How to Write Code

Section 1.3–1.4

Dr. Mayfield and Dr. Lam

Department of Computer Science
James Madison University

Sep 11, 2015

CS 280 weekly rubric

Contest						
Criteria	**Ratings**					**Pts**
Live Contest	Present 2 pts		Late 1 pts		Absent 0 pts	2 pts
UVa Problems	Solved 2+ 2 pts	Solved 1 and Tried 2+ 1.5 pts	Solved 1 or Tried 2+ 1 pts	Tried 1 0.5 pts	No submission 0 pts	2 pts
					Total Points: 4	

Submit programs via Canvas for partial credit

- Trying two problems is a C
- Solving one of them is a B^+
- Solving two of them is an A

Don't forget!

Organize your files

- Create a course directory (e.g., CS280)
- Create subdirectory for each problem
  - acm2013A-TextRoll
  - uva11172-Relational
  - ...

Automate the testing

- sample.in and sample.out files
- run and diff before submitting

Use a simple text editor

Advantages for CS 280

- Faster — no project setup required
  - Your files end up in the right place
- Forces you to learn (no autocomplete)

Tips for `gedit` (Edit > Preferences)

View Highlight current line

View Highlight matching bracket

Editor Tab width (*change to 4*)

Editor Insert spaces instead of tabs

Editor Enable automatic indentation

Development process

1. Write code to read the input
   - Debug by printing the input
2. Write code to print the output
   - Double check the formatting
3. Write code to solve the problem
   - Run interactively as needed
   - Debug with print statements
4. Test code using sample.in/out
   - Need to write additional cases
5. Submit to UVa Online Judge
   - Incorrect = 20 minutes penalty

Let's do one together

ACM 2013 Problem A: Text Roll

UVa 11172: Relational Operators

From Java to C

Java

```java
public class Hello {
    public static void main(String[] args) {
        System.out.println("Hello World!");
    }
}
```

C

```c
#include <stdio.h>

int main()
{
    printf("Hello World!\n");
    return 0;
}
```

First C++ program

C++

```
#include <iostream>
using namespace std;

int main()
{
    cout << "Hello World!" << endl;
}
```

I/O streams

- `cin` is standard input, `cout` is standard output
- `endl` inserts a newline character *and flushes*

C++ operators

- << is called the insertion operator
- Multiple <<'s can be chained together

std namespace

The C++ standard library includes `cin`, `cout`, etc.

```
#include <iostream>

int main()
{
    // notice the scope operator
    std::cout << "Hello World!";
}
```

Everything is defined in the `std` namespace

```
#include <iostream>
using namespace std;

int main()
{
    cout << "Hello World!";
}
```

Comparison: formatting output

Java

```java
// print two decimal places
System.out.printf("%.2f", result);
```

C

```c
#include <stdio.h>

// print two decimal places
printf("%.2f", result);
```

C++

```cpp
#include <cstdio>

// print two decimal places
printf("%.2f", result);
```

The best features of C are also in C++

Comparison: parsing input

Java

```
import java.util.Scanner;

Scanner in = new Scanner(System.in);
int cases = in.nextInt();
while (in.hasNext()) {
    String word = in.next();
    ...
```

Less code!

C

```
int cases;
char word[128]; /* explicit size */

scanf("%d", &cases);
while (scanf("%s", word) != EOF) {
    ...
```

C++

```
int cases;
string word;

cin >> cases;
while (cin >> word) {
    ...
```

Reading lines in C/C++

C

```c
int main()
{
    int cases;
    char line[1024];

    scanf("%d", &cases);
    fgets(line, 1024, stdin);
}
```

C++

```cpp
int main()
{
    int cases;
    string line;

    cin >> cases;
    getline(cin, line);
}
```

The `getline` function reads to the end of the current line

- Note the extraction operator (>>) does not!
- But it will skip whitespace to find the next item
- `fgets` retains the newline at the end of each line