Evaluating a Formula

- **Problem (The Registrar's Headache):**
  Student's are to be allowed to register for a certain course offered by an outstanding Professor based on their grade point average (x) and a complicated formula given below.

  Write a C program to produce a table for the value produced by the formula for x = 0.00 up to x = 4.00, increasing x by 0.50 each time.

  If the value of the formula is greater than or equal to zero the student is to be admitted into the class. Otherwise, the student can not register for the class.

- **Formula:**

$$y = \frac{x^3 + 7x - 1}{x^2 - \frac{x + 5}{3}}$$

- **Program:**

```
/* program prob2
 * create a table to evaluate a formula y = f(x)
 */
void main()
{
   declare x and y as real variables
   action part of program
}
```

Data Types for Real Numbers

- Data Type **float**:
  "**float**" creates a single precision real variable.

  ```
  float gpa;
  ```

- Data type **double**:
  "**double**" creates a double precision real variable. Double precision numbers can represent larger and smaller numbers than single precision numbers and with more precision (more bits are used to store the number).

  ```
  double gpa;
  ```

- **Return to Our Problem:**

  ```
  /*  program prob2
   *  create a table to evaluate a formula y =  f(x)
   */
  void main()
  {
      double x,y;

      action part of program
  }
  ```

Copyright © December 20, 2004 by Chaim Ziegler, Ph.D.

Setting Up the for loop

- **Another Revision:**

```
/*  program prob2
 *  create a table to evaluate a formula y =  f(x)
 */
void main()
{
    double x,y;

    for (x =  0.00; x < =  4.00; x =  x +  0.50) {
        compute result =  the formula applied to gpa
        print gpa and result
        if result is > =  0 then print "Admit"
    }
}
```

Copyright © December 20, 2004 by Chaim Ziegler, Ph.D.

Writing the Formula in C

- **Formula:**

$$y = \frac{x^3 + 7x - 1}{x^2 - \frac{x + 5}{3}}$$

- **C Statement:**

```
y = x * x * x + 7 * x - 1 / x * x - x + 5 / 3;        {wrong}
```

- **Order (Precedence) of Operations:**
  a) Unary Minus, Unary Plus, + + , --
  b) * , /, %
  c) + , -

- **Associativity of Operations:**
  Associativity determines which of two operators has priority given that they have the same precedence.
  a) For binary operators, associativity is left to right.
  b) for unary operators, associativity is right to left.
  e.g., y = -x+ + ;

- Example:
  Let a = 10; b = 2

$$y = \frac{a}{b + 3}$$

```
y = a / b + 3          {y = 8 wrong}
y = a / (b + 3)        {y = 2 correct}
```

- **Our Formula:**

```
y = (x * x * x + 7 * x - 1) / (x * x - (x + 5) / 3);
```

- **Note:**
  When in doubt, use parentheses!

Copyright © December 20, 2004 by Chaim Ziegler, Ph.D.

Printing Real Numbers

- **printf** specifications for real numbers:

  %f will print a real number in decimal format with six decimal places to the right of the decimal point.

```
x = 1.0;
y = 6.5;
printf("gpa = %f result = %f\n",x,y);
```

  This prints:

```
   gpa = 1.000000 result = 6.500000
```

- **Another Revision:**

```
/*  program prob2
 *  create a table to evaluate a formula y = f(x)
 */
#include < stdio.h>
void main()
{
   double x,y;

   for (x = 0.00; x < = 4.00; x = x + 0.50) {
      y = (x * x * x + 7 * x - 1) / (x * x - (x + 5) / 3);
      printf("gpa = %f result = %f\n",x,y);
      if result is > = 0 then print "Admit"
   }
}
```

The if (Conditional) Statement

- **The if (Conditional) Statement:**

  if (*condition*)
      simple_statement

- **Compound Statement Form:**

  if (*condition*) {
      compound_statement
  }

- **Yet Another Revision:**

```
/*  program prob2
 *  create a table to evaluate a formula y =  f(x)
 */
#include < stdio.h>
void main()
{
    double x,y;

    for (x =  0.00; x < =  4.00; x =  x +  0.50) {
        y =  (x *  x *  x +  7 *  x - 1) / (x *  x - (x +  5) / 3);
        printf("gpa =  %f result =  %f\n",x,y);
        if (y > =  0)
            printf(" Admit\n");
    }
}
```

Copyright © December 20, 2004 by Chaim Ziegler, Ph.D.

The Program So Far

- **Program:**

```
/*  program prob2
 *  create a table to evaluate a formula y =  f(x)
 */
#include < stdio.h>
void main()
{
    double x,y;

    for (x =  0.00; x < =  4.00; x =  x +  0.50) {
        y =  (x *  x *  x +  7 *  x - 1) / (x *  x - (x +  5) / 3);
        printf("gpa =  %f result =  %f\n",x,y);
        if (y > =  0)
            printf(" Admit\n");
    }
}
```

- **Output**:

```
gpa =  0.000000 result =  0.600000
 Admit
gpa =  0.500000 result =  -1.657895
gpa =  1.000000 result =  -7.000000
gpa =  1.500000 result =  154.500000
 Admit
gpa =  2.000000 result =  12.600000
 Admit
gpa =  2.500000 result =  8.566667
 Admit
gpa =  3.000000 result =  7.421052
 Admit
gpa =  3.500000 result =  7.048673
 Admit
gpa =  4.000000 result =  7.000000
 Admit
```

Copyright © December 20, 2004 by Chaim Ziegler, Ph.D.

Creating a Readable Table

- **Table Header:**
  ```
  printf("Table of Function Values\n");
  ```
- **Double Spacing:**
  ```
  printf("Table of Function Values\n\n");
  ```
- **Staying on the Same Line:**
  ```
  printf("gpa =  %f result =  %f",x,y);
  ```
- **Starting a New Line:**
  ```
  printf("\n");
  ```
- **Printing a Table Trailer:**
  ```
  printf("\nThe table is finished!\n");
  ```
- **Column Headings:**
  ```
  printf("Grade Point Average  Value of Formula  Status\n");
  ```

Copyright © December 20, 2004 by Chaim Ziegler, Ph.D.

The Complete Program:

```
/*  program prob2
 *  create a table to evaluate a formula y =  f(x)
 */
#include < stdio.h>
void main()
{
   double x,y;

   printf("Table of Function Values\n\n");
   printf("Grade Point Average  Value of Formula  Status\n");
   for (x =  0.00; x < =  4.00; x =  x +  0.50) {
      y =  (x *  x *  x +  7 *  x - 1) / (x *  x - (x +  5) / 3);
      printf("      %f            %f",x,y);
      if (y > =  0)
         printf("       Admit");
      printf("\n");
   }
   printf("\nThe table is finished!\n");
}
```

- **Output**:

```
Table of Function Values

Grade Point Average  Value of Formula  Status
      0.000000           0.600000       Admit
      0.500000           -1.657895
      1.000000           -7.000000
      1.500000           154.500000       Admit
      2.000000           12.600000       Admit
      2.500000           8.566667       Admit
      3.000000           7.421052       Admit
      3.500000           7.048673       Admit
      4.000000           7.000000       Admit

The table is finished!
```

Aligning Columns

- **The TAB Character "\t":**
  Inserting the tab character "\t" into the format string causes the cursor to tab to the next tab position before printing.

- **Final Version:**

```
/*  program prob2
 *  create a table to evaluate a formula y =  f(x)
 */
#include < stdio.h>
void main()
{
   double x,y;

   printf("\t\t\tTable of Function Values\n\n");
   printf("Grade Point Average\tValue of Formula\tStatus\n");
   for (x =  0.00; x < =  4.00; x =  x +  0.50) {
      y =  (x *  x *  x +  7 *  x - 1) / (x *  x - (x +  5) / 3);
      printf("%f\t\t%f",x,y);
      if (y > =  0)
         printf("\t\tAdmit");
      printf("\n");
   }
   printf("\nThe table is finished!\n");
}
```

- **Output**:

```
         Table of Function Values

Grade Point Average  Value of Formula  Status
0.000000             0.600000          Admit
0.500000             -1.657895
1.000000             -7.000000
1.500000             154.500000        Admit
2.000000             12.600000         Admit
2.500000             8.566667          Admit
3.000000             7.421052          Admit
3.500000             7.048673          Admit
4.000000             7.000000          Admit

The table is finished!
```

Escape Sequences

- In C, the computer uses the \ (backslash) symbol as an **escape character**. It tells the computer to treat the next character as special.

- The combination of the \ (backslash) followed by a character is known as an **escape sequence**.

- **Common Escape Sequences in C:**
  \n - the newline character (linefeed)
  \t - the tab character
  \b - the backspace character
  \" - the double quote character
  \\ - the backslash character
  \% - the percent sign character (some compilers use %%)
  \0 - the null character
  \a - BEL (makes an audible signal)
  \f - formfeed (new page on printer)
  \r - carriage return (return to beginning of line)

- Example:

```
printf("\% \\ \"\n");

% \ "
```

Compound Assignment Operators

- + = , -= , * = , /= , % =

  gpa = gpa + 0.50; can be written as gpa + = 0.50;
  gpa = gpa - 0.50; can be written as gpa -= 0.50;
  gpa = gpa * 0.50; can be written as gpa * = 0.50;
  gpa = gpa / 0.50; can be written as gpa /= 0.50;
  num = num % 7; can be written as num % = 7;

- Note:

  ```
  number = number + 1;  is equivalent to
  number + = 1;         is equivalent to
  number+ + ;           is equivalent to
  + + number;
  ```

Precedence of Arithmetic, Assignment, & Relational Operators

Precedence	Associativity
a) Unary Minus, Unary Plus, + + , --	right to left
b) * , /, %	left to right
c) + , -	left to right
d) < , < = , > , > =	left to right
e) = = , !=	left to right
f) = , + = , -= , * = , /= , %=	right to left

Scientific Notation

- Scientific notation allows a real number to be represented by its significant digits times a power of ten.

- In C, real numbers (data types float and double) are stored as floating point numbers.

- Floating point numbers are stored a **mantissa** plus an **exponent**. The mantissa represents the significant digits of the number with 1 digit to the left of the decimal point. The exponent tells how far and in which direction the decimal point must move (float) to get the actual value.

- Examples:

  0.50 = 5.000000e-01 = 5×10^{-1}

  6000 = 6.000000e+ 03 = 6×10^{3}

  3.25 = 3.250000e+ 00 = 3.25×10^{0}

  -0.02 = -2.000000e-02 = -2.0×10^{-2}

- **Printing in Scientific Notation:**
  - To print real numbers in scientific notation, use the **%e** conversion specification within the format string.
  - The number will print with a mantissa that has 1 digit to the left of the decimal point and 6 digits to the right of the decimal point.
  - The exponent prints as "e" followed by a sign followed by a 2 digit exponent value.

- **Example:**

```
num = 0.50;
printf("%e", num);
```

Mixed Mode Arithmetic

- An arithmetic expression that applies an operator to two operands of the same type produces a result of that type.

- In C, it is possible to perform arithmetic operations on operands of different types. This is called **mixed mode arithmetic.**

- When an operator is applied to two operands of different types, C uses its rule of automatic type conversion to convert the value of the more restrictive type to that of the less restrictive type (e.g., "int" is more restrictive than "double" or "float"

- Example

```
double real_num, real_sum;
int int_num, int_sum;

int_num = 5;
real_num = 5.6;
int_sum = int_num + real_num;
real_sum = int_num + real_num;
printf("int_sum = %d and real_sum = %f\n",
   int_sum,real_sum);
```

- Output:

```
int_sum = 10 and real_sum = 10.600000
```

- Notes:
  - When a real number is assigned to an integer, the fractional part is truncated.

  - For real division, one or both of the operands must be real:

```
real_num = int_num/2;            // wrong
real_num = (double)int_num/2;    // correct (type casting)
```

Standard (Predeclared) Library Functions:

- sqrt(x)
```
real_num =  sqrt(64);      // real_num =  8.0
```

- abs(integer)
```
int_num =  abs(25);        // int_num =  25
int_num =  abs(-25);       // int_num =  25
```

- fabs(realnum)
```
real_num =  fabs(-2.72);  // real_num =  2.72
```

- ceil(realnum)
```
int_num =  ceil(3.54);     // int_num =  4
```

- floor(realnum)
```
int_num =  floor(3.54);    // int_num =  3
```
  (Note: same as int_num = 3.54)

- Note: to round a real number you can write
```
int_num =  real_num +  0.5;
```

- Note: to use the above library functions, you must include the following directive:

```
#include < math.h>
```

Data Type char

- A variable of data type **char** stores a single character.
- Example

```
char letter;

letter =  'a';           // note the single quotes
```

- To print a variable of type char, use the **%c** conversion specification within the format string.

```
if (letter !=  'b')
    printf("letter has the value %c\n",letter);
```

Debugging

- **Types of Errors:**

  - **Compilation Errors (Syntax Errors)**

```
printit("hello");       // misspelled keyword
x := x + 3;             // wrong assignment symbol
a = (b + 1/3            // missing right parenthesis
x + 3 = 5;              // expression on left side of assignment
printf("error")         // missing semicolon
```

  - **Execution Errors:**

    An error detected once the program is running

    a) uninitialized variables used in assignment statements

```
   x = x + 5;           // where x was never initialized
```

    b) divide by zero

```
   y = (x + 25)/0;
```

  - **Logical Errors:**

    Normally, the hardest type of error to catch.

    The program may run without any execution errors, but the result is wrong!

- **Find the Compilation Errors:**

```
void main()
{
    double x,y;
    print("Grade Point Average\tValue of Formula\tStatus\n");
    for (x = 0.00; x < = 4.00; x = x + 0.50) {
        y = x * x * x + 7 * x - 1) / (x * x - (x + 5) / 3)
        printf("%f\t\t%f",X,Y);
        if (y > = 0) printf("\t\tAdmit");
        printf("\n")
    }
}
```

Copyright © December 20, 2004 by Chaim Ziegler, Ph.D.

Standard Input/Output Streams

- C has three standard I/O streams:
  - **stdin**
    The standard input stream (stdin) is associated with the **keyboard**. Unless specified otherwise, all input to your program comes from the keyboard.
  - **stdout**
    The standard output stream (stdout) is associated with the **monitor**. Unless specified otherwise, all output from your program goes to the monitor.
  - **stderr**
    The standard error stream (stderr) is associated with the **monitor**. You must explicitly send output to **stderr**. It is not automatic.

Copyright © December 20, 2004 by Chaim Ziegler, Ph.D.

Files

- A **file** is a collection of data (usually stored on a disk.)

- An **input file** contains items for input to the program (i.e., items you might otherwise type in from the keyboard during interactive data entry).

- An **output file** contains items output from your program (i.e., items you might otherwise have sent to the screen).

- **Data Type FILE * :**
  - A **file pointer** is used to access a file from a C program.
  - A file pointer has data type **FILE *** .

  e.g.;

```
    FILE * infile;
    FILE * changes;
    FILE * outfile;
    FILE * printer;
```

Copyright © December 20, 2004 by Chaim Ziegler, Ph.D.

Opening and Closing a File

- Before a file can be accessed from a C program, it must be **opened.** After using the file, it must be **closed** before terminating the program.

- **Syntax to Open a File:**
  ***filepointer* = fopen(*filename,mode*);**

  where ***filename*** is a string representing the file to be opened.

- **Mode Settings:**
  "r" - The file is opened for reading. The file should already exist.
  "w" - The file is opened for writing. If the file does not exist, it will be created. If the file exists, It will be opened to the beginning of the file. (Any writing will overwrite what is already in the file.)
  "a" - The file is opened for writing. If the file does not exist, it will be created. If the file exists, It will be opened at the end of the file. Any writing will appended to the end of the file.

- **Examples:**
```
infile =  fopen("letter.dat","r");
changes =  fopen("c:\\hw\\changes.dat","r");
outfile =  fopen("b:newfile.out","w");
printer =  fopen("prn","w");     // opens the printer for output
```

- **Syntax to Close a File:**
  fclose(*filepointer*);

```
fclose(infile);
fclose(changes);
fclose(outfile);
fclose(printer);                 // closes the printer (prn)
```

Output Using fprintf()

- **fprintf() Syntax:**
  fprintf(*destfile*,"*format_string*",< *optional_list_of_variables*>);

- **Examples:**

```
FILE * outfile;
FILE * printer;
int num =  5;

outfile =  fopen("b:newfile.out","w");
printer =  fopen("prn","w");

fprintf(outfile,"Output of My Program\n");
fprintf(printer,"The number is %d\n",num);
fprintf(stdout,"The number is %d\n",num);

fclose(outfile);
fclose(printer);

NOTE: DO NOT OPEN OR CLOSE stdout
```

Copyright © December 20, 2004 by Chaim Ziegler, Ph.D.

- **Example:**

```
/*  program prob2
 *  create a table to evaluate a formula y =  f(x)
 */
#include < stdio.h>
void main()
{
    double x,y;
    FILE * outfile;

    outfile =  fopen("b:prob2.out","w");
//  outfile =  stdout;                //un-comment for testing pgm

    fprintf(outfile,"\t\t\tTable of Function Values\n\n");
    fprintf(outfile,"Grade Point Average\tValue of Formula\tStatus\n");
    for (x =  0.00; x < =  4.00; x =  x +  0.50) {
        y =  (x * x * x +  7 * x - 1) / (x * x - (x +  5) / 3);
        fprintf(outfile,"%f\t\t%f",x,y);
        if (y > =  0)
            fprintf(outfile,"\t\tAdmit");
        fprintf(outfile,"\n");
    }
    fprintf(outfile,"\nThe table is finished!\n");

    fclose(outfile);
}
```