CS 101: Computer Programming and Utilization

08-C++ Control Flow

Instructor: Sridhar Iyer
IIT Bombay

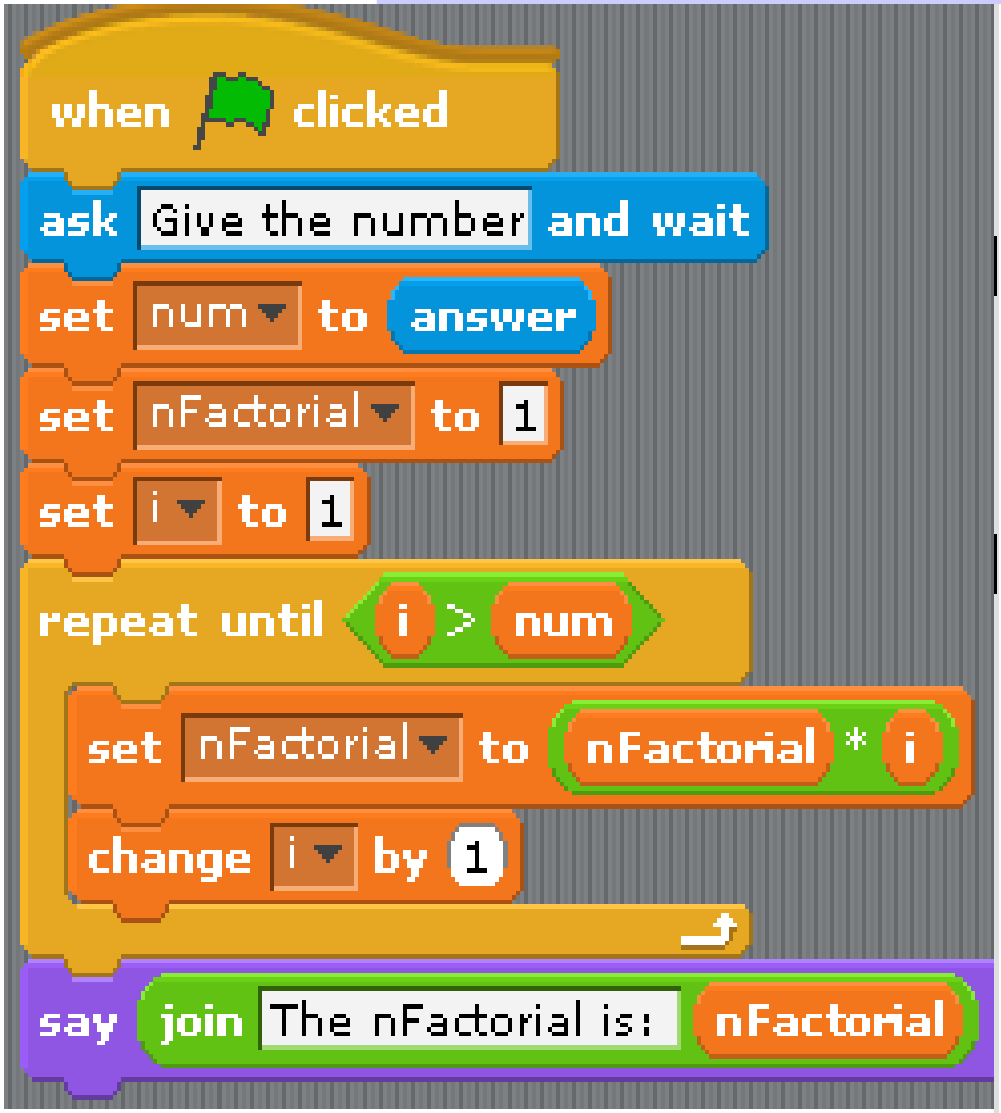
Activity: Write a program

to compute factorial of a given number

Write it in at least 2 of the following:

- psuedo-code
- Scratch
- C++

- Input N from keyboard; Initialize M to 1;
- Repeat M = M*i, where i goes from 1 to N
- Output M to display




```
#include <iostream>
using namespace std;
int main() {
int num, nFactorial = 1;
    cout<< "give the value of num: "; cin >> num;
    for (int i = 1; i <= num; i++) {
        nFactorial *= i;
    }
    cout<< " nFactorial is: " << nFactorial <<endl;
return 0;
}
```

Modify the program to

Calculate factorial for many numbers, taking each one from the input

Then do testing – favourable cases, boundary conditions

Run: demo08-factorial.cpp and its modification. Also demo08-factorial.sb

C++ constructs seen so far

- Including libraries, namespaces:
  - #include<iostream>; using namespace std
- Functions: main()
- Data types and variable declarations:
  - int n; float nFactorial; char flag = 'y';
- Arithmetic operations, expresions: 5*(F-32)/9
  - Boolean values and operations: (x && y)
  - Type conversions: Experiments in lab 04
- Assignment statement: c = 5*(F-32)/9;
- Compiling and executing a C++ program

More C++ constructs seen today

- Compound assignment: nFactorial *= i;
- Increment/Decrement: i++ (different from ++i)
- Condition blocks: if (flag != 'y') { … };
- Conditional expressions: c = (a < b)? a : b;
    - If (a < b) c = a; else c = b;
- Loops: for (i=0; i <= n; i++) { … };
- Nested loops: while () { ... for () {...}; ... };
- Infinite loops and break: while (1) {… break};

Note: how did we get here

- Rather than my describing constructs in C++ , you have learnt them by directly using them
  - that too, in a short span of time!
  - For descriptions of these constructs, read the notes at the end of these slides

- Two reasons for this achievement of yours:
  - We wrote pseudo-code to first get the logic of the program right, without worrying about syntax
  - We wrote Scratch programs to develop familiarity with the logic of many commonly used constructs, so transitioning to C++ syntax is easier

Think-Pair-Share: swapping two numbers

```
float x = 5, y = 11;
float temporary = x;
x = y;
y = temporary;
```

Can you swap without using a temporary variable?

Hand Execution

Or

Single Stepping

x	y	temporary
5	11	
5	11	5
11	11	5
11	5	5

Another problem: Try this solo

- How many times must we divide a number by 10 until the result goes below 1?

- Given input x
- Output should be numDivs –
  - the number of times you had to divide x by 10 before you got the result to go below 1
- Do this on your own:
  - Assume that this is a quiz question and write
  - First get the logic in pseudo-code, then C++

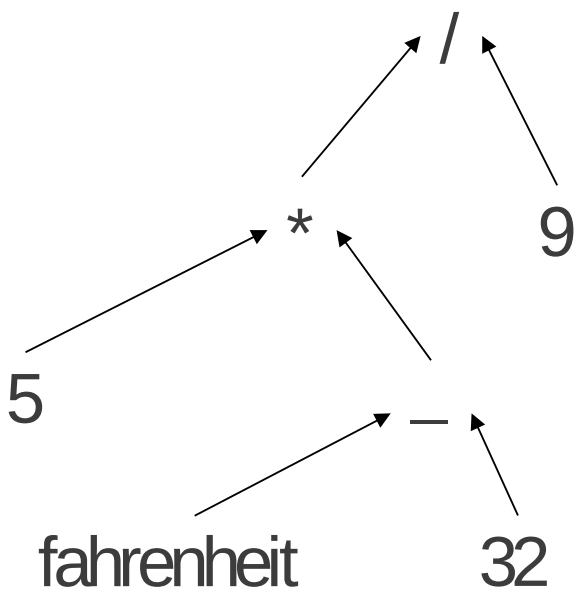
Announcements - Quiz1

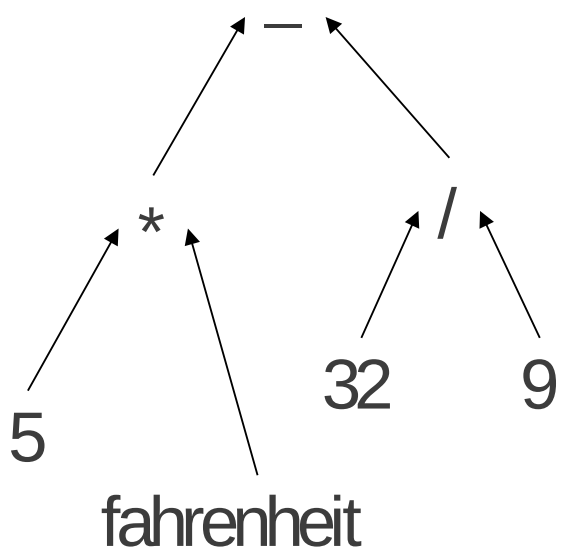
- Portion: All of what we have done in Scratch
  - C++ programming is not included for Quiz1
  - 1 hour duration – Friday, 8th Feb 2013, 8:25 AM
- Questions: predict the output of programs, debug given programs, write programs for given problems, in addition to conceptual questions
- Memory aid: You can bring one page of notes **in your own handwriting** (A-4 size sheet, 2 sides).
  - No printouts, no photocopies.
  - All electronic devices should be switched OFF.
- Re-Quiz: Assume that there is no re-Quiz.
  - Even if you have a “valid reason” and I agree to a re-quiz, it will be much tougher than this Quiz itself. So dont bunk!

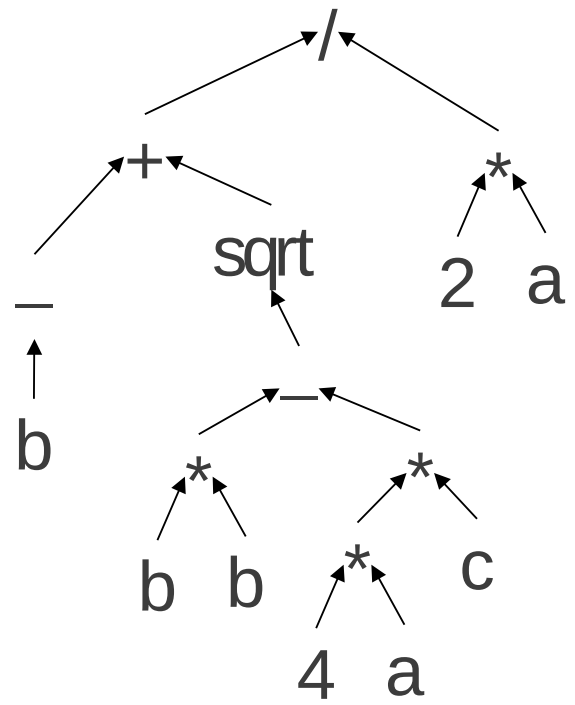
Reading Notes

Arithmetic expressions

- `5 * (fahrenheit – 32) / 9`
- `5 * fahrenheit – 32/9`
- `(- b + sqrt(b*b – 4*a*c)) / (2*a)`
  - `sqrt` is a library function like `main` we already saw
- Operator precedence and expression trees







Logical expressions

- Compare numeric expressions
  - `5 < 13, 1e5 >= 2e6`
  - `a == b+1` (note the double equals), `c != d`
- Each expression is true or false
  - Internally represented as integers 1 and 0
- Combine using *and*, *or*, *not*
  - `(a == b+1) && (c != d) || !(e <= f)`
- Operator precedence like with numbers
- Logical expressions used to control the execution of statements

Assigning values to variables

- Lhs = Rhs
- Lhs is a variable name
  - Later we will consider arrays, pointers etc.

- Rhs is an expression compatible with the type of the lhs
  - `centigrade = 5*(fahrenheit – 32)/9;`

- Assignment statement has value = rhs
  - Lets us cascade `x = y = z+1;`

Compound assignment

```
/* read two numbers from cin, print
  sum of their squares */
int sum = 0, num;
cin >> num;
sum += (num * num);
cin >> num;
sum += (num * num);
cout << sum << endl;
```

How about

```
int va = 5;
int vb = (va += 2);
```

"expression with side effect" - va is modified

Increment/decrement

- Special case of compound assignment
- Syntax: `++va` and `vb++`
- `++va` means increase `va` by one and then access the incremented value

- `vb++` means access the current value of `vb` and then increment it before any further access
  - May be slightly inefficient because the old value must be remembered
  - `vb = 2 * (va++);`

- Similarly `va--` and `--vb`

Statement block

- A simple statement assigns a variable the value of an expression
- A block looks like `{statement;statement;…statement;}`
- 0 or 1 statement allowed for uniformity
- Walk down the list executing one statement after another
- Effect of each statement on memory completes before next executed

```
{ float x = 5, y = 11;
  float temporary = x;
  x = y;
  y = temporary;      }
```

- Note on **scope**: Outside {…} cannot use variables declared inside

If-then-else

- Store in variable b the absolute value of variable a

```
int a, b;
cin >> a;
if (a >= 0) {
  b = a;
}
else {
  b = -a;
}
```

- Store in variable c the smaller of the values of variables a and b

```
int a, b, c;
cin >> a >> b;
if (a < b) {
  c = a;
}
else {
  c = b;
}
```

- Else part is optional
- Cascades/nests allowed
- Statement blocks also optional but best used

Example: withdrawing money from bank

```
cin >> deduct;
if (deduct > balance) {
  cout << “Account overdrawn\n”;
}
else {
  cout << “Successfully withdrawing ”
 << deduct << endl;
  balance -= deduct;
  // emit paper money
}
```

Curly brackets

- You can also write then or else parts without curly brackets, but this could be dangerous
- Best to always use curly brackets even if not needed

```
int m = 5;
int n = 10;
if (m >= n)
  ++m;
  ++n;
cout << "m=" << m << "n=" << n << endl;
```

Increment of n happens outside the if-then action, which only increments m

Conditional expression

- Format: `cond ? ifExpr : elseExpr`
- Earlier examples rewritten
  - `b = (a > 0)? a : -a;`
  - `c = (a < b)? a : b;`
- If in doubt, use (parens) to make sure expression tree is correct
- Use sparingly, to avoid errors
- Nesting quickly gives unreadable code:

```
(a > 0)? a : ((-a < b)?
    100+c : c-100)
```

Iteration: Approximating the logarithm

- How many times must we divide a number x by 10 until the result goes below 1?
- `while (condition) statementOrBlock`

Prolog

Loop body

Epilog

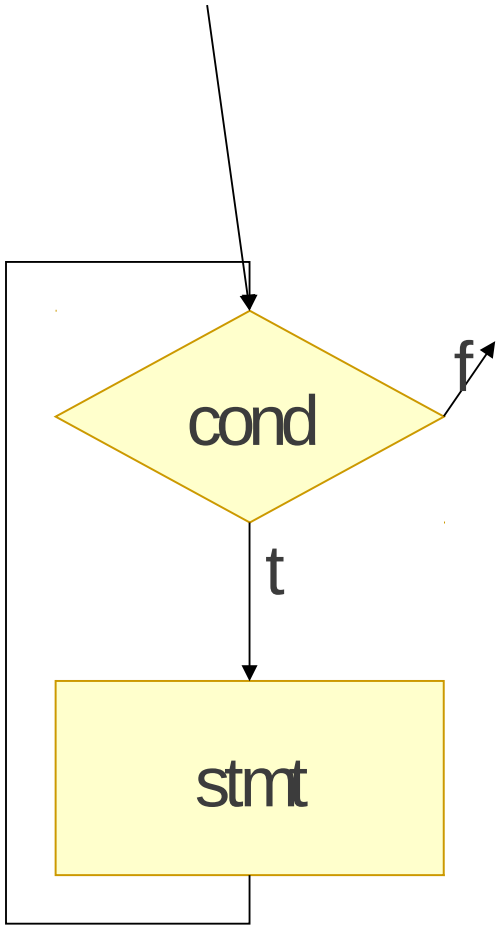
```
float x;
cin >> x;
int numDivs = 0;
while (x > 1) {
  x = x / 10;
  numDivs = numDivs + 1;
}
cout << numDivs;
```

Another shorthand:
`x /= 10;`

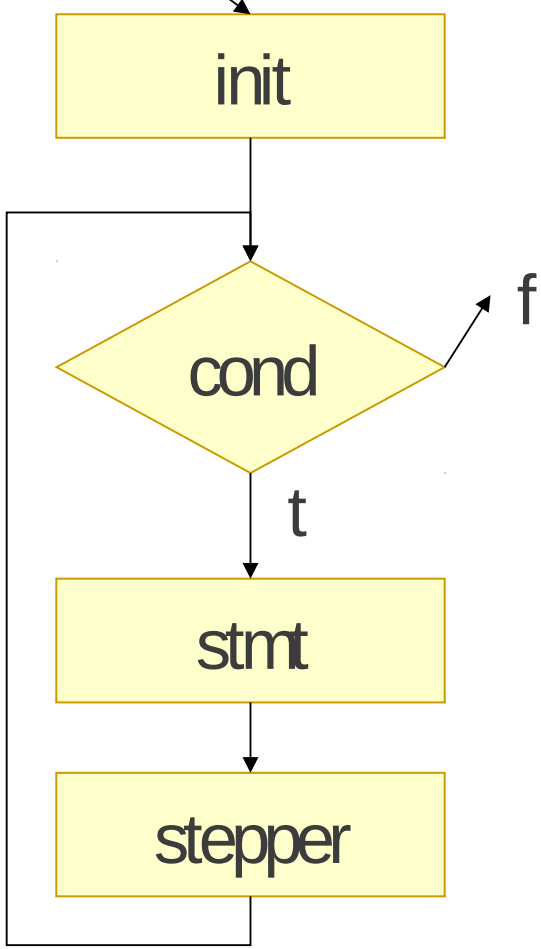
More shorthands:
`numDivs += 1;`
or
`++numDivs;`

While and for

while (cond) { stmt }



for (init; cond; stepper) { stmt }



The infinite loop

```
while (true) {
  cout << “I will lecture clearly and slowly\n”;
  cout << “I will remain quiet in class\n”;
}
```

Insanity: doing the same thing over and over again and expecting different results.

Albert Einstein

Are inputs in increasing order?

- Keep reading input number for ever, and check if it is greater than the previous number read

```
#include <iostream>
#include <limits>
using namespace std;
double prev = numeric_limits<double>::min();
while (true) {
  double cur;
  cin >> cur;
  if (cur <= prev) {
    cout << “Not in increasing orde
  }
  prev = cur;
}
```

How do we get out of the infinite loop?

`break` and `continue`

- Sometimes, we need to exit from a loop in an unexpected way
- Or, in the middle of the loop body, we may want to abandon that iteration and start on the next iteration
- The `break` and `continue` statements provide these capabilities
- Use sparingly, only in a clean, well-lighted area

More notes

- See the cpp-tutorial posted on Moodle
  - Cpp resources folder
- See any textbook on C++
  - Ranade's book, softcopy posted on Moodle
  - Cohoon
- Many websites have good C++ tutorials
  - www.cplusplus.com