Programming in C

Lecture #2
Kenneth W. Flynn
RPI CS

Outline
- Application: Floating Point
- Exercise: Activity 2.1
- Theory: Algorithms
- Applications:
  - If Statements
  - Relational Operators
- Exercise: Activity 2.2
- Application: Switch Statements
- Exercise: Activity 2.3

Floating Point
- C has many different kinds of variables
  - Last week, we say variables of type int
  - Another kind of variable is a floating point number, float
- `float x, y;
  x = 3.3;
y = -2.9e8;`

Floating Point Aritmetic
- +, -, *, and / all work as expected
- Can we combine ints and floats?
  - Yes
  - `float x = 3.3;
    int y = 3;
    float z = x / y;
    /* Z is now 1.1 */`

Types And Division
- Consider the statements:
  - `int x = 5, y = 2;
    float z;
    z = x / y;`
  - This does not make z = 2.5!
  - Since x and y are both integers, integer division is applied between them. The result is 2. z is now 2.
  - So how do we do this? Casting
Casting

- **Casting**: Changing the type of a variable
- In C, you can change any variable into any other type, for better or for worse.
- Casting is done as follows:

```c
int x = 5, y = 2;
float z;

z = (float) x / y;
/* OR */
z = x / (float) y;
```

---

I've Been Cast Into Such a Great Role!

- \( z = (\text{float}) \frac{x}{y}; \)
- \( z = (\text{float}) \frac{5}{2}; \)
- \( z = \frac{5.0}{2} \)
- \( z = 2.5 \)
- Since the numerator is now a floating point variable, floating point division takes place
- Note that this doesn't change \( x \)'s value

---

Break A Leg, And You Might Need a Cast

- Subtler casting occurs as follows:
  ```c
  int x = 3;
  float y;

  y = x; /* x is cast to a float implicitly */
  ```

---

Input and Floating Points

- Last week, we saw `scanf`
- We can use `scanf` and `printf` with floating point numbers as well.
- New format specifier: `%f`

---

Floating Point Example - .

- /* Floating Points */

```c
#include <stdio.h>

int main (void)
{
  float x;
  printf("Please enter a float:");
  scanf("%f", &x);
  printf("You entered: %f.\n", x);
  return 0;
}
```

---

Example Output

- `~>a.out`  
  Please enter a float:7.3  
  You entered: 7.300000.
  `~>`

Kenneth W. Flynn (7-12)  
10/25/98
Activity 2.1

Fractional Mode

- Write a program to read fractions from the user, and then display them back:
- Sample Execution:

  Numerator: 5
  Denominator: 2

  You entered 5/2 (2.500000)
  Squared is 25/4 (6.250000)
  Thank you.

Theory

Algorithms

- Algorithms
  - Sequence of steps for solving a problem in a finite amount of time
  - For each program you write, you should develop an algorithm.
  - The most common (and best) technique for doing so is...
  - *Top Down Design*: Construct a "high-level" series of steps for a problem. Refine each step into sub-steps

Top Down Design Example

- Our Fractional Mode Program
- Step 1: Get the num & den from the user
- Step 2: Compute the fraction
- Step 3: Compute the squares
- Step 4: Display output

Refining...

- Step 1: Get the num & den from the user
  - 1a: Get numerator
  - 1b: Get denominator
- Step 2: Compute the fraction
- Step 3: Compute the squares
  - 3a: Compute square of num
  - 3b: Compute square of den
  - 3c: Compute float square
- Step 4: Display output
  - 4a: Display fraction
  - 4b: Display squares

Kenneth W. Flynn  (13-18)
10/25/98
Refining More...

- Step 1: Get the num & den from the user
  - 1a: Get numerator
    - Display prompt
    - Read input
  - 1b: Get denominator
    - Display prompt
    - Read input

The If Statement

- C provides many decision making constructs which allow you to direct the flow of control in your program
- The simplest is the if statement
- "If a condition is true, then do this, else do this"
- We have 3 tasks:
  - Figure out the condition
  - Figure out the "Then" part
  - Figure out the "Else" part

The Divider

- Our fraction program had a problem: what happens if we enter 0 for the denominator?
- Our program crashes!
- Programs should NEVER crash. Errors such as this should be handled in a clean, consistent way.
- Let's modify our algorithm

Improving Fractional Mode

- Step 1: Get the num & den from the user
  - 1a: Get numerator
  - 1b: Get denominator
- Step 2: If den is 0, print error, otherwise...
- Step 3: Compute the fraction

The C If Statement

- if (condition)
  /* Then Part */
  else
  /* Else Part */

- Note that there is no keyword "then," only "else"
- One of the following:
  - Single Statement
  - Block, surrounded by braces
If Example

```c
/* Compute fraction */

if (den == 0)
    printf("ERROR:Den is 0\n");
else
{
    fract = (float) num / den;
    /* ... */
}
```

Relational Operators

- `x == y`
- Is `x` and `y` equal?
- Note that this is not `=`
- `!=`
- Not equal
- `x > y`
- Is `x` greater than `y`?
- `<`, `>=`, `<=`
- Less than, Greater than or equal to

Increment and Decrement

- `x++`, `++x`
  - Adds one to `x`
- `x--`, `--x`
  - Subtract one from `x`
- Postfix (afterwards) does the action after the current expression
- Prefix (before) does the action before the current expression

Increment and Decrement Examples:

```c
int x = -1;
x++;
printf("X is %d\n", x++);
printf("X is %d\n", x);
printf("X is %d\n", --x);
```

Which Is Bigger?

- Write an algorithm in English for the following program before coding it
- Write a program that gets two integers from the user, determines which is bigger, and displays one of the following messages:
  - The FIRST is bigger
  - The SECOND is bigger
  - They are EQUAL
Switch

Switch statements are sort of a special if-then-else-if-then-else type construct. An integer value is determined, depending on this value, one of many cases is evaluated.

Switch Syntax

```
switch (value)
{
    case val1: /* Statements */
        break;
    case val2: /* Statements */
        break;
    /* More cases */
    default: /* Statements */
        break;
}
```

Switch Notes

- Statements in a case are executed until a break is reached, even into other cases.
- Default case is evaluated if no other case matches. Include a default case in all switch statements.
- Use switch statements as an alternative to massively nested if-then-else structures.

Switch Example

```
int x, y, z;
int err = 0; /* Init to false */

printf("Num Den: ");
scanf("%d", &x, &y);
switch (y) /* Better as an if... */
{
    case 0: err = 1; /* Set error */
        break;
    default: z = x / y;
        break;
}
if (!err)
    printf("Z is %d\n", z);
else
    printf("ERROR!\n");
```
The Chooser

Write a program that gets one number from the user. If the number is 0, print out "Zero", if it is 1, print out "One", etc. Handle the numbers between 0 and 5. All other numbers should produce an error message. Use a switch statement.