Code Style

Why do we write code for humans?

• Easier to read
• Easier to understand
• Less mistakes
• Faster overall development time

Good Coding Practices

What is good style?

• Indentation and Bracketing
• Names of variables and functions
• Repetition (or not) of code
• Clear comments

Consistency

• The easier it is to read and understand, the less mistakes we'll make

Poor Code Style

Can we work with code that's hard to read?

- This is functionally our Dice Checking program:

```c
int main(void) {
    int x; int a; int b; int y;
    printf("Please enter how many sides are on your dice: "); scanf("%d", &x);
    printf("Please enter the value of the first die: "); scanf("%d", &a);
    if (a<1) { printf("Die roll value: %d is outside of the range 1 - %d.\n", a,
        // this bit does the dice thing
        a = a % x; if (a == 0) a = x;}
    if (a > x) {
        printf("Die roll value: %d is outside of the range 1 - %d.\n", a, x);
        a = a %x; if (a== 0) a = x;}
    printf("Your roll is: %d\n", a);
    printf("Please enter the value of the second die: "); scanf("%d", &b);
    if (a < 1 || a > x) {
        printf("Die roll value: %d is outside of the range 1 - %d.\n", b, x);
b = b % x;
        if (b ==0) b = x;
    }
}
```

What went wrong?

We want more than: “Oh wow, that’s a mess”

What are the specific improvements that can make this better?

In the face of disaster, keep a clear head and focus on what can be fixed
Specific Issues

Header comment doesn’t show the program’s intentions
No blank lines separating different components
Multiple expressions on the same line
Inconsistent indenting
Inconsistent spacing
Variable names don’t make any sense
Comments don’t mean anything
Inconsistent bracketing of if statements
Bracketing is not indented
Inconsistent structure of identical code blocks
The easter egg - there’s actually incorrect code also!

Keeping your house (code) clean

Regular care is always less work than a big cleanout
Write comments before code
Name your variables carefully before you use them
{ everything inside gets indented 4 spaces
} line up your closing brackets vertically with the line that opened them
One expression per line
Maintain consistency in spacing

Comments before code

Comments before code. It’s like planning ahead
Making plans with comments
You can fill them out with correct code later
Most of these comments can stay even after you’ve written the code

// Checking against the target value
if () {
   // success
} else if () {
   // tie
} else {
   // failure (all other possibilities)
}

Naming Variables

Variable names are for humans
Can you describe what a variable is in a word or two?
If another student was to read this name, would it make sense?
Does it distinguish it well against the other variables?
**Indentation**

A common convention is to use 4 spaces for indentation

```c
int main(void) {
    // everything in main is indented 4 spaces
    int total = 5;
    if (total > 10) {
        // everything in this if is indented 4 more
        total = 10;
    }
    // this closing curly bracket lines up
    // vertically with the if statement
    // that opened it
}
// this curly bracket lines up vertically
// with the main function that opened it
```

**Spacing**

Operators need space to be easily read

```c
// NOT LIKE THIS!
int a=20;
int b=22;
int total;
if (a<b&&b>=15) {
    total=a+b;
}
// Like this :)
int a = 20;
int b = 22;
int total;
if (a < b && b >= 15) {
    total = a + b;
}
```

**One Expression Per Line**

Any single expression that runs should have its own line

```c
// NOT LIKE THIS!
int num_1; int num_2;
num_1 = 25; num_2 = num_1 + 10;
if (num_1 < num_2) { num_1 = num_2; }
```

```c
// Like this :)
int num_1;
int num_2;
num_1 = 25;
num_2 = num_1 + 10;
if (num_1 < num_2) {
    num_1 = num_2;
}
```

**More Information about Coding Style**

Code Style isn’t just to make it look nice

- Reduces errors later in development
- Makes it easier to test and modify
- Overall, speeds up development
- Makes your co-workers hate you less
- Your assignments have coding style marks (more on this when they release)
### Code Style Summary

The course webpage has a Style Guide

- Wherever you end up coding, there will be different styles
- Our style is only one of them, but a good place to start!

### Code Review

#### What is a code review?

- Having other coders look over your code
- Having an active discussion about the code
- Automated testing can test functionality, but not necessarily usability
- Humans can help you improve as a human!
- Similar to proof-reading a document
- Super valuable to discuss different approaches to the same problem

#### Why do we review code?

<table>
<thead>
<tr>
<th>As the code writer</th>
<th>As the code reviewer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Get feedback on how easy it is to understand our code</td>
<td>Get to see how someone else writes code</td>
</tr>
<tr>
<td>Hear about other people's ideas on solving the same problem</td>
<td>Learn more about different ways to solve problems</td>
</tr>
</tbody>
</table>

#### Different ways to review code

<table>
<thead>
<tr>
<th>Pair Programming</th>
<th>More formal review</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lab partners actively discussing solutions</td>
<td>Finish a section of code, then ask people to review it</td>
</tr>
<tr>
<td>Live reviewing and discussion while in development</td>
<td>Sometimes in person, sometimes using software tools</td>
</tr>
</tbody>
</table>
How to do Pair Programming well

- Also, learning how to work with other programmers!
- One person on the keyboard (sharing screen in a breakout room)
  - Thinking about how to structure the C and syntax
- One person over the shoulder (watching the shared screen)
  - Thinking about how to solve the problem
- Active discussion between the two of you as you go (mics on)
- This means the code is constantly under review
- Programming with others is one of the best ways to learn!

Conducting a Formal Code Review

Reviewing a finished piece of code

Reviewers will read the code and help with it

Remember, we’re judging the code, not the coder!

We’re all learning . . . this is not about picking at mistakes

Points to Discuss

Where is it easy or hard to understand the code?

What are the different possible ways the code can solve the problem?

Any little issues we can help solve?

What not to do in a Code Review

These things will not help us learn better code:

“You did this wrong”

“Your code is bad”

“Here are all the mistakes in this code”

We’re doing this to help ourselves and others learn more!

No judgement, only help!

How does one help someone else learn?

Understand that it’s very hard to put your work up for review

We’re not here to judge the code’s standard

We’re here to help everyone learn more

There is no single right way to solve a problem

If your way and someone else’s way are different, you can both be right

Next week’s Tutorial will have a demo Code Review

Your tutor will do the first review so you can see what it’s like

After this, every code review will be lead by students

You can also get together with other students to review your Lab work