CS70

Fall 2000

Some Good and Bad Style Examples

1 Some Really Awful Code

You're not expected to understand all of the following; it's C code that uses some horrible and unreliable tricks.

```
/* Kolar's Hanoi Tower algorithm no. 3 */

#include <stdio.h>
#include <stdlib.h>

#define PR      (void)printf(
#define PE      (void)fprintf(stderr,

#define ALLO(x) { if((x = (int *)malloc((n+3) * sizeof(int))) == NULL) {\
                  PE #x " allocation failed!\n"); exit(1); }}

main(int argc, char *argv[])
/*========================*/
{
 int i, *a, *b, *c, *p, *o1, *o2, *e, n, n1;

 n = atoi(argv[1]);
 n1 = n+1;
 ALLO(a)
 ALLO(b)
 ALLO(c)

 a[0] = 1; b[0] = c[0] = n1;
 a[n1] = n1; b[n1] = n+2; c[n1] = n+3;
 for(i=1; i<n1; i++) {
   a[i] = i; b[i] = c[i] = 0;
 }
```

```
 o1 = a;
 if(n&1) { o2 = b; e = c; }
 else    { o2 = c; e = b; }

 e[--(*e)] = 1; (*o1)++;
 PR"\n|"); for(i=n;i>=a[0];i--) PR" %d",a[i]);
 PR"\n|"); for(i=n;i>=b[0];i--) PR" %d",b[i]);
 PR"\n|"); for(i=n;i>=c[0];i--) PR" %d",c[i]);
 PR"\n");
 p = e; e = o1; o1 = o2; o2 = p;

 while(*c>1) {
  if(o1[*o1] > e[*e]) o1[--(*o1)] = e[(*e)++];
  else                e[--(*e)] = o1[(*o1)++];
  PR"\n|"); for(i=n;i>=a[0];i--) PR" %d",a[i]);
  PR"\n|"); for(i=n;i>=b[0];i--) PR" %d",b[i]);
  PR"\n|"); for(i=n;i>=c[0];i--) PR" %d",c[i]);
  PR"\n");
  p = e; e = o1; o1 = o2; o2 = p;
  e[--(*e)] = 1; (*o1)++;
  PR"\n|"); for(i=n;i>=a[0];i--) PR" %d",a[i]);
  PR"\n|"); for(i=n;i>=b[0];i--) PR" %d",b[i]);
  PR"\n|"); for(i=n;i>=c[0];i--) PR" %d",c[i]);
  PR"\n");
  p = e; e = o1; o1 = o2; o2 = p;
 }
}
```

Here are some questions:

1. What is the purpose of the variable `p`?
2. Where are all the places `o1` is used?
3. If the first assignment statement sets `n` to 4, what will be placed in `b[0]` through `b[4]` by the first `for` statement? What about `a[0]` through `a[4]`?
4. Is the variable `n1` really necessary?

2 Making Code Easy to Modify

2.1 Poor

```
int a, b = 0, c, *d;
```

Note: `d` is a pointer to an integer, something we'll cover later.

2.2 Better

```
int a;
int b = 0;
int c;
int *d;
```

2.3 Even Better

```
int          temp;                  // Used for swapping two students
int          alreadyDidIt = 0;      // NZ if students already swapped
int          totalStudents;         // No. of students enrolled
int         *studentNumber;         // Ptr to ID no. of current student
```

3 Helping the Reader Find Things

3.1 Version 1: Bad

```
int function1(...);
char* function2(...);
const class longnamethatshardtoparse & function3(...);
```

3.2 Version 2: Better

```
int                     function1(...);
char*                   function2(...);
const class LongNameThatsEasierToParse
                        function3(...);
```

3.3 Version 3: Still Better

```
int             gcd(...);   // Return greatest common divisor
char*           nextWhite(...);
                            // Return pointer to next whitespace
                            // ..or NULL if none found
const class StudentEnrollmentInformation
                findStudentGivenName(...);
                            // Locate student info based on full
                            // ..name.  If no such student, create
                            // ..one and return that.
```

4 An Example from K&P

4.1 Version 1

```
if (LC == 0  &&  RC == 0)
    child = 0;
else if (LC == 0)
    child = RC;
else
    child = LC;
```

4.2 Version 2

```
if (LC == 0)
    child = RC;
else
    child = LC;
```

Which is better? Why? Can it be improved further?

5 Some Sample Comments

5.1 Silliness

```
a = b; // assign b to a
return a; // return a
```

5.2 Giving Useful Information

(Note: these statements don't make sense in sequence; each should be considered as if it stood alone.)

```
a = b;      // Assume result will equal b

a = b;      // Initialize a to most likely answer

a = b;      // In this case, answer was already computed

return a;

return a;   // Success!

return a;   // In this case, result equals original value
```

P.S. It's not accidental that the above comments are vertically aligned.

5.3 Comments as a Design Tool

```
/*
 * Now that we have inserted the new record, the heap
 * condition may be violated.  To restore the condition,
 * we must "percolate" the inserted item upwards through
 * the heap.  The method for doing so is as follows:...
 */
```

5.4 Does It Need Comments?

```
for (int i = 0;  i < 100;  i++)
    a[i] = 1;
```

5.5 The Hollywood Question: "What's My Motivation?"

```
/*
 * a is a 100-element array of flags, representing
 * whether the jersey number equal to the index is
 * still available.  We begin by assuming that all
 * values are available.  When we issue a particular
 * number, we set the flag to zero.  We can limit the
 * array to 100 elements because the size of a
 * football squad is limited by the rules of the game,
 * and football jerseys are traditionally limited to
 * two digits.
 */
```

6 Magic Numbers

```
const int           MAX_JERSEY_NUMBER = 99;

/*
 * The following is a nasty C++ trick you'll want to know:
 */
enum {
                    MAX_STUDENTS_IN_CLASS = 150
};
```