ECE/CS 151
Intro to Programming I

The Details So Far...
Chap. 2 – 3.4
Reading Characters w/ `scanf()`

```c
#include <stdio.h>

int main () {
    int x;
    char c;

    printf("Enter a number: ");
    scanf("%d", &x);
    printf("Enter a character: ");
    scanf("%c", &c);

    return 0;
}
```

"scanf_ex.c" 13L, 180Cc
12,1 All
Uh-oh, it didn’t ask me for a char
One solution...

```c
#include <stdio.h>

int main () {
    int x;
    char c;

    printf("Enter a number: ");
    scanf("%d", &x);
    getchar(); /*Read the newline character*/
    printf("Enter a character: ");
    scanf("%c", &c);

    printf("%d %c\n", x, c);

    return 0;
}
```

"scanf_ex.c" 16L, 263Cc

6, 3

All
Another Solution

```c
#include <stdio.h>

int main () {
    int x;
    char c;

    printf("Enter a number: ");
    scanf("%d", &x);
    getchar(); /*Read the newline character*/
    printf("Enter a character: ");
    scanf("%c%c", &c);

    printf("%d %c\n", x, c);

    return 0;
}
```

Case w/o breaks & if statements

```c
switch( x ) {
    case 0:
        printf("X is zero\n");
    case 1:
        printf("X is one\n");
    case 2:
        printf("X is two\n");
    default:
        printf("Invalid number!!!\n");
}

if( x == 0) {
    printf("X is zero\n");
}
if( x == 1) {
    printf("X is one\n");
}
if(x == 2) {
    printf("X is two\n");
}
printf("Invalid number!!!\n");
```
What does this do?

<expression1> ? <expression2> : <expression3>

• Evaluate expression1
  – If true, return expression2
  – If false, return expression3

• Example:
  
x = (10 > 4) ? 10 : 4;
  return (10 > 4) ? 10 : 4;
  printf("%d\n", (10 > 4) ? 10 : 4);
Why don’t we want to do this?

```c
if ( x > y) {
    printf("X is greater than Y\n");
}
if ( x < y) {
    printf("X is less than Y\n");
}
if(x == y) {
    printf("X is equal to Y\n");
}
```
Why don’t we want to do this?

if( x > y) {
    printf(“X is greater than Y
”);
}
else if( x < y) {
    printf(“X is less than Y
”);
}
else if(x == y) {
    printf(“X is equal to Y
”);
}
Why don’t we want to do this?

if( x > y) {
    printf(“X is greater than Y\n”);
}
else {
    if( x < y) {
        printf(“X is less than Y\n”);
    }
    else {
        printf(“X is equal to Y\n”);
    }
}
Common Error w/ If/Else...

```c
if( grade >= 90 ) {
    printf(“You get an A”);
}
if( grade >= 80 ) {
    printf(“You get an B”);
}
if( grade >= 70 ) {
    printf(“You get an C”);
}
if( grade >= 60 ) {
    printf(“You get an D”);
}
if( grade < 60 ) {
    printf(“You get a F”);
}
```
Multiple Relational Ops

• What if we want to check if a grade is between two numbers, 93–100 for an A, 90–92 for an A-?

```java
if(grade <= 100 && grade >= 93) {
    printf(“You get an A\n”);
}
else if(grade < 93 && grade >= 90) {
    printf(“You get an A-\n”);
}
else if(grade < 90 && grade >= 87) {
    printf(“You get an B+\n”);
}
...
```
Multiple Relational Ops

• What if we want to check for an invalid grade, i.e. greater than 100 OR less than 0?

```c
if(grade > 100 || grade < 0) {
    printf(“Not a valid grade!!!\n”);
}
```
What does this print?

```c
int x = 5;
if( x < 10 ) {
    x = x + 4;
    printf(“I added 4 to x and it is now %d\n”, x);
} else {
    x = x - 4;
    printf(“I subtracted 4 from x and it is now %d\n”, x);
}
printf(“The else clause is optional (see next if).\n”);
if( x < 7 ) {
    printf(“x is less than 7 or this wouldn’t print. x=%d\n”, x);
}
• What if x =10?
```
Revisit Increment/Decrement Op

- **Prefix**
  
  ```
  ++x;
  ```

- **Postfix**
  
  ```
  x++;
  ```

- **What is the difference?**
  
  ```
  x = n++; VS. x = ++n;
  ```

  ```
  printf("%d\n", x++); VS. printf("%d\n", ++x);
  ```
Revisit #define

#define DOZEN 12
   Will replace:
   int donuts = numBoxes * DOZEN;
   with
   int donuts = numBoxes * 12;

#include <stdio.h>
#define MYSTERY_OP %
int main() {
   int x = 5 MYSTERY_OP 9;
   printf("x = %d\n", x);
}

• What part of the compilation handles the #define statements?
• How do we see this output from this process?
Programming Errors

• Syntax errors
  – Misuse of C language
  – How are they caught?

• Logic errors
  – Doesn’t perform task correctly (aka. bugs)
  – How are they caught?

• Runtime errors
  – Stops your program from running
  – How are they caught?
Syntax Error Examples

• Missing main function
• Use of identifier not declared
• Misspelled Words
• Forget a Semicolon
• Forget Required Keyword
• Missing quote, curly brace, and parenthesis
• Use of single quotes instead of double
Logic Error Examples

• Poorly written programs
  – Add instead of subtract (incorrect operation)
  – Using last two digits for date
  – Same error message for different errors
  – Program that never ends
  – Add one to the largest integer (could be syntax)
  – Divide by zero (could be syntax)
Runtime Error Examples

• Open a file that doesn’t exist
• Segmentation fault
  – Infinite loop that eats memory
  – Forget & in scanf
Debugging Errors

• Syntax:
  – READ compiler errors (pay attention to line #)
  – Use google to search for error

• Logic/Runtime
  – Use printf to find where the code is breaking
    • Print variable values
    • Print indicator messages
Handling Errors – Quiz #3

• What can we do to prevent these errors?
  – Overflow
  – Divide by zero

• Test it before you use it!!!