

## CS137 Lecture 1 Review

```
    ↗ standard input-output header file
    #include <stdio.h>

    ↗ empty parameter list
return type ← int main(void) {
    printf("Hello World. \n");
    ↗ the 'f' in printf stands for "formatted"
    ↗ printf stands for "formatted"
    return 0;
    ↗ return status to caller
}
```

## How to RUN CODE

- ① `nano prog1.c` → create prog1.c  
\* default name given to the built executable  
↑
- ② `gcc prog1.c` → compile into executable (a.out)
- ③ `./a.out` → `./` refers to the current directory
- ④ `gcc -o prog1 prog1.c` → sets output file to the first parameter
- ⑤ `./prog1`

Kilroy



### Other terminal commands

- **pwd** → view which directory is current
- **cd** → move one level up in the directory tree
- **cd DIR-NAME** → move down one level into the directory named "DIR-NAME"
- **ls**  
↳ lists all of the files in the current directory



## CS137 - Making Decisions (Sep 15, 2022)

```
#include <stdio.h>
int main(void) {
    int i=3;
    if (i%2 == 0) → missing curly brackets (syntax)
    if (i==0) printf("zero\n");
    else printf("how odd\n");
    return 0;
}
```

This "else" statement belongs to this "if" statement

↳ the code prints nothing

indentation doesn't matter in C

↳ "Dangling else" issue

\* each "else" statement belongs to the nearest "if" statement...

- Cascaded if statements → "if" statement followed by "else if" and "else" statement(s)
- Nested if statements → "if" statements nested within other "if" statement(s)

## TERNARY CONDITIONAL OPERATOR

expr1 ? expr2 : expr3

\* If expr1 is true, this conditional statement returns the value of expr2. Otherwise, it returns expr3

(continued...)



E.g.  $i, j \in \mathbb{Z}$

```
printf ("10/d\n", i > j ? i = j);
```

↳ returns maximum of  $i$  and  $j$

## SWITCH STATEMENTS

Syntax:

lets C know that  
this case is finished

C  
checks  
through  
the  
cases

```
switch (expr) {  
  ↓ { case const-expr : statements break;  
      ...  
      case const-expr : statements break;  
      default : statements  
}
```

case const-expr → similar to if ( $expr = \text{const-expr}$ )

\* If no "break;" appears in a case, the flow of control will fall through to the following cases until a "break;" is reached

default → if other statements are not met  
(similar to "else" statement)

\* If no "break;" it will continue checking the following cases



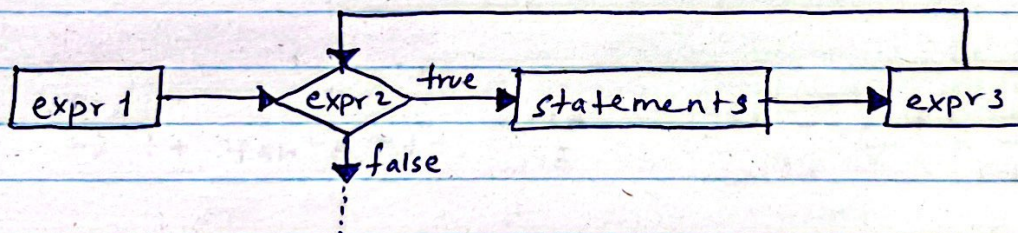
## CS137 - LOOPS: FOR LOOP

Syntax:

```
for (expr1; expr2; expr3) { statements }
```

- $\text{expr1} \Rightarrow$  initializer before loop is executed
- $\text{expr2} \Rightarrow$  usually a condition
- $\text{expr3} \Rightarrow$  usually an incrementer/decrementer

\* In C99 etc.,  $\text{expr1}$  can be an initialization of a new variable whose scope is only for the loop



Hilroy