PRO1

Mid-course Summary

Glyn Morrill

Definition

PRO1 is an initial course in programming (in C++).

A definition: Programming is the design of virtual machines.

Danger: Machines that don't work properly do harm.

Moral: Program well.

Features of good programming

- Time efficiency (time is money)
- Space efficiency (memory is cheap, but must be handled)
- Transparency (understandability)
- Elegance (beauty is truth)
- Concision/parsimony (good, for brief: twice good)
- Complementarity of code and documentation (comment what the code leaves unsaid)

Program scheme

```
// P12345

#include <iostream>
using namespace std;

// Pre: (Input ... awaits s.t. ...)
// Post: (Output ... printed s.t. ...)

int main() {
    instruction_1
    ⋮              // (at least one comment)
    instruction_n
}
```

Basic data types: constant expressions

`int` integer: `0`, `1`, `-1`, `2`, `-2`, `3`, `...`

`float` real: `0.0`, `0.0000001`, `-1.0`, `3.1415927`, `...`

`double` real: `3.141592653589793`, `...`

`bool` Boolean: `true` ($\{\emptyset\}$), `false` ($\{\}$) symbolic and numeric

`char` character: `'a'`, `'b'`, `...`, `'z'`, `...` symbolic and numeric

(`string` string: `""`, `"a"`, `"aa"`, `"aab"`, `...`)

Basic data types: operators

`int` + (plus), - (minus), * (product), / (quotient), % (remainder)

(x / 0, x % 0 undefined: execution error)

`float`, `double` + (plus), - (minus), * (product), / (division)

(x / 0.0 undefined: execution error)

`bool` `not` ($\dot{\neg}^{\{\emptyset\}}$), `and` ($\cap$), `or` ($\cup$)

`char` +, - (on numeric values), ...

Comparisons (decreasing order of recommendation)

Even handed

- < (monotone simple) <= (monotone dipthong)
- > (antitone simple) >= (antitone dipthong)

Uneven handed

- == (equality)
- != (nonequality)

Expressions (decreasing precedence)

Unary:

- numerical signs: `+`, `-`;
- boolean negation: `not` `!`.

Binary:

- multiplicatives: `*`, `/`, `%`;
- additives: `+`, `-`;
- inequations: `<`, `<=`, `>`, `>=`;
- equations: `==`, `!=`;
- boolean conjunction: `and` `&&`;
- boolean disjunction: `or` `||`.

Left association for binary operators of the same precedence.

Smart binary boolean evaluation: $\{\emptyset\}$ || . . . = $\{\emptyset\}$; {} && . . . = {}

Recommendation: put spaces before and after binary operators

Declare and initialise variables with minimal scope

Compilers might warn ‘variable declared out of scope’.

```
int m;
double sum, min, max;
cin >> m >> sum;
min = max = sum;
for (int i = 1, i < m, ++i) {
    double x;
    cin > x;
    sum += x;
    if (x < min) min = x;
        else if (max < x) max = x;
    }
int avg = sum / m;
```

Statements

- `cin >>`, `cout <<` (note direction of arrows)
- assignment: `Variable = Expression` ("becomes")
- conditional: `if` BoolExp `...` (`else ...`)
- `while (BoolExp) ...`
- `for (int i = IntExp; BoolExp; i = IntExp(i)) {`
  ⋮
  }
- which abbreviates
  `int i = IntExp;`
  `while` BoolExp {
  ⋮
  i = IntExp(i);
  }