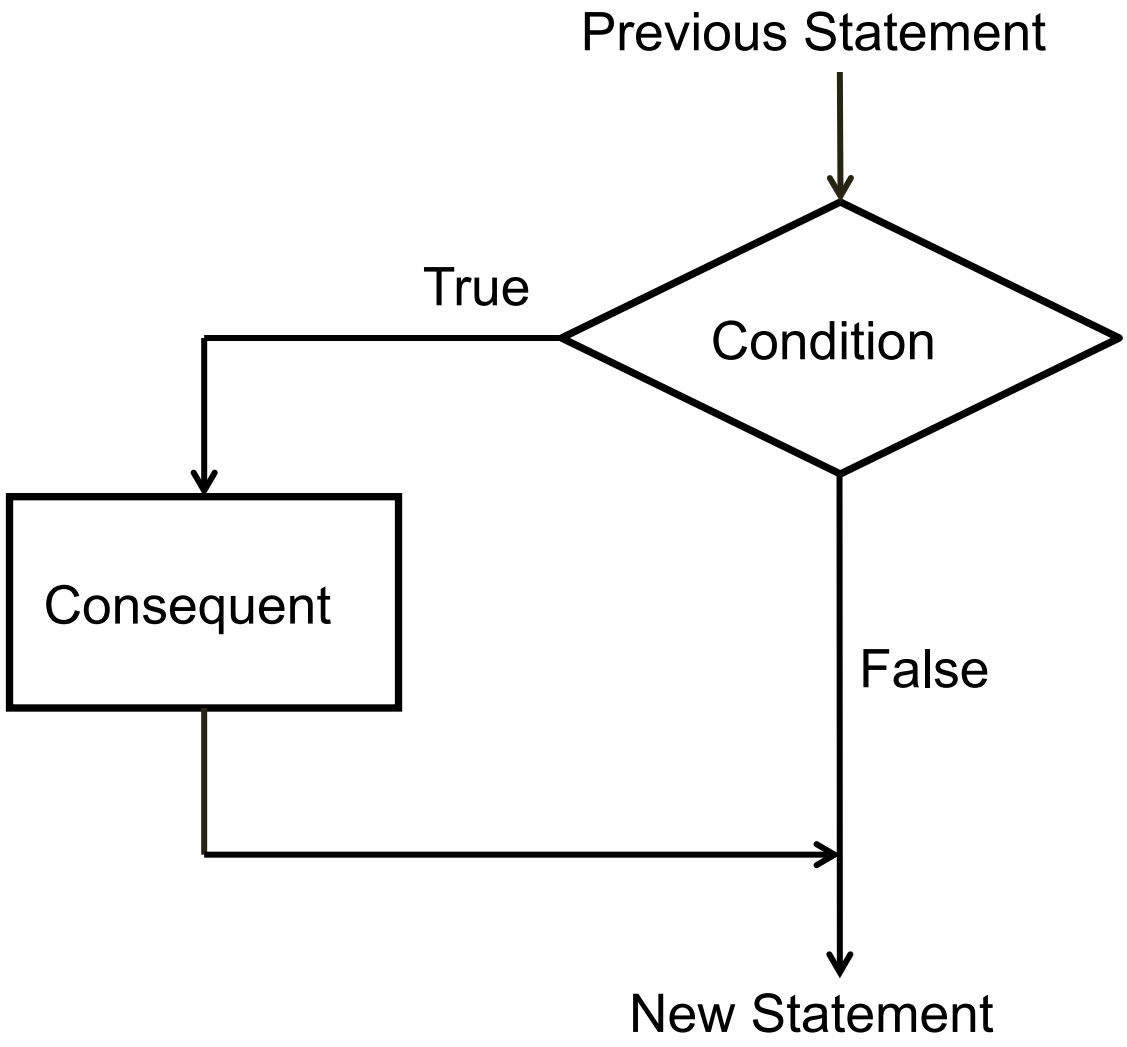
```
main_program {
    float income;
    cin >> income;
    if (income <= 180000)
        cout << "No tax owed" << endl;
    if (income > 180000)
        cout << "You owe tax" << endl;
}
```

Checks both conditions separately even though they are mutually exclusive.

Flowcharts – tools for program visualization

- Pictorial representation of a program using **Boxes** and **Arrows**.
  - 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



A More General Form of the IF Statement

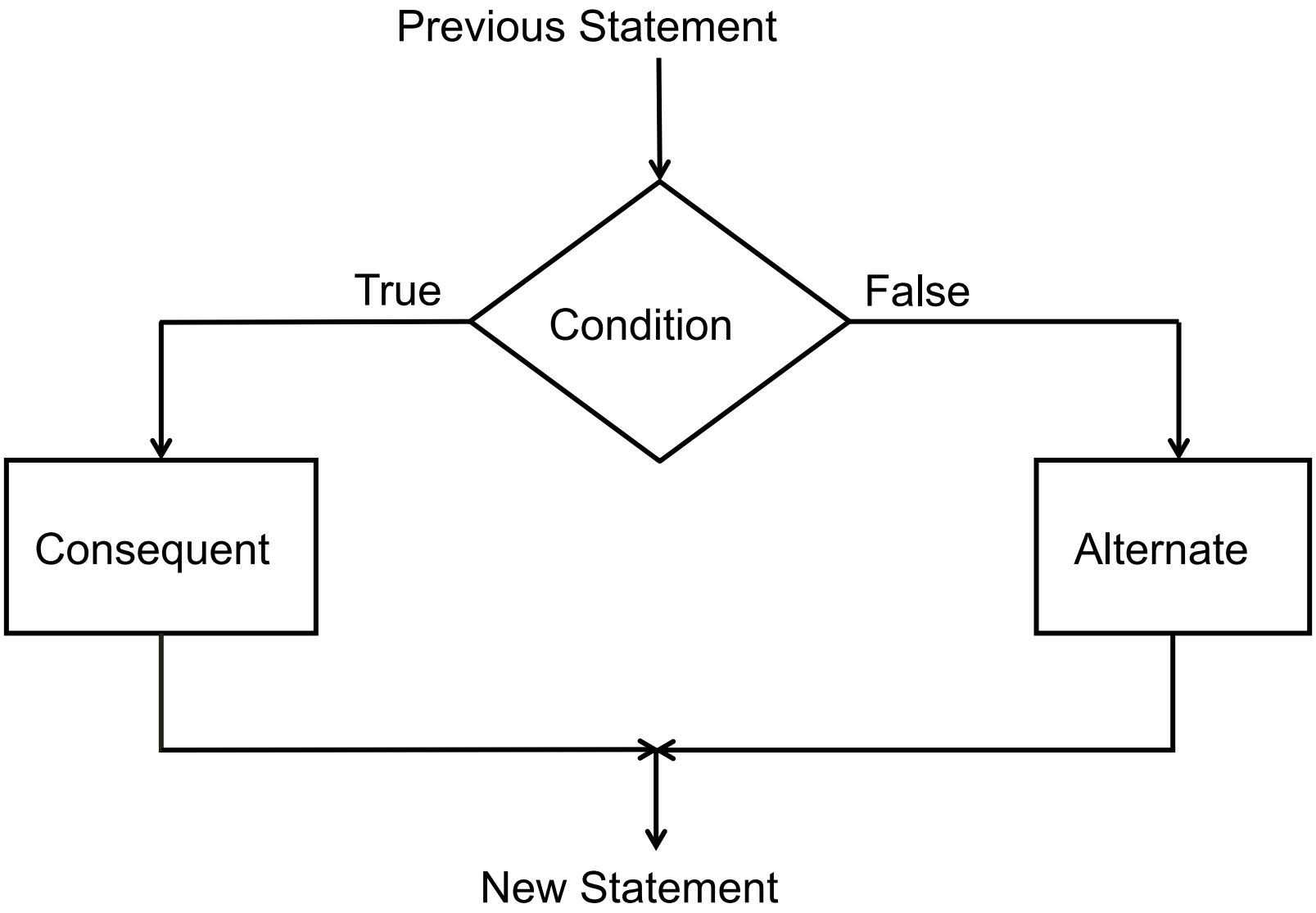
```
if (condition) consequent else alternate
```

The condition is first evaluated

If it is true, then consequent is executed

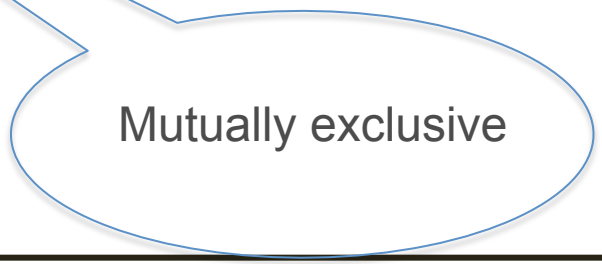
If the condition is false, then alternate is executed

Flowchart of the IF-ELSE statement



A "better" Program for our Simple Problem

```
main_program {
   float income, tax;
      cin >> income;
   if (income <= 180000)
      cout <<  "No tax owed."  << endl;
   else
      cout <<  "You owe tax."  << endl;
}
```



Most General Form of the IF-ELSE Statement

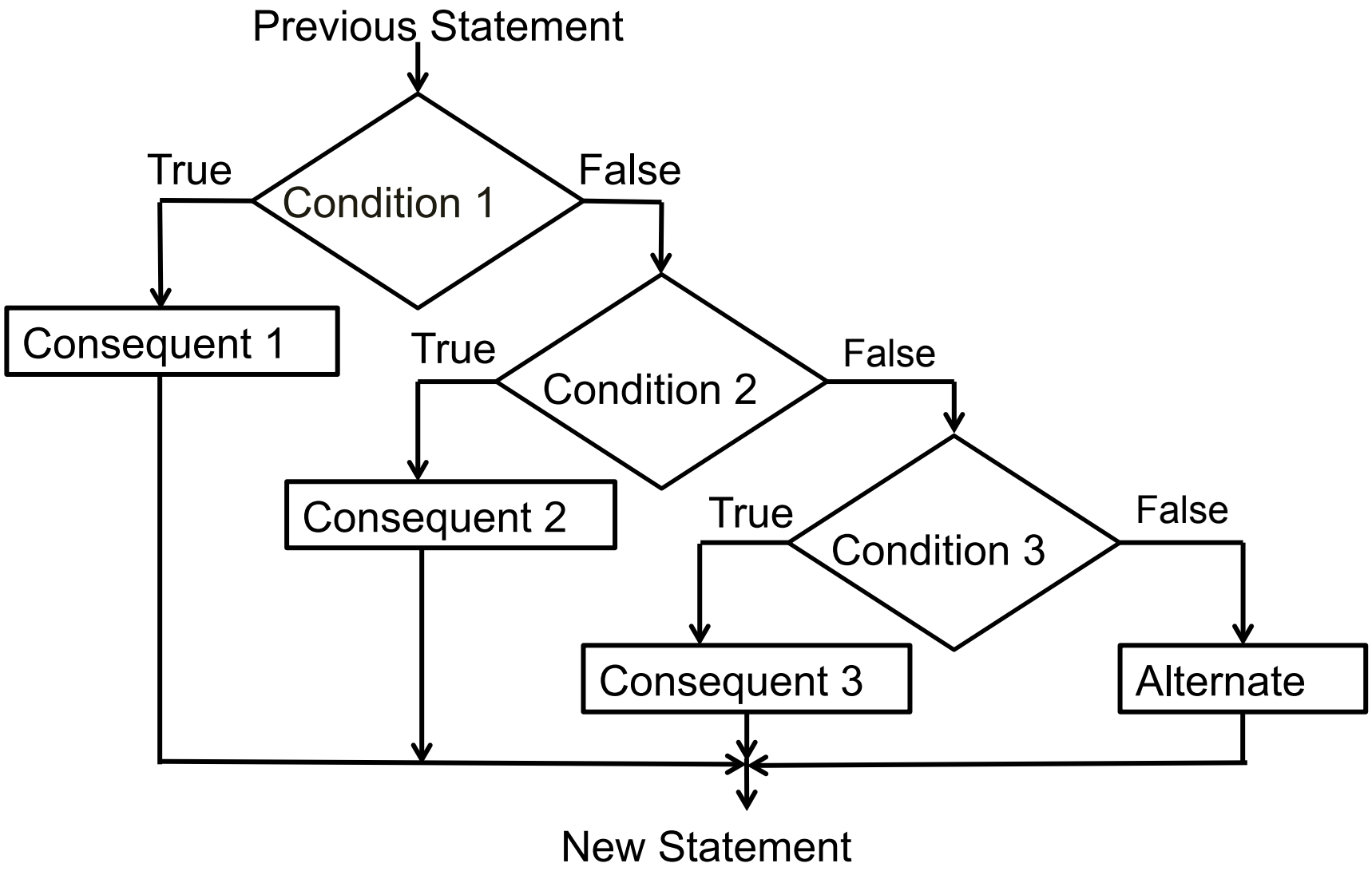
```
if (condition1) consequent1
else if (condition2) consequent2
…
else if (condition-n) consequent-n
else alternate
```

Evaluate conditions ***in order***

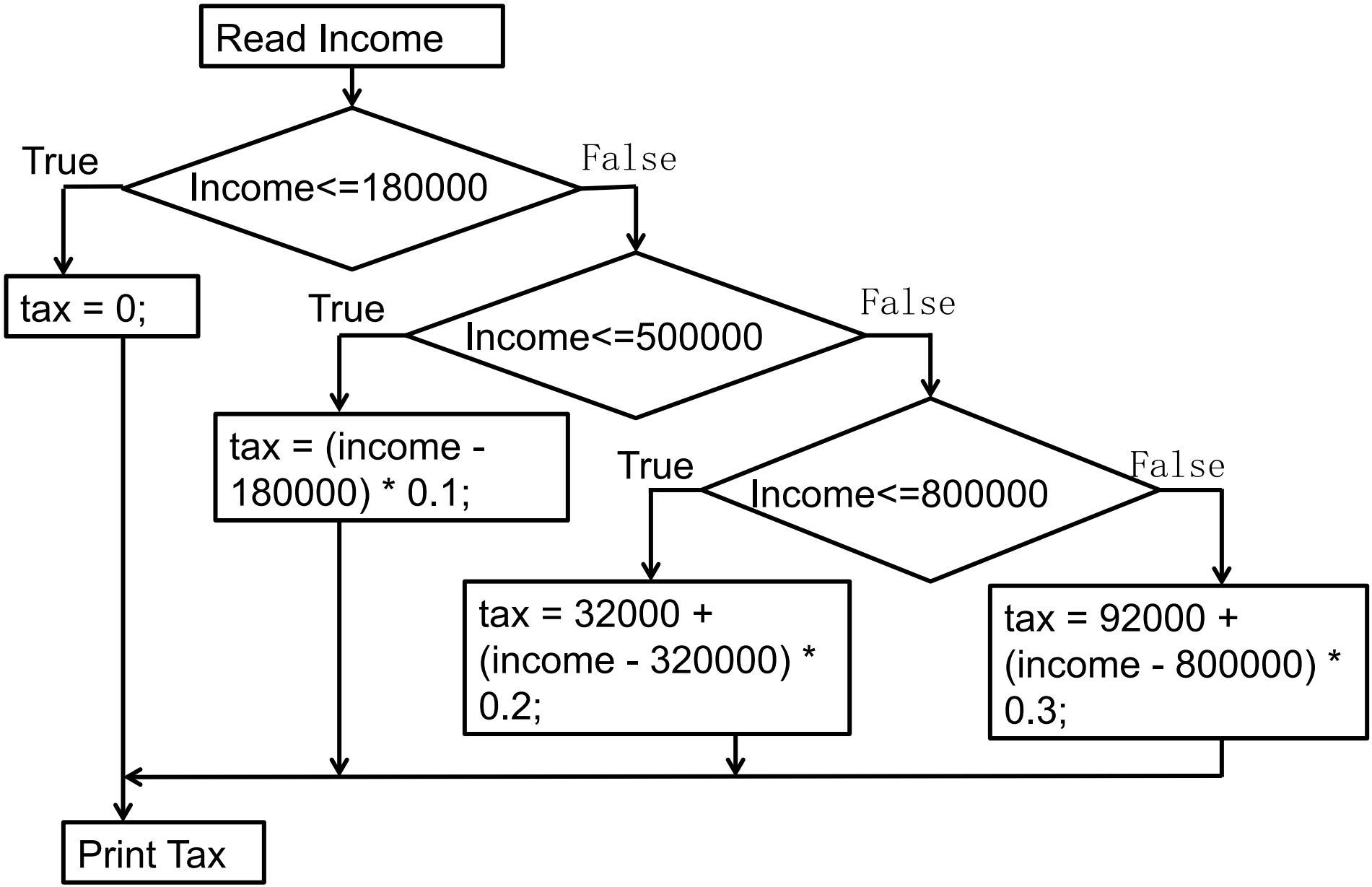
Some condition true => execute the corresponding consequent. Do not evaluate subsequent conditions

All conditions false => execute alternate

Flowchart of the General IF-ELSE Statement (with 3 conditions)



Tax Calculation Flowchart



Tax Calculation Program

```
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;
}
```

Complex conditions – conjunctions, disjunctions

- `condition1 && condition2` : true only if both true

  `condition1 || condition2` : true only if at least one is true

  `! condition` : true if only if condition is false

- Components of a complex conditions may themselves be complex conditions, e.g.

  ```
  !((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

```
if (income > 800000) {
    tax = 92000 + (income – 800000)*0.3;
    cout << “In highest tax bracket.\n” ;
}
```

\n : Newline character. Another way besides endl

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

Example...Prime

Let's try using the accumulation idiom with a boolean variable in a condition.

Be careful of = vs ==

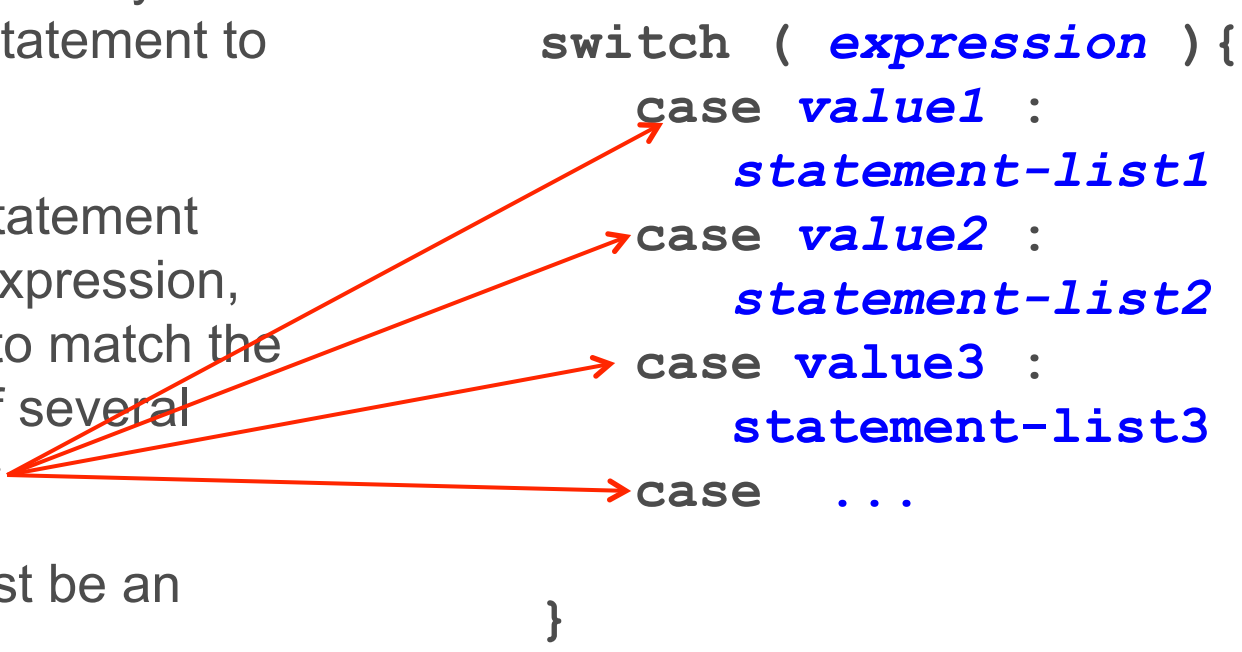
Example...Prime

```
main_program {
    int x;  cin >> x;       // read x
    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;
}
```

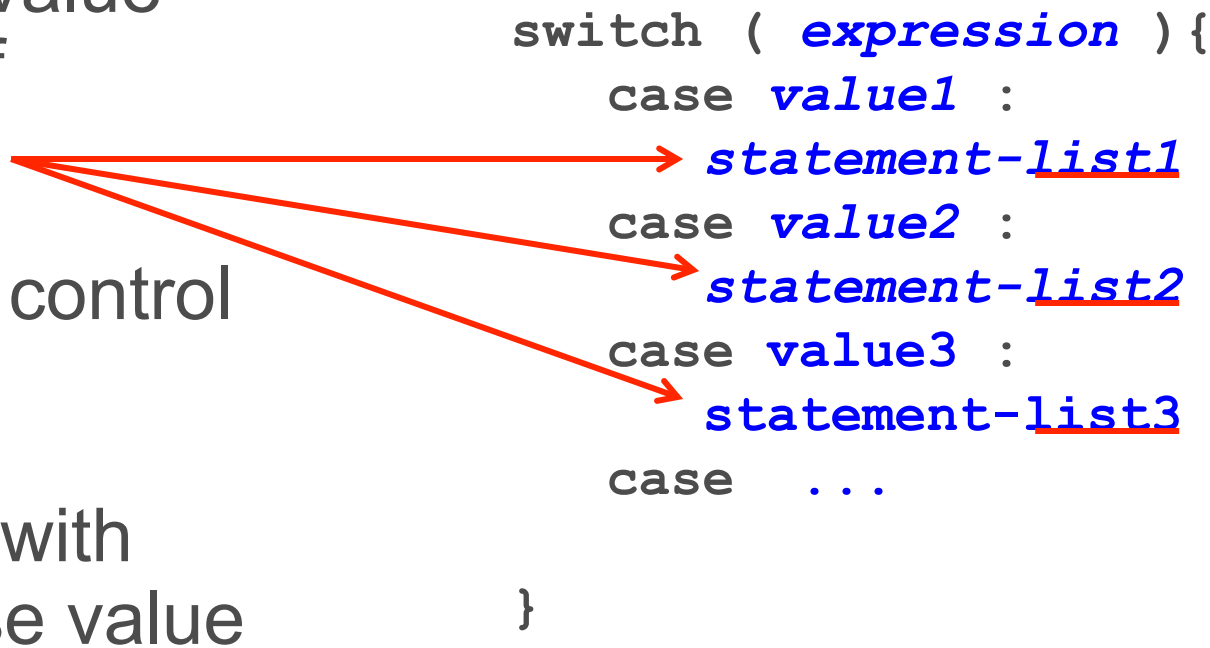
The Switch Statement

- The *switch statement* provides another way to decide which statement to execute next
- The `switch` statement evaluates an expression, then attempts to match the result to one of several possible *cases*
- The match must be an ***exact*** match.



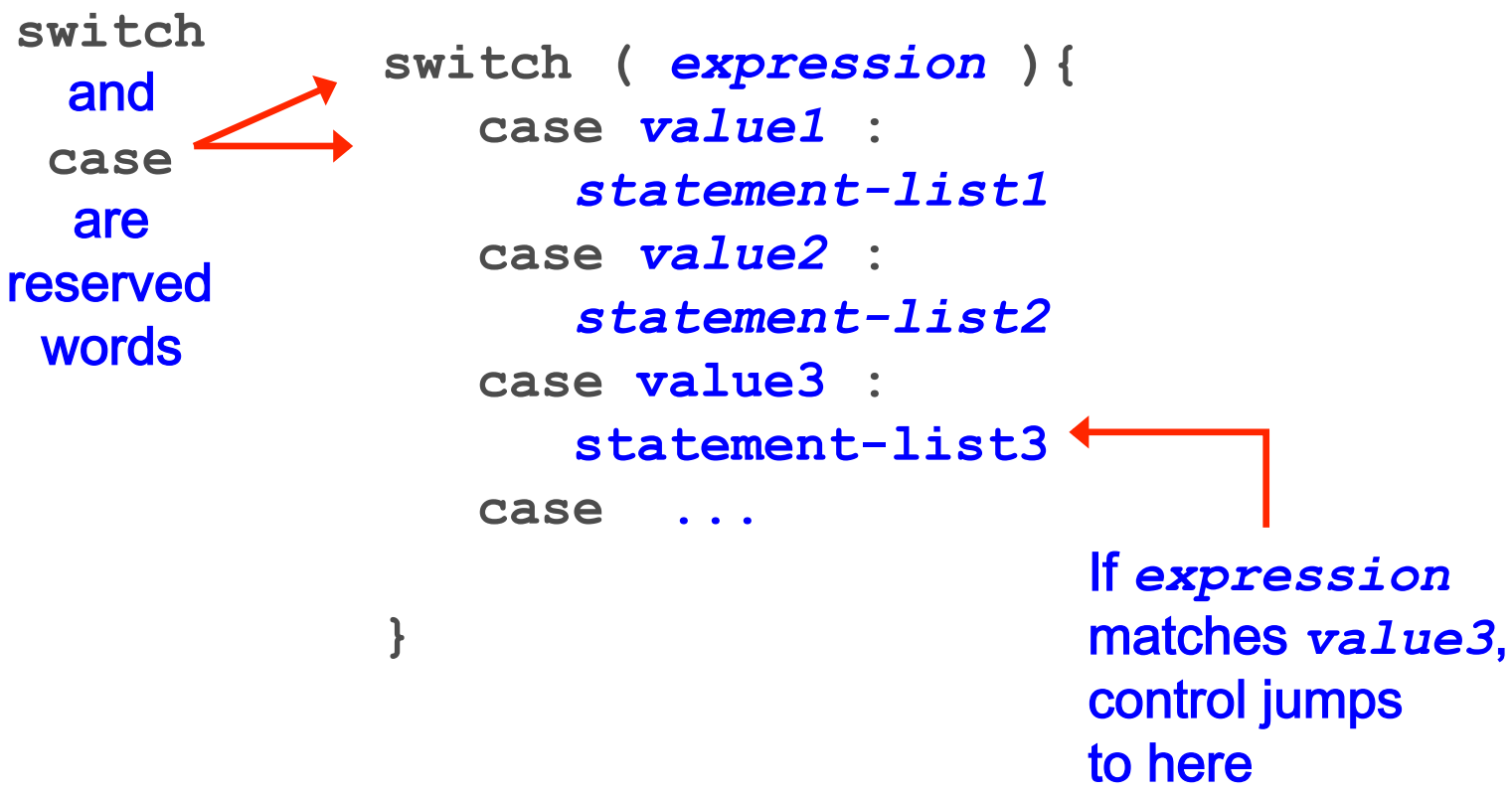
The Switch Statement

- Each case contains a value and a list of statements
- The flow of control transfers to statement associated with the first case value that matches



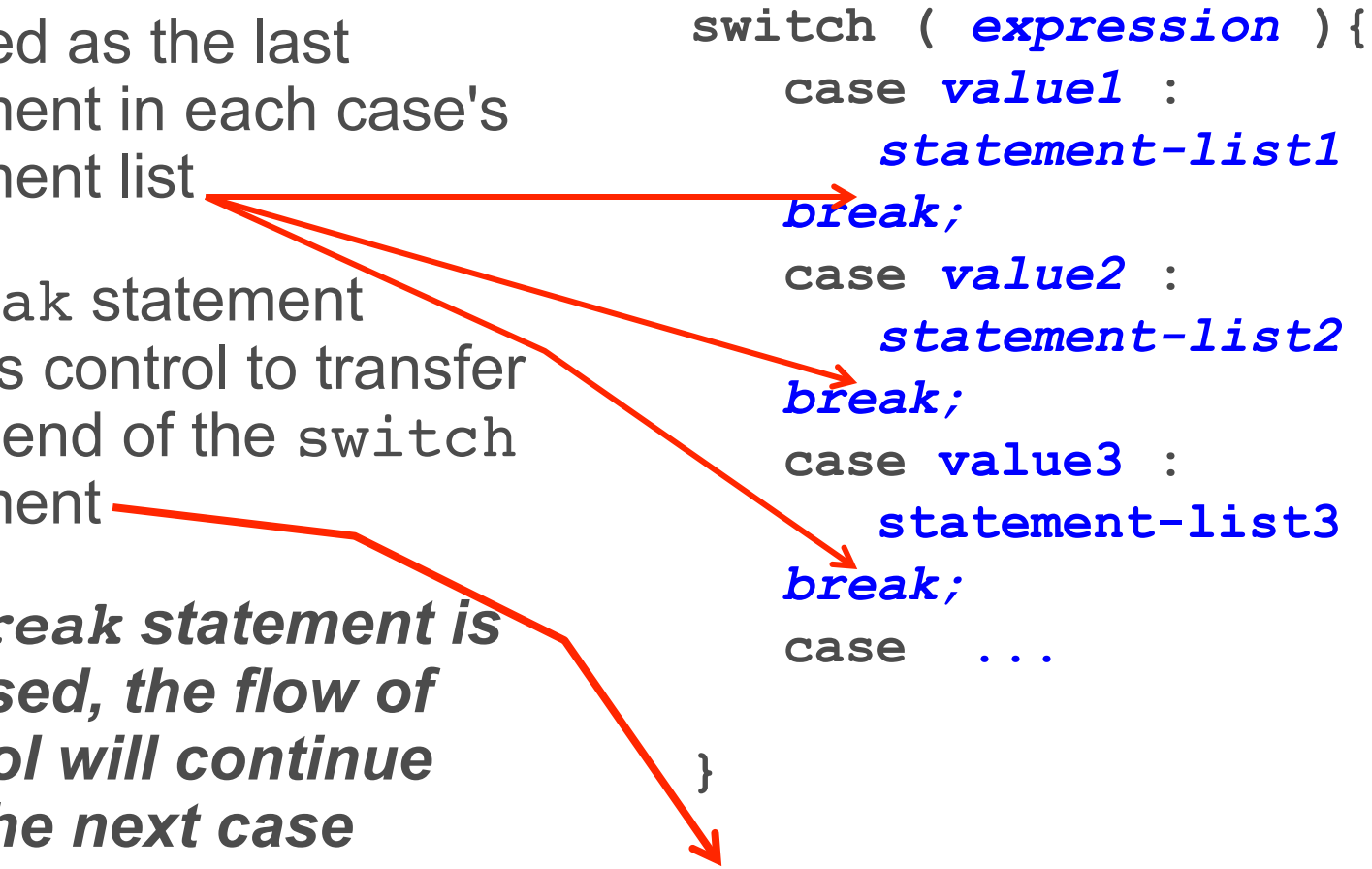
Switch - syntax

- The general syntax of a `switch` statement is:



The break Statement

- The *break statement* can be used as the last statement in each case's statement list
- A `break` statement causes control to transfer to the end of the `switch` statement
- ***If a `break` statement is not used, the flow of control will continue into the next case***



Switch Example

- Examples of the switch statement:

```
switch (option){
   case 'A':
      aCount++;
      break;
   case 'B':
      bCount++;
      break;
   case 'C':
      cCount++;
      break;
}
```

Switch – no breaks!!!

- Another Example:

```
switch (option){
   case 'A':
      aCount++;
      break;
   case 'B':
      bCount++;
      break;
   case 'C':
      cCount++;
      break;
}
```

```
switch (option){
   case 'A':
      aCount++;
   case 'B':
      bCount++;
   case 'C':
      cCount++;
}
```

Switch - default

- A `switch` statement can have an optional *default case*
- The default case has no associated value and simply uses the reserved word **`default`**
- If the default case is present, control will transfer to it if no other case value matches
- If there is no default case, and no other value matches, control falls through to the statement after the switch

The switch Statement

- Switch with default case:

```
switch (option){
   case 'A':
      aCount++;
      break;
   case 'B':
      bCount++;
      break;
   case 'C':
      cCount++;
      break;
   default:
      otherCount++;
   break;
}
```

To Switch or not to Switch

- The expression of a `switch` statement must result in an ***integral type***, meaning an integer (`byte`, `short`, `int`, `long`) or a `char`
- It **cannot** be a `boolean` value or a floating point value (`float` or `double`)
- The implicit boolean condition in a `switch` statement is **equality**
- **You cannot perform relational checks with a `switch` statement**

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;