Lecture 12.2

Topics

1. Basic Programming Structures – Revisited
2. Menu Setup with `while` and `do-while` Loops

---

1. Basic Programming Structures – Revisited

Recall that there are 3 programming structures as illustrated in the figure below.

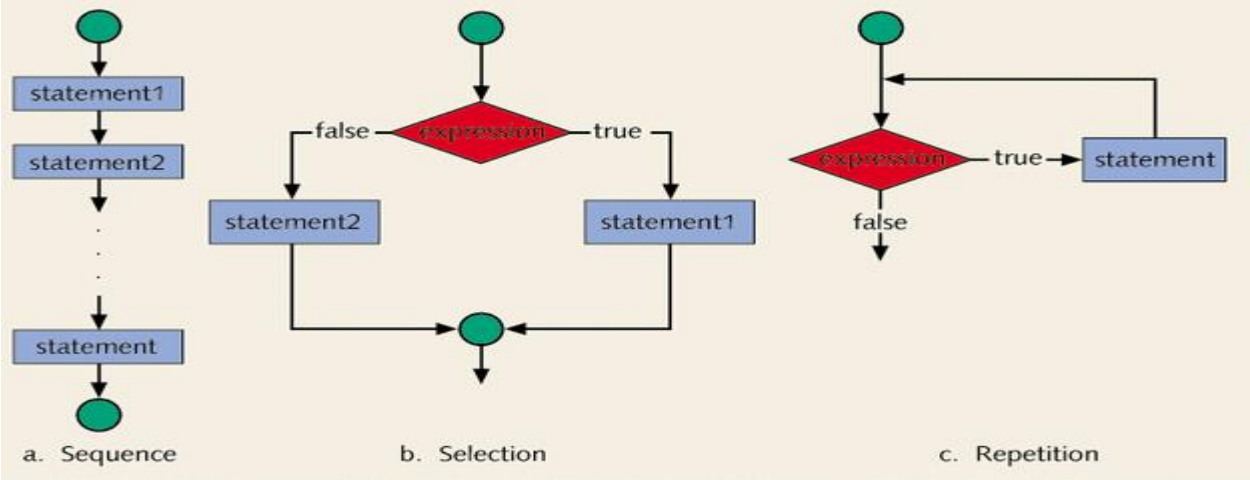

Figure 1 Three basic programming structures

With these 3 basic programming structures, solutions to any given problem will be possible. Recent lectures have been geared toward menu applications, which will have its final implementation with a combination of `switch` and `do-while` loop.

2. Menu Setup with `while` and `do-while` Loops

In previous discussions, the `for` and `while` loops were introduced in which a concept of a menu application was also mentioned.

In this lecture, a formal description of a menu application will be given through the use of a `while` loop and then a `do-while` loop will be taken in as its final form.

2.1 Example

A sample output of a menu application is given below where a program will allow us to select and run the options until we decide to stop.

```
            MENU --

            (1) Add
            (2) Subtract
            (3) Multiply
            (4) Divide
            (5) Quit

      Select and enter an integer for option + ENTER: 1

      Enter first operand: 2
```

```
Enter second operand: 3

2.000000 + 3.000000 --> 5.000000

        MENU --

        (1) Add
        (2) Subtract
        (3) Multiply
        (4) Divide
        (5) Quit

Select and enter an integer for option + ENTER: 9

Invalid Option!

        MENU --

        (1) Add
        (2) Subtract
        (3) Multiply
        (4) Divide
        (5) Quit

Select and enter an integer for option + ENTER: 3

Enter first operand: 5

Enter second operand: 6

5.000000 * 6.000000 --> 30.000000

        MENU --

        (1) Add
        (2) Subtract
        (3) Multiply
        (4) Divide
        (5) Quit

Select and enter an integer for option + ENTER: 5

It is fun! Bye ...
```

Discussion will be given in class.

2.2 Menu With `while` Loop

Consider that code given below.

```
int main() {
  int option;
```

```
  int count;

  option = 0;
  while (option != 2) {
    printf("\nMENU"\
           "\n1. Performing Task"\
           "\n2. Quit");
    printf("\nEnter the run option (1 : run or 2 : stop): ");
    scanf("%d", &option);

    if (option == 1) {
      // TODO
      //   Performing the required task
    } else if (option == 2) {
      // stop
      printf("\n\QUIT!\n");
    } else {
      // wrong option
      printf("\n\tWRONG OPTION!\n");
    }
  }


  return 0;
}
```

In the above code fragment, the combination of the initialization and the **`while`** loop would provide the menu operations just fine.

2.2 Menu as Functions – `menuWhile()` and `menuDoWhile()`

However, to remove the "*decision of initializing the **`option`** with zero (**`0`**)*", a **`do-while`** loop is to be described and used as given below with the 2 functions of `menuWhile()` and `menuDoWhile()`.

```
void menuWhile() {
  int option;

  option = 0;
  while (option != 2) {
    printf("\nMENU"\
           "\n1. Performing Task"\
           "\n2. Quit");
    printf("\nEnter the run option (1 : run or 2 : stop): ");
    scanf("%d", &option);

    if (option == 1) {
      // TODO
      //   Performing the required task
      printf("\n  Performing the required task!\n");
    } else if (option == 2) {
      // stop
      printf("\n  Quit!\n");
    } else {
      // wrong option
```

```
      printf("\n  WRONG OPTION!\n");
    }
  }
}
```

With the **`while`** loop is to be replaced with the **`do-while`** loop, the code is shown as follows,

```
void menuDoWhile() {
  int option;

  do {
    printf("\nMENU"\
           "\n1. Performing Task"\
           "\n2. Quit");
    printf("\nEnter the option (1 : run or 2 : stop): ");
    scanf("%d", &option);

    if (option == 1) {
      // TODO
      //   Performing the required task
      printf("\n  Performing the required task!\n");
    } else if (option == 2) {
      // stop
      printf("\n  Quit!\n");
    } else {
      // wrong option
      printf("\n  WRONG OPTION!\n");
    }
  } while (option != 2);
}
```

2.3 Final Form of Menu – `menu()`

However, the final form of our menu application is to have the extended **`if-else if-else`** replaced with the **`switch`** within the `menuDoWhile()` function.

The code is given below.

```
void menu() {
  int option;

  do {
    printf("\nMENU"\
           "\n1. Performing Task"\
           "\n2. Quit");
    printf("\nEnter the option (1 : run or 2 : stop): ");
    scanf("%d", &option);

    switch (option == 1) {
      case 1:
        // TODO
        //   Performing the required task
        printf("\n  Performing the required task!\n");
        break;
      case 2:
        // stop
```

```
          printf("\n  Quit!\n");
          break;
        default:
          // wrong option
          printf("\n  WRONG OPTION!\n");
      }
    } while (option != 2);
}
```

A complete sample program is given below.

```
/**
 * Program Name: cis6L122Menu.c
 * Discussion:   Conditioning, Loops & Functions
 * Written By:   Your Name
 * Date:
 */

#include <stdio.h>

void menuWhile(void);

void menuDoWhile(void);

void menu(void);

int main() {
  menu();

  return 0;
}

// Function Definitions

void menu() {
  int option;

  do {
    printf("\nMENU"\
           "\n1. Performing Task"\
           "\n2. Quit");
    printf("\nEnter the option (1 : run or 2 : stop): ");
    scanf("%d", &option);

    switch (option == 1) {
      case 1:
        // TODO
        //   Performing the required task
        printf("\n  Performing the required task!\n");
        break;
      case 2:
        // stop
        printf("\n  Quit!\n");
        break;
```

```
      default:
        // wrong option
        printf("\n  WRONG OPTION!\n");
    }
  } while (option != 2);
}

void menuDoWhile() {
  int option;

  do {
    printf("\nMENU"\
           "\n1. Performing Task"\
           "\n2. Quit");
    printf("\nEnter the option (1 : run or 2 : stop): ");
    scanf("%d", &option);

    if (option == 1) {
      // TODO
      //   Performing the required task
      printf("\n  Performing the required task!\n");
    } else if (option == 2) {
      // stop
      printf("\n  Quit!\n");
    } else {
      // wrong option
      printf("\n  WRONG OPTION!\n");
    }
  } while (option != 2);
}

void menuWhile() {
  int option;

  option = 0;
  while (option != 2) {
    printf("\nMENU"\
           "\n1. Performing Task"\
           "\n2. Quit");
    printf("\nEnter the run option (1 : run or 2 : stop): ");
    scanf("%d", &option);

    if (option == 1) {
      // TODO
      //   Performing the required task
      printf("\n  Performing the required task!\n");
    } else if (option == 2) {
      // stop
      printf("\n  Quit!\n");
    } else {
      // wrong option
      printf("\n  WRONG OPTION!\n");
    }
```

```
  }
}
/* PROGRAM OUTPUT

MENU
1. Performing Task
2. Quit
Enter the option (1 : run or 2 : stop): -1

  WRONG OPTION!

MENU
1. Performing Task
2. Quit
Enter the option (1 : run or 2 : stop): 1

  Performing the required task!

MENU
1. Performing Task
2. Quit
Enter the option (1 : run or 2 : stop): 4

  WRONG OPTION!

MENU
1. Performing Task
2. Quit
Enter the option (1 : run or 2 : stop): 1

  Performing the required task!

MENU
1. Performing Task
2. Quit
Enter the option (1 : run or 2 : stop): 2

  WRONG OPTION!
*/
```