CSC 101 Lecture Notes Week 3

Summary of C Program Structure

The `char` Data Type

More on Conditionals and Functions

I. The overall structure of a simple C program.

I. The overall structure of a simple C program.

```
/* #includes for stdio.h, etc */
```

I. The overall structure of a simple C program.

```
/* #includes for stdio.h, etc */

/* Function prototypes. */
```

I. The overall structure of a simple C program.

```
/* #includes for stdio.h, etc */

/* Function prototypes. */

/* The main function, which
        calls other functions */
```

I. The overall structure of a simple C program.

```
/* #includes for stdio.h, etc */

/* Function prototypes. */

/* The main function, which
        calls other functions */

/* Function definitions */
```

II. The `char` Data Type

A. So far we've seen `int` and `double`.

II. The `char` Data Type

A. So far we've seen `int` and `double`.

B. Another basic type is `char`.

II. The `char` Data Type

A. So far we've seen `int` and `double`.

B. Another basic type is `char`.

C. Introduced in Chapter 2, page 56.

II. The `char` Data Type

A. So far we've seen `int` and `double`.

B. Another basic type is `char`.

C. Introduced in Chapter 2, page 56.

D. You'll use in Program 2 and Lab 4.

III. Another update stats program.

III. **Another update stats program.**

A. Treat any negative input as an error.

III. **Another update stats program.**

A. Treat any negative input as an error.

B. Do not compute stats if any negative inputs.

III. **Another update stats program.**

A. Treat any negative input as an error.

B. Do not compute stats if any negative inputs.

C. Updated program looks like this:

Up to here, same as before ...

```
/*
 * Input the numbers.
 */
scanf("%lf%lf%lf", &x1, &x2, &x3);
```

Up to here, same as before ...

```
/*
 * Input the numbers.
 */
scanf("%lf%lf%lf", &x1, &x2, &x3);

/*
 * Consider any negative input to be
 * an error.  Do not compute stats.
 */
```

Up to here, same as before ...

```
/*
 * Input the numbers.
 */
scanf("%lf%lf%lf", &x1, &x2, &x3);

/*
 * Consider any negative input to be
 * an error.  Do not compute stats.
 */
if ((x1 < 0) || (x2 < 0) || (x3 < 0)) {
```

Up to here, same as before ...

```
/*
 * Input the numbers.
 */
scanf("%lf%lf%lf", &x1, &x2, &x3);

/*
 * Consider any negative input to be
 * an error.  Do not computer stats.
 */
if ((x1 < 0) || (x2 < 0) || (x3 < 0)) {
    printf("All inputs must be non-negative\n");
}
```

Up to here, same as before ...

```
/*
 * Input the numbers.
 */
scanf("%lf%lf%lf", &x1, &x2, &x3);

/*
 * Consider any negative input to be
 * an error.  Do not compute stats.
 */
if ((x1 < 0) || (x2 < 0) || (x3 < 0)) {
    printf("All inputs must be non-negative\n");
}
else {
```

Up to here, same as before ...

```
/*
 * Input the numbers.
 */
scanf("%lf%lf%lf", &x1, &x2, &x3);

/*
 * Consider any negative input to be
 * an error.  Do not compute stats.
 */
if ((x1 < 0) || (x2 < 0) || (x3 < 0)) {
    printf("All inputs must be non-negative\n");
}
else {
    /* Compute and output the results. */
    . . .
}
```

IV. Highlights of Chapter 4 in the Book

A. *Section 4.1: Control Structures*

1. Explains that a program is a sequence of statements, separated by semi-colons.

2. We've seen this all over the place.

B. *Textbook Section 4.2: Conditions*

1. These are relational and logical expressions.
2. Used in `if` statements.
3. Look closely at Tables 4.1 and 4.5.

Section 4.2, cont'd

4. *Operator precedence* shown in Table 4.6.

   a. Explains difference between

```
a + b * c
```

   versus

```
(a + b) * c
```

Section 4.2, cont'd

b. In the first expression, multiplication is done before addition.

Section 4.2, cont'd

b. In the first expression, multiplication is done before addition.

c. In the second expression, parentheses force addition before multiplication.

Section 4.2, cont'd

5. Another super important topic is the difference between '=' and '==' in C.

Section 4.2, cont'd

5. Another super important topic is the difference between '=' and '==' in C.

   a. The single equals sign means

      "assign a value to a variable".

Section 4.2, cont'd

5. Another super important topic is the difference between '=' and '==' in C.

   a. The single equals sign means

      "assign a value to a variable".

   b. The double equals sign means

      "compare two values".

Section 4.2, cont'd

c. What this means for now:

always use == in conditional expressions

Section 4.2, cont'd

c. What this means for now:

always use == in conditional expressions

NOT =

Section 4.2, cont'd

6. Definitely read this section of the book.

C. *Sections 4.3 and 4.4: The If Statement*

1. These sections have some additional examples and explanation.

2. Worth the read.

D. *Sections 4.5 and 4.6: Decision Steps in Algorithms, More Problem Solving*

1. Some useful examples presented here.

2. Thorough reading of these sections not as important as preceding sections.

E. *Section 4.7: Nested if Statements and Multiple-Alternative Decisions*

1. Covers more advanced use if `if`.

2. Definitely worth reading.

F. *Section 4.8: The switch Statement*

1. Explains `switch` statement you need in Lab 3.

2. In lab, you'll use switch with integer values, instead of the char values shown in book.

3. See comparative `if`, `switch` example below.

G. *Section 4.9: Common Programming Errors*

1. A short bit, worth the read.

V. Using Elseless If Statements with Mid-Function Return Statements

A. The examples we've seen have shown a number of different ways to use conditional statements.

V. Using Elseless If Statements with Mid-Function Return Statements

A. The examples we've seen have shown a number of different ways to use conditional statements.

B. Another way you might find useful in programming assignment 2 involves the use of a `return` statement in the middle of a function.

Elseless If with Return, cont'd

C. Here's a version of stats program with this idea:

```
/*
 * Consider any negative input to be an error
 */
```

```
/*
 * Consider any negative input to be an error
 */
if ((x1 < 0) || (x2 < 0) || (x3 < 0)) {
```

```
/*
 * Consider any negative input to be an error
 */
if ((x1 < 0) || (x2 < 0) || (x3 < 0)) {
    printf("\nAll inputs must be non-negative
    return 0;
}
```

```
/*
 * Consider any negative input to be an error
 */
if ((x1 < 0) || (x2 < 0) || (x3 < 0)) {
    printf("\nAll inputs must be non-negative
    return 0;
}

/* The else is now gone. */

/* Compute and output the results. */
printf("Sum = %f\n", ... )
printf("Mean = %f\n", ... )
printf("Standard Deviation = %f\n\n", ...
```

VI. Example Showing Equivalent Alternate Uses of If and Switch Statements.

A. Book explains `switch` statement.

VI. Example Showing Equivalent Alternate Uses of If and Switch Statements.

A. Book explains `switch` statement.

B. Can make program more clear or efficient.

VI. Example Showing Equivalent Alternate Uses of If and Switch Statements.

A. Book explains `switch` statement.

B. Can make program more clear or efficient.

C. Here is a comparative example ...

```
/****
 *
 * This program illustrates equivalent uses of if,
 * if_else and switch statements.
 *
 *    . . .
 *
 */
```

```
int main() {
    use_if_else();
    use_if_return();
    use_switch_break();
    use_switch_return();
    return 0;
}
```

```
int main() {
    use_if_else();          /* if-else logic */
    use_if_return();
    use_switch_break();
    use_switch_return();
    return 0;
}
```

```
int main() {
    use_if_else();          /* if-else logic */
    use_if_return();        /* if, return logic */
    use_switch_break();
    use_switch_return();
    return 0;
}
```

```
int main() {
    use_if_else();          /* if-else logic */
    use_if_return();        /* if, return logic */
    use_switch_break();     /* switch, break logic */
    use_switch_return();
    return 0;
}
```

```
int main() {
    use_if_else();          /* if-else logic */
    use_if_return();        /* if, return logic */
    use_switch_break();     /* switch, break logic */
    use_switch_return();    /* switch, return logic */
    return 0;
}
```

same comment for all four functions


```
/**
 * Input a number between 1 and 3 from the
 * terminal.  Do some different calculation based
 * on the value of the number.  If the number is
 * not in the range 1 through 3, output an error.
 */
```

function use_if_else


```
void use_if_else() {
    int number;
    printf("Please input a number between 1 and 3: "
    scanf("%d", &number);
```

function use_if_else


```
    if (number == 1) {
        /* Do processing for input 1 ... */

    }
    else if (number == 2) {
        /* Do processing for input 2 ... */

    }
    else if (number == 3) {
        /* Do processing for input 3 ... */

    }
    else {
        /* Print error message. */
    }
}
```

function use_if_return


```
    if (number == 1) {
        /* Do processing for input 1 ... */
        return;
    }
    if (number == 2) {
        /* Do processing for input 2 ... */
        return;
    }
    if (number == 3) {
        /* Do processing for input 3 ... */
        return
    }

    /* Print error message. */
}
```

function use_switch_break


```
    switch (number) {
    case 1:
        /* Do processing for input 1 ... */
        break;
    case 2:
        /* Do processing for input 2 ... */
        break;
    case 3:
        /* Do processing for input 3 ... */
        break;
    default:
        /* Print error message. */
    }
}
```

function use_switch_return


```
    switch (number) {
    case 1:
        /* Do processing for input 1 ... */
        return;
    case 2:
        /* Do processing for input 2 ... */
        return;
    case 3:
        /* Do processing for input 3 ... */
        return;
    default:
        /* Print error message. */
    }
}
```