

```

        case command2: perform operation2; break;
        .
        .
        case commandn: perform operationn; break;
        default: print error message; break;
    }
}

```

To illustrate this arrangement, let's develop a program that maintains a checkbook balance. The program will offer the user a menu of choices: clear the account balance, credit money to the account, debit money from the account, display the current balance, and exit the program. The choices are represented by the integers 0, 1, 2, 3, and 4, respectively. Here's what a session with the program will look like:

```

*** 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

```

When the user enters the command 4 (exit), the program needs to exit from the switch statement *and* the surrounding loop. The break statement won't help, and we'd prefer not to use a goto statement. Instead, we'll have the program execute a return statement, which will cause the main function to return to the operating system.

```

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");

```