Control Flow

adopted from KNK C Programming : A Modern Approach

switch3.c

```
char op;    float i, j;
printf ("Enter a simple expression (Eg. i / j)\n");
scanf ("%f %c %f", &i, &op, &j);
switch (op) {
    case '+': printf ("%.2f\n", i + j ); break;
    case '-': printf ("%.2f\n", i - j ); break;
    case '*': printf ("%.2f\n", i * j ); break;
    case '/':
        if (j != 0)  printf ("%.2f\n", i / j );
        else         printf("Cannot divide by zero\n");
        break;
    default:
        printf ("Available operators are +, -, *, /");
        break;
}
```

Program: Printing a Table of Squares

- The `square.c` program uses a while statement to print a table of squares.
- The user specifies the number of entries in the table:

```
This program prints a table of squares.
Enter number of entries in table: 5
          1         1
          2         4
          3         9
          4        16
          5        25
```

square.c

```
/* Prints a table of squares using a while statement */

#include <stdio.h>

int main(void)
{
  int i, n;

  printf("This program prints a table of squares.\n");
  printf("Enter number of entries in table: ");
  scanf("%d", &n);

  i = 1;
  while (i <= n) {
    printf("%10d%10d\n", i, i * i);
    i++;
  }

  return 0;
}
```

Program: Summing a Series of Numbers

- The `sum.c` program sums a series of integers entered by the user:

```
This program sums a series of integers.
Enter integers (0 to terminate): 8 23 71 5 0
The sum is: 107
```

- The program will need a loop that uses `scanf` to read a number and then adds the number to a running total.

sum.c

```
/* Sums a series of numbers */

#include <stdio.h>

int main(void)
{
  int n, sum = 0;

  printf("This program sums a series of integers.\n");
  printf("Enter integers (0 to terminate): ");

  scanf("%d", &n);
  while (n != 0) {
    sum += n;
    scanf("%d", &n);
  }
  printf("The sum is: %d\n", sum);

  return 0;
}
```

Program: Calculating the Number of Digits in an Integer

- The `numdigits.c` program calculates the number of digits in an integer entered by the user:

```
Enter a nonnegative integer: 60
The number has 2 digit(s).
```

- The program will divide the user's input by 10 repeatedly until it becomes 0; the number of divisions performed is the number of digits.
- Writing this loop as a `do` statement is better than using a `while` statement, because every integer—even 0—has at least one digit.

numdigits.c

```
/* Calculates the number of digits in an integer */

#include <stdio.h>

int main(void)
{
  int digits = 0, n;

  printf("Enter a nonnegative integer: ");
  scanf("%d", &n);

  do {
    n /= 10;
    digits++;
  } while (n > 0);

  printf("The number has %d digit(s).\n", digits);

  return 0;
}
```

Program: Printing a Table of Squares (Revisited)

- The `square.c` program (Section 6.1) can be improved by converting its `while` loop to a `for` loop.

square2.c

```
/* Prints a table of squares using a for statement */

#include <stdio.h>

int main(void)
{
  int i, n;

  printf("This program prints a table of squares.\n");
  printf("Enter number of entries in table: ");
  scanf("%d", &n);

  for (i = 1; i <= n; i++)
    printf("%10d%10d\n", i, i * i);

  return 0;
}
```

Program: Printing a Table of Squares (Revisited)

- C places no restrictions on the three expressions that control the behavior of a `for` statement.
- Although these expressions usually initialize, test, and update the same variable, there's no requirement that they be related in any way.
- The `square3.c` program is equivalent to `square2.c`, but contains a `for` statement that initializes one variable (`square`), tests another (`i`), and increments a third (`odd`).
- The flexibility of the `for` statement can sometimes be useful, but in this case the original program was clearer.

square3.c

```
/* Prints a table of squares using an odd method */

#include <stdio.h>

int main(void)
{
  int i, n, odd, square;

  printf("This program prints a table of squares.\n");
  printf("Enter number of entries in table: ");
  scanf("%d", &n);

  i = 1;
  odd = 3;
  for (square = 1; i <= n; odd += 2) {
    printf("%10d%10d\n", i, square);
    ++i;
    square += odd;
  }

  return 0;
}
```

for-to-while.c

```
/* case 1 */
    for (int i = 10; i > 0; --i)
        printf("T minus %d and counting\n", i);

/* case 2 */
    printf("\n\ncase 2\n");
    for (int i = 10; i > 0; i--)
        printf("T minus %d and counting\n", i);

/* case 3 */
    printf("\n\ncase 3\n");
    int i = 10;
    while (i > 0) {
        printf("T minus %d and counting\n", --i);
     // printf("T minus %d and counting\n", i--);
```

The `continue` Statement: continue1.c

- A loop that uses the `continue` statement:

```
n = 0;
sum = 0;
while (n < 10) {
  scanf("%d", &i);
  if (i == 0)
    continue;
  sum += i;
  n++;
  /* continue jumps to here */
}
```

The `continue` Statement: continue2.c

- The same loop written without using `continue`:

```
n = 0;
sum = 0;
while (n < 10) {
  scanf("%d", &i);
  if (i != 0) {
    sum += i;
    n++;
  }
}
```

The `goto` Statement: goto.c

- If C didn't have a `break` statement, a `goto` statement could be used to exit from a loop:

```
for (d = 2; d < n; d++)
  if (n % d == 0)
    goto done;
done:
if (d < n)
  printf("%d is divisible by %d\n", n, d);
else
  printf("%d is prime\n", n);
```

The goto Statement: goto2.c

```
for (d = 2; d < n; d++)
    if (n % d == 0)
      goto prime;
    else
      goto general;

  printf("Does this line print out?\n");
  printf("if it does not print, then why not?\n");

  general:
  if (d < n)
    printf("%d is divisible by %d\n", n, d);

  prime:
  if (d > n)
    printf("%d is prime\n", n);
```

Program: Balancing a Checkbook

- Many simple interactive programs present the user with a list of commands to choose from.
- Once a command is entered, the program performs the desired action, then prompts the user for another command.
- This process continues until the user selects an “exit” or “quit” command.
- The heart of such a program will be a loop:

```
for (;;) {
   prompt user to enter command;
   read command;
   execute command;
}
```

Program: Balancing a Checkbook

- Executing the command will require a `switch` statement (or cascaded `if` statement):

```
for (;;) {
   prompt user to enter command;
   read command;
   switch (command) {
      case command1: perform operation1; break;
      case command2: perform operation2; break;
        .
        .
        .
      case commandn: perform operationn; break;
      default: print error message; break;
   }
}
```

Program: Balancing a Checkbook

- The `checking.c` program, which maintains a checkbook balance, uses a loop of this type.
- The user is allowed to clear the account balance, credit money to the account, debit money from the account, display the current balance, and exit the program.

Program: Balancing a Checkbook

```
*** ACME checkbook-balancing program ***
Commands: 0=clear, 1=credit, 2=debit, 3=balance, 4=exit

Enter command: 1
Enter amount of credit: 1042.56
Enter command: 2
Enter amount of debit: 133.79
Enter command: 1
Enter amount of credit: 1754.32
Enter command: 2
Enter amount of debit: 1400
Enter command: 2
Enter amount of debit: 68
Enter command: 2
Enter amount of debit: 50
Enter command: 3
Current balance: $1145.09
Enter command: 4
```

checking.c

```
/* Balances a checkbook */

#include <stdio.h>

int main(void)
{
  int cmd;
  float balance = 0.0f, credit, debit;

  printf("*** ACME checkbook-balancing program ***\n");
  printf("Commands: 0=clear, 1=credit, 2=debit, ");
  printf("3=balance, 4=exit\n\n");
  for (;;) {
    printf("Enter command: ");
    scanf("%d", &cmd);
    switch (cmd) {
      case 0:
        balance = 0.0f;
        break;
```

checking.c

```
      case 1:
        printf("Enter amount of credit: ");
        scanf("%f", &credit);
        balance += credit;
        break;
      case 2:
        printf("Enter amount of debit: ");
        scanf("%f", &debit);
        balance -= debit;
        break;
      case 3:
        printf("Current balance: $%.2f\n", balance);
        break;
      case 4:
        return 0;
      default:
        printf("Commands: 0=clear, 1=credit, 2=debit, ");
        printf("3=balance, 4=exit\n\n");
        break;
    }
  }
}
```