1 quadratic.c

#include <stdio.h>
#include <math.h>

main () { 
    float a, b, c; /* input variables */
    float x1, x2; /* roots */
    float d; /* discriminant */

    /* Prompt the user for the coefficients */
    printf("Enter a, b, c: ");
    scanf("%f %f %f", &a, &b, &c);

    d = b * b - 4 * a * c;

    if (d == 0) {
        /* both roots are the same */
        x1 = -b / (2 * a);
        printf("Both roots are %5.2f.\n", x1);
    } else {
        if (d > 0) {
            /* real roots */
            x1 = (-b - sqrt(d)) / (2 * a);
            x2 = (-b + sqrt(d)) / (2 * a);
            printf("The roots are %5.2f and %5.2f.\n", x1, x2);
        } else {
            /* complex roots */
            x1 = -b / (2 * a); /* real part */
            x2 = sqrt(-d) / (2 * a); /* imaginary part */
            printf("The roots are %5.2f +/- %5.2fi.\n", x1, x2);
        } /* end if d > 0 */
    } /* end if d == 0 */
}

Computation structures:
• printf / scanf
• “if-then-else" construct.
• if nesting.
• Indentation.
• end if comments
• math library

2 year.c

/*
 year.c

 This is an example of a doubly-nested loop (one for each calendar, and another to step through calendars). Uses switch statement to test which month the outer loop is processing. Finally, uses procedural abstraction to create a "print_month" primitive.
*/

#include <stdio.h>

/* define constants for computation */
#define DAYS_IN_WEEK 7
#define MONTHS_IN_YEAR 12
#define JANUARY 1
#define FEBRUARY 2
#define MARCH 3
#define APRIL 4
#define MAY 5
#define JUNE 6
#define JULY 7
#define AUGUST 8
#define SEPTEMBER 9
#define OCTOBER 10
#define NOVEMBER 11
#define DECEMBER 12
#define SHORT_MONTH 28
#define MEDIUM_MONTH 30
#define LONG_MONTH 31
void print_month(int, int);

int main ()
{
    int weekDay;
    int i;

    printf("Enter a day of the week (0-6): ");
    scanf("%d", &weekDay);

    /* check for errors, exit if found */
    if ((weekDay < 0) || (weekDay > 6)) {
        printf("Error in Day of Week choice. Exiting.\n");
        exit(1);
    }

    /* outer loop: iterate over the months of the year */
    for (i = 1; i <= MONTHS_IN_YEAR; i++) {

        printf("\n");
        /* use a switch statement to decide which month we’re on */
        switch(i) {
            /* short months (28 days) */
            case FEBRUARY:
                print_month(weekDay, SHORT_MONTH);
                weekDay = (weekDay + SHORT_MONTH) % DAYS_IN_WEEK;
                break;

            /* medium months (30 days) */
            case APRIL:
            case JUNE:
            case SEPTEMBER:
            case NOVEMBER:
                print_month(weekDay, MEDIUM_MONTH);
                weekDay = (weekDay + MEDIUM_MONTH) % DAYS_IN_WEEK;
                break;

            /* long months (31 days) */
            case JANUARY:
            case MARCH:
            case MAY:
            case JULY:
            case AUGUST:
            case OCTOBER:
            case DECEMBER:
                print_month(weekDay, LONG_MONTH);
            default:
                print_month(weekDay, LONG_MONTH);

        }
    }
}
weekDay = (weekDay + LONG_MONTH) % DAYS_IN_WEEK;
break;
}
}
return (0);
}

void print_month(int weekDay, int monthDay)
{
    int i;
    /* print header line */
    printf(" Su Mo Tu We Th Fr Sa\n");
    /* prints spaces before the first date is printed */
    for (i = 0; i < weekDay; i++) {
        printf(" ");
    }
    /* prints the dates */
    for (i = 1; i <= monthDay; i++) {
        printf("%3d", i);
        if (!((i+weekDay) % DAYS_IN_WEEK))
            printf("\n");
    }
    printf("\n");
}

Computation structures / concepts:

- Procedural abstraction.
- Loop nesting.
- Switch statement.
- Use of fall through inside the switch statement.
- Use of break inside the switch statement.

3 matmul.c
/*
matmul.c

Matrix Multiply

This code reads in two matrices of integers, and then computes their product. Demonstrates 2-dimensional arrays.

#include <stdio.h>
#define SIZE 5

int main()
{
  int matrixA[SIZE][SIZE];
  int matrixB[SIZE][SIZE];
  int matrixC[SIZE][SIZE];
  int i, j, k;

  /* Read in the first matrix */
  printf("Enter first matrix of %d x %d integers:\n", SIZE, SIZE);
  for (i = 0; i < SIZE; i++) {
    for (j = 0; j < SIZE; j++) {
      scanf("%d", &matrixA[i][j]);
    }
  }

  /* Read in the second matrix */
  printf("Enter second matrix of %d x %d integers:\n");
  for (i = 0; i < SIZE; i++) {
    for (j = 0; j < SIZE; j++) {
      scanf("%d", &matrixB[i][j]);
    }
  }

  /* Perform matrix multiply */
  for (i = 0; i < SIZE; i++) {
    for (j = 0; j < SIZE; j++) {
      /* compute 1 element of the matrix multiplication */
      matrixC[i][j] = 0;
      for (k = 0; k < SIZE; k++) {
        matrixC[i][j] += (matrixA[i][k] * matrixB[k][j]);
      }
    }
  }
}
} /* Print the result matrix */
printf("Result:\n");
for (i = 0; i < SIZE; i++) {
    for (j = 0; j < SIZE; j++) {
        printf("%d ", matrixC[i][j]);
    }
    printf("\n");
}

Computation structures / concepts:

- Matrix multiplication algorithm
- Multi-dimensional arrays
- Indexing inside nested loops