Numbers in C++

Abhiram Ranade

Topics

- Reasoning about programs:
  - Loop invariants
- Representing characters
- Some of the operators in C++

Program to find n!

```
main_program{
  int n; cin >> n;
  int fac = 1, i = 1;
  repeat(n){
     fac = fac * i;
     i = i + 1;
  }
  cout << fac;
}
```

Can we be sure that this program is correct?

- Some people wondered when we wrote the program “Is this program computing n! or (n-1)! or even (n+1)!?”
- Such confusion is natural.
- How can it be avoided?

Invariants

- Literal meaning: “Quantity that does not change”
- Invariants in physics: Conservation laws. Total mass is constant before and after ...
- Invariants in programming: any formal statement about values taken by variables in a program.

Example of an invariant

```
main_program{
  int n; cin >> n;
  int fac = 1, i = 1;
  repeat(n){ // On t^th entry:
                    // i = t, fac = (t-1)!
     fac = fac * i;
     i = i + 1;
  }
  cout << fac;
}
```

Loop invariants

- What is the value of the variables when control enters a loop for the t^{th} time? State as a function of t.
- Used for explaining how the program works. Will help you to write correct programs.
- Sometimes invariants are “obvious”, but sometimes they must be proved.

Proving Invariants

- Mathematical induction.
- Base case: Is the invariant true on first entry?
  - Argue by examining the code before the loop.
- Induction step: Assume they are true on t^{th} entry. Then prove true for $t+1^{th}$ entry.
  - Argue by examining the code in the loop.

Is this true on entry?

```
main_program{
  int n; cin >> n;
  int fac = 1, i = 1;
  repeat(n){ // On t^th entry:
                    //i = t, fac = (t-1)!
     fac = fac * i;
     i = i + 1;
  }
  cout << fac;
}
```

i = 1 = t,
fac = 1 = 0! = (t-1)!
Base case proved.

Induction step

```
main_program{
  int n; cin >> n;
  int fac = 1, i = 1;
  repeat(n){    // On t^th entry: i = t,
                // fac = (t-1)!
     fac = fac * i;
     i = i + 1;
  }
  cout << fac;
}
```

Required: on t+1th entry,
i = t+1, fac = t!

What happens during iteration t?
fac = fac * i = (t-1)! * t = t!
i = i + 1= t + 1
Exactly what is needed on t+1th entry!

Is the program correct?

- Values at the end of iteration t = values at the beginning of iteration t+1.
- Values at end: values at beginning of iteration n+1 if it had been there.
- Values at end: i = n+1, facn = n!

Practice assignment

- Write invariants for the programs in this week's lab.

Character data type

char:

- 1 byte (typically).
- Behaves like int for purposes of arithmetic.
- can move data from int to char etc.
- Behaviour different for << and >>.

Example

```
char c;  int x;
cin >> c;  // say user types letter a
                  // c gets ASCII code.
x = c;
cout << x; // 97 printed.
c++;          // 1 added to c.
cout << c; // letter b printed.
```

Character Literals

- '<single character>' : ASCII value of the character

```
char c = 'a', d = '*' ;  int p = 'b';
```

- Some other literals: '\n' : enter key.

```
cout << p << '\n';  // same as endl
```

- ASCII codes a-z are consecutive. Also A-Z. Also 0-9.

Case conversion program

```
main_program{
  char in_ch;
  cout << “Lower case character: ”;
  cin >> in_ch;
  cout << “Upper case:”
          << in_ch + 'A' – 'a' << '\n';
}
```

Text processing

- Requires storing and operating on many characters. “character strings”
  - will consider later.

Operators on numbers

- % : remainder operator.

int p = 100 % 37; // 26

  - precedence same as *,/

- ++ : increment operator

p++; // same as p = p + 1;

++p; // same as p = p + 1;

  - difference between them? later.

- -- : decrement. subtract 1. p-- or –-p.

Accumulating assignments (compound assignments)

- += : add and assign;

```
p += q;  // same as p = p + q;
```

- *=, -=, /= : similar.
- Useful in accumulation idiom:

```
sum += term;
```

- ++, -- useful in sequence generation idiom.

“Clever” C++

- Assignment expression: var = exp
  itself has value = what was assigned.

```
float p, q;
p = (q = 3.5);  // p becomes 3.5
p = q = 3.5;    // “right associative”
int r;
p = r = 3.5;  // r = ?, p = ?
```

“Clever” C++

- Increment, decrement expressions.

```
p = ++q;                r = s++;
```

- Equivalent to

```
q = q+1;                r = s;
p = q;                  s = s+1;
```

- Similarly --.
- a = b += c--; // Dont even think about it!