Chapter 3: Structured Program Development

* Before writing a program to solve a problem, understand the problem and plan for it first.
* Any computing problem can be solved by executing a series of actions in a specific order.
* A procedure for solving a problem in term of
  1. The actions to be executed
  2. The order in which these actions are to be executed
* This procedure is called an algorithm.
* Specifying the order in which statements are to be executed in a computer program is called program control.
* Pseudocode is an artificial and informal language that helps programmers develop algorithms.
* Pseudocode particularly useful for developing algorithm that will be converted in structure programs.
* Pseudocode is similar to everyday English: it is convenient and user-friendly although it is not an actual computer programming language.

* Normally, statements in a program are executed one after the other in the order in which they are written. This is called sequential execution.
* Various statements enable the programmer to specify that the next statement to be executed may be other than the next one in a sequence. This is called transfer of control.
* All programs could be written in terms of only three control structures (and without goto or jump), namely the "sequence structure", the "selection structure" and the "repetition structure".

* A flowchart is a graphical representation of an algorithm or of a portion of an algorithm. Flowcharts are drawn using certain special-purpose symbols such as rectangles, diamonds, ovals, and small circles; these symbols are connected by arrows called flowlines.
* Flowcharts are also useful for developing and representing algorithms.

* In C, "selection structure" are
    1. if (...) {...} : single-selection structure
    2. if (...) {...} else {...} : double-selection structure
    3. switch (...) {...} : multiple-selection structure
* In C, "repetition structure" are
    1. while (...) {...}
    2. do {...} while (...)
    3. for (...) {...}
* That is all there is in C. Each C program is formed by combining as many of each type of control structure as is appropriate for the algorithm the program implements.

The "If" Selection Structure

* It is used to choose among alternative courses of action.
* E.g.

```
        if (grade >= 60)
            printf("Passed\n");
```

* A decision can be made on any expression - if the expression evaluates to zero, it is treated as false, and if the expression evaluates to nonzero, it is treated as true.

The "If/Else" Selection Structure
---
* The "if" selection structure performs an indicated action only when the condition is true; otherwise the action is skipped.
* The "if/else" selection structure allows the programmer to specify that different actions are to be performed when the condition is true than when the condition is false.
* E.g.

```
        if (grade >= 60)
            printf("Passed\n");
        else
            printf("Failed\n");
```

* C provides the conditional operator (?:) which takes three operands.
    1. a condition
    2. the value for the entire expression if the condition if true
    3. the value for the entire expression if the condition if false
* E.g.

```
        printf("%s\n", grade >= 60 ? "Passed" : "Failed");
```

* The format control string of the "printf" contains the conversion specification "%s" for printing a character string.

* Nested "if/else" structures test for multiple cases by placing "if/else" structures inside "if/else" structures.
* E.g.

```
        if (grade >= 90)
            printf("A\n");
        else
            if (grade >= 80)
                printf("B\n");
            else
                if (grade >= 70)
                    printf("C\n");
                else
                    if (grade >= 60)
                        printf("D\n");
                    else
                        printf("F\n");
```

* Or perfer to write as

```
        if (grade >= 90)
            printf("A\n");
        else if (grade >= 80)
            printf("B\n");
        else if (grade >= 70)
            printf("C\n");
        else if (grade >= 60)
            printf("D\n");
        else
            printf("F\n");
```

* To include several statements in the body of an "if" (or body of other structures), enclose the set of statements in braces ("{" and "}").
* A set pf statements contained within a pair of braces is called a compound statement.
* A compound statement can be placed anywhere in a program that a single statement can be placed.
* E.g.

```
        if (grade >= 60)
```

```
            printf("Passed.\n");
        else
        {
            printf("Failed.\n");
            printf("You must take this course again.\n");
        }
```

* Note: Don't place a semicolon after the condition in an if structure.

The While Repetition Structure

* A repetition structure allows the programmer to specify that an action is to be repeated while some condition remains true.
* E.g.

```
        product = 2;

        while (product <= 1000)
            product = 2 * product;
```

* When the condition in the "while" structure becomes false, the repetition is terminated. Program execution continues with the next statement after the "while".

Assignment Operators

* C provides several assignment operators for abbreviating assignment expressions.
* E.g. The statement

```
        c = c + 3;
```

can be abbreviated with the addition assignment operator "+=" as

```
        c += 3;
```

* The "+=" operator adds the value of the expression on the right of the operator to the value of the variable on the left of the operator and stores the result in the variable on the left of the operator.
* Any statement of the form:

```
        variable = variable operator expression;
```

where operator is one of the binary operator "+", "-", "*", "/", or "%", can be written in the form:

```
        variable operator= expression;
```

Increment and Decrement Operators

* C provides the unary increment operator, "++", and the unary decrement operator, "--".
* E.g.

```
        c++;
        ++c;
        d--;
        --d;
```

* If increment or decrement operators are placed before a variable, they are referred to as the preincrement or predecrement operators, respectively.
* If increment or decrement operators are placed after a variable, they are referred to as the postincrement or postdecrement operators, respectively.
* Preincrementing (predecrementing) a variable causes the variable to be incremented (decremented) by 1, then the new value of the variableis used in the expression in which it appears.
* Postincrementing (Postdecrementing) a variable causes the current

value of the variable to be used in the expression in which it appears, then the variable to be incremented (decremented) by 1.
* E.g.

```
#include <stdio.h>

main()
{
    int c;

    c = 5;
    printf("%d\n", c);
    printf("%d\n", c++);
    printf("%d\n\n", c);

    c = 5;
    printf("%d\n", c);
    printf("%d\n", ++c);
    printf("%d\n", c);

    return 0;
}
```

* It is important to note that when incrementing or decrementing a variable in a statement by itself, the preincrement and postincrement forms have the same effect.
* Only a simple variable name may be used as the operand of an increment or decrement operator.

* The data type "float" can be used to handle numbers with decimal points (called floating-point number).
* C provides the unary cast operator (explicit conversion) to produce a floating-point calculation with integer values. E.g.

```
average = (float) total / counter;
```

* To ensure that the operands of a operator are of the same type, the compiler performs an operation called promotion (implicit conversion) on selected operands.

* Execrise
    1. A class of ten students took a quiz. The grades (integers in the range 0 to 100) for this quiz are available to you. Determine the class average on the quiz.
    2. Develop a class averaging program that will process an arbitrary number of grades each time the program is run.
    3. A college offers a course that prepares students for the state licensing exam for real estate brokers. Naturally, the college wants to know how well its students did on the exam. You have been asked to write a program to summarize the results. You have been given a list of these 10 students. Next to each name is written a 1 if the student passed the exam and a 2 if the student failed. Your program should display a summary of the test results indicating the number of students who passed and the number of students who failed. If more than 8 students passed the exam, print the message "Raise tuition".