CSE / ENGR 142 Programming I

Style

Aspects of Quality Software

- Getting the syntax right
  - This may seem hard at first, but turns out to be the easiest part of all
- Getting the logic right
  - Sometimes difficult, but absolutely essential
- Today's focus: Programming with good style
  - What does this mean, and why does it matter?

Programming Style

- **A program is a document:**
  - **Some of it is read by a computer.**
  - **ALL of it is read by people.**
  - **Donald Knuth: "literate programming"**
- **"Style" is a catch-all term for people-oriented programming.**
  - **comments, spacing, indentation, names**
  - **clear, straightforward, well-organized code**
  - **code quality**

Style in CSE 142

- It is common for employers to have style requirements that all programmers must follow.
- Along the way, we will suggest and sometimes require particular points of style in programs that are turned in for CSE 142.
  - "Along the way" starts today!

/* Comments */

Comment block at front of program

Comment block per major section

small ones throughout

```
/************************************************************
* Program:   Mi_To_Km
* Purpose:   Miles to kilometers conversion
* Author:    A. Hacker, 1/18/00  Section AF (Turing)
************************************************************/

/* Calculate volume of cylinder and ...
 * Inputs:       radius, height, ...
 * Output:       volume, ...
 * Assumes:      radius, height nonnegative */
 .
 .
 .
/* Tell user it's negative. */
```


Comments

- **Say *why*, not *what*:**
  - **NO: /* subtract one from sheep */**
    sheep = sheep - 1;
  - **YES: /* account for the sheep that the big bad wolf just ate.*/**
    sheep = sheep - 1;

Spaces

- Use blank lines to separate major sections.
- Vertically align like things:

```
x         = 5 ;
y_prime = 7 ;
z_axis    = 4.3;
```

- Leave space around operators:

No: `y=slope*x+intercept;`

Yes: `y = slope * x + intercept ;`

Use parentheses for emphasis, too

Yes: `y = (slope * x) + intercept ;`

Indentation
Like an outline, indent subordinate parts.

Identifiers (Review)

- Identifiers name variables and other things
  - Letters, digits, and underscores (_)
  - Can't begin with a digit
  - Not a reserved word like *double*, *return*
- "Case-sensitive"
  - *VAR*, *Var*, *var*, *vAr* are all different
- Using all CAPITAL letters is legal...
  - but usually reserved for *#define* constants (soon to be explained)

What's in a Name?

- Extremely valuable documentation.
- Microsoft Excel has over 65,000 variables.
- How long is just right?
  - *m*
  - *mph*
  - *miles_per_hour*
  - *average_miles_per_hour_that_the_car_went_before_noon*
- Avoid similar names: *mph* vs. *Mph* vs. *mqh*

More Examples

OK	Illegal	Legal, but what about the style?
rectangleWidth	10TimesLength	a1
rectangle_Width	My Variable	l
rectangle_width	int	O
length_10_Rectangle		rectangleWidth *and* rectanglewidth *or* rectangle_Width

Clarity

Do "obvious" things the obvious way

No: `x = (y = x) + 1 ;`

Yes:
```
y = x ;
x = x + 1;
```

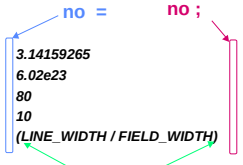
Don't be tricky, cute, or clever without GOOD reason.

If so, comment it!

#define

Named constants:

```
#define PI              3.14159265
#define AVOGADRO        6.02e23
#define LINE_WIDTH      80
#define FIELD_WIDTH     10
#define FIELDS_PER_LINE (LINE_WIDTH / FIELD_WIDTH)
...
area = PI * radius * radius ;
lines = fields / FIELDS_PER_LINE ;
```



Notes:

yes UPPER CASE

yes ( )

Why #define?

- Centralize changes
- No "magic numbers" (unexplained constants)
  - use good names instead
- Avoid typing errors
- Avoid accidental assignments to constants

```
double pi ;
pi = 3.14 ;
...
pi = 17.2 ;
```

vs.

```
#define PI 3.14
...
PI = 17.2 ;
```

← syntax error

Putting It All Together

```
/* Convert miles per hour to feet per second
 * Author: ...
 * Date:   ...
 */

#include <stdio.h>

/* conversion constants. */
#define FEET_PER_MILE          5280.0
#define SECONDS_PER_HOUR       (60.0 * 60.0)

int main(void)
{
  double miles_per_hour;       /* input mph */
  double feet_per_second;      /* corresponding feet/sec */
  double feet_per_hour;        /* corresponding feet/hr  */

  /* prompt user for input */
  printf("Enter a number of miles per hour: ");
  scanf("%lf", &miles_per_hour);

  /* convert from miles per hour to feet per second */
  feet_per_hour = miles_per_hour * FEET_PER_MILE;

  feet_per_second = feet_per_hour / SECONDS_PER_HOUR;

  /* format and print results */
  printf("%f miles per hour is equal to %f feet per "
         "second.\n", miles_per_hour, feet_per_second);

  return(0);
}
```


Many small points; Big cumulative effect...

```
#include<stdio.h>
int main(void){double v1,v2,v3,v4,v5;pr\
intf("Enter a number of miles per hour:\
");scanf("%lf",&v1);v5=v1*14.6666667;pr\
intf("%f miles per hour is equal to %f \
feet per second.\n",v1,v5);return(0);}
```


Style Summary: Clarity is Job #1

- DO
  - Use plenty of comments
  - Use white space
  - Use indentation
  - Choose descriptive names
  - Use named constants
- DON'T
  - be terse, tricky
  - place speed above correctness, simplicity
  - use "magic numbers"