C Programming #4

Repetition

International College, KMITL

Basic Loop Structures

- Constructing a repeating section of code requires that four elements be present:
  - Repetition statement
    - while statement
    - for statement
    - do-while statement
  - Condition
  - A statement that initially sets the condition being tested
  - A statement within the repeating section of code that alters the condition so that it eventually becomes false

The while Statement

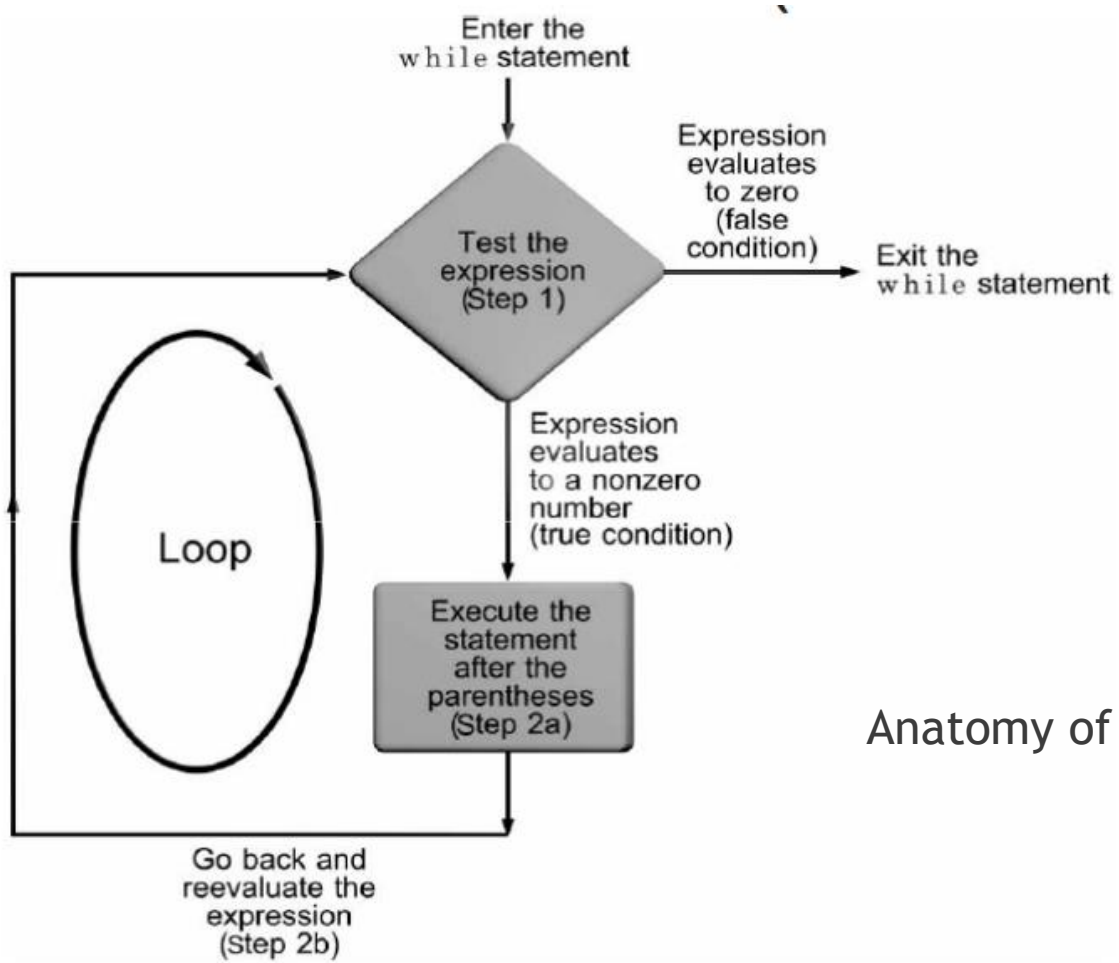
- The general form of the while statement is

```
while (expression) statement;
```

- The transfer of control back to the start of a while statement to reevaluate the expression is known a program loop
- The following is a valid but infinite loop:

```
while (count <= 10)
        printf("%d ",count);
```

The while Statement (continued)



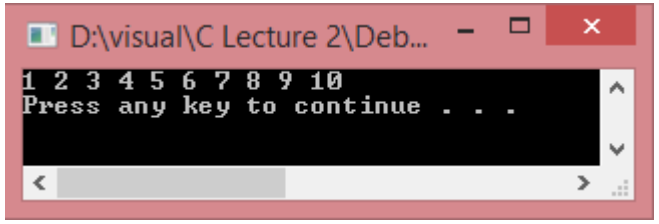
Anatomy of a while loop

The while Statement (continued)

```c
#include <stdio.h>

void main() {
    int count = 1;
    while (count <= 10)
    {
        printf("%d ", count);
        count++;
    }
    printf("\n");

    system("pause");
}
```



The while Statement (continued)

```c
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>

void main() {
    int count = 1;
    float num;

    while (count <= 5)
    {
        printf("Enter a number: ");
        scanf("%f", &num);
        printf("The number entered is %f\n\n", num);
        count++;
    }

    system("pause");
}
```

Check Point #1

The break and continue Statements

- A break forces an immediate exit from while, switch, for, and do-while statements only

```c
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>

void main() {
    int num;

    while (1)
    {
        printf("Enter a number: ");
        scanf("%d", &num);
        if (num > 9)
            break; /* break out of the loop */
    }
    system("pause");
}
```

D:\visual\C Lecture 2\Debug\...

```
Enter a number: 2
Enter a number: 1
Enter a number: 4
Enter a number: 3
Enter a number: 2
Enter a number: 7
Enter a number: 5
Enter a number: 12
Press any key to continue . . .
```

The break and continue Statements (continued)

- The continue applies to loops only; when a continue statement is encountered in a loop, the next iteration of the loop begins immediately

```c
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>

void main() {
    int num, total = 0;

    while (1)
    {
        printf("Enter a number: ");
        scanf("%d", &num);
        if (num == 0)
            break;
        if (num < 0 || num > 100)
            continue;
        total += num;
        printf("Total = %d\n\n", total);
    }
    printf("Total = %d\n", total);

    system("pause");
}
```

D:\visual\C Lecture 2\Debug\...

```
Enter a number: 5
Total = 5

Enter a number: 2
Total = 7

Enter a number: 104
Enter a number: 4
Total = 11

Enter a number: -1
Enter a number: 0
Total = 11
Press any key to continue . . .
```

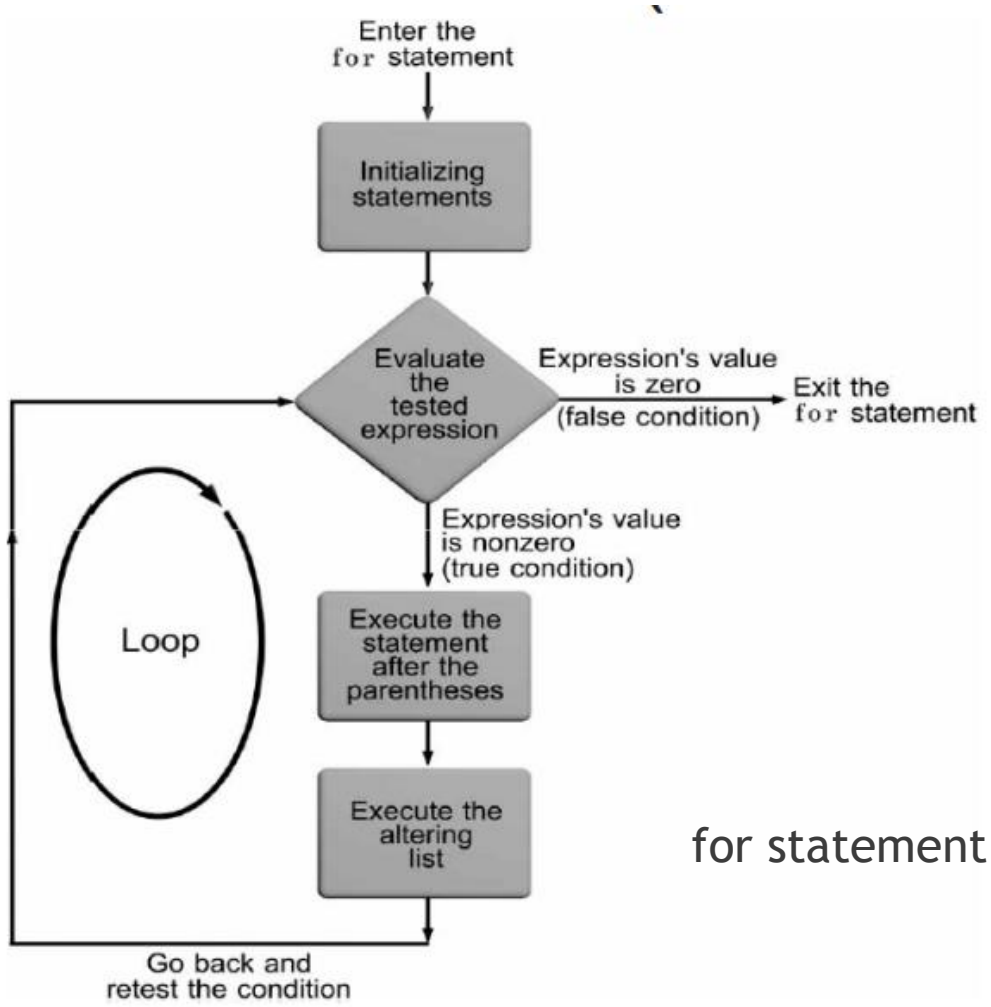
The for Statement

- The for statement combines all four elements required to easily produce a loop on the same line

```
for (initializing list; tested expression; altering list)
        statement;
```

- This statement does not require that any of the items in parentheses be present or that they actually be used for initializing or altering the values in the expression statements
  - However, the two semicolons must be present
    - for (; count <= 20;) is valid
    - Omitting tested expression results in infinite loop

The for Statement (continued)



for statement flow of control

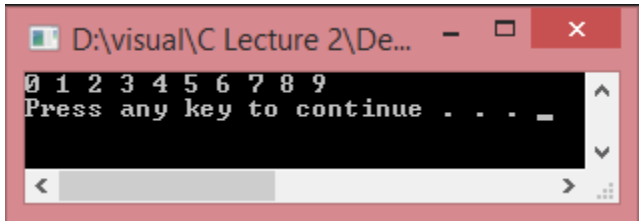
The for Statement (continued)

```c
#include <stdio.h>

void main() {
    int i;

    for (i = 0; i < 10; i++)
    {
        printf("%d ", i);
    }
    printf("\n");

    system("pause");
}
```



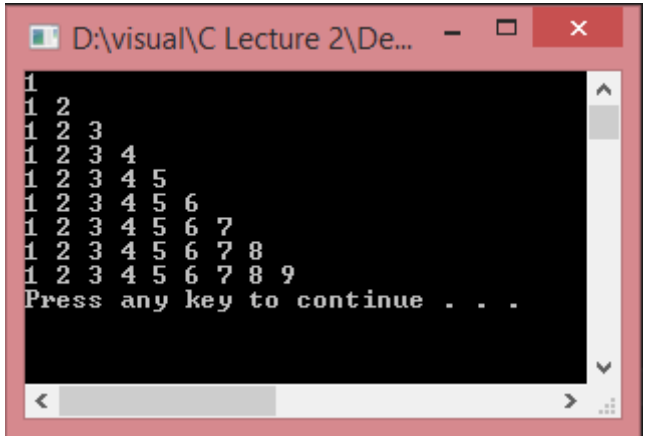
The for Statement (continued)

```c
#include <stdio.h>

void main() {
    int i, j;

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

    system("pause");
}
```



Check Point #2

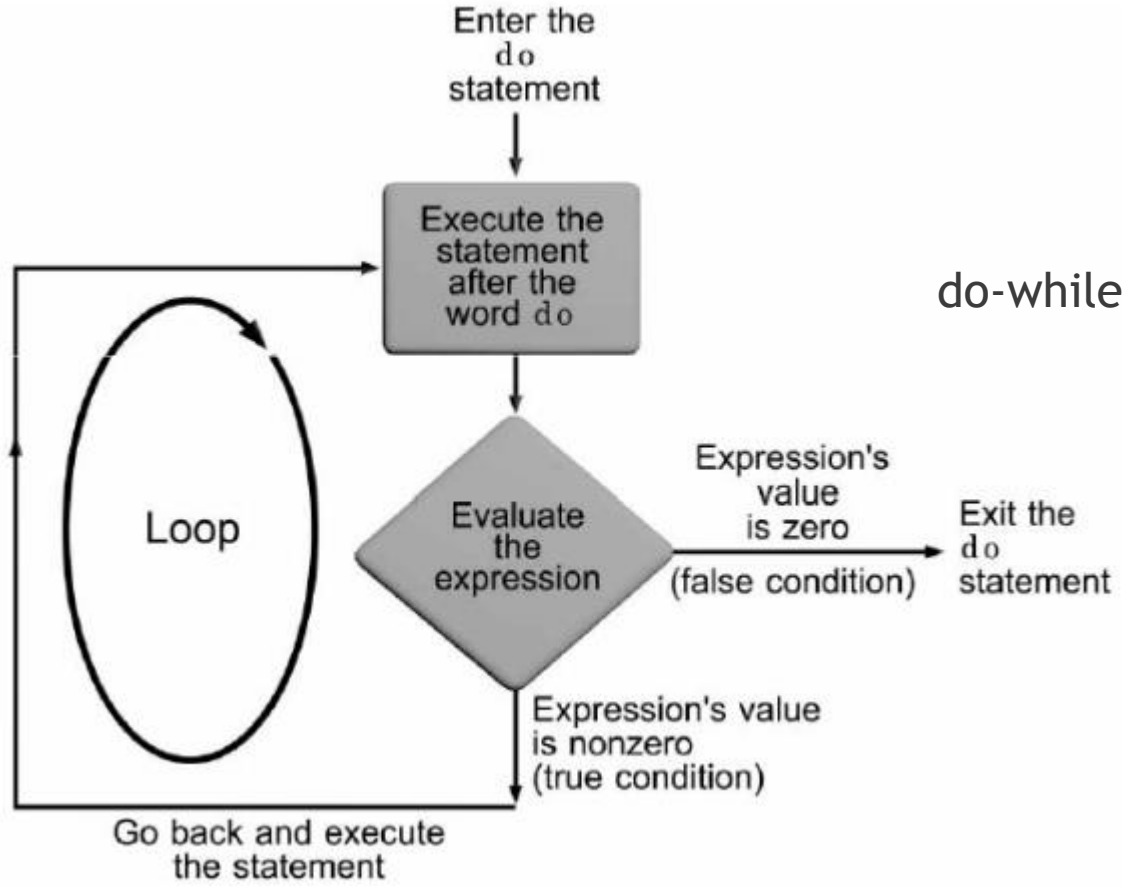
The do-while Statement

- The general form of the do statement is

```
do
        statement;
while (expression);
```

- do-while is a posttest loop
- One type of application is ideally suited for a posttest loop:
  - Input data validation application

The do-while Statement (continued)



do-while statement flow of control

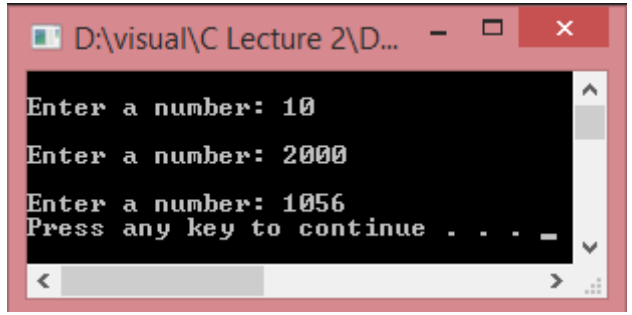
The do-while Statement (continued)

```c
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>

void main() {
    int num;

    do
    {
        printf("\nEnter a number: ");
        scanf("%d", &num);
    } while (num < 1000 || num > 1999);

    system("pause");
}
```



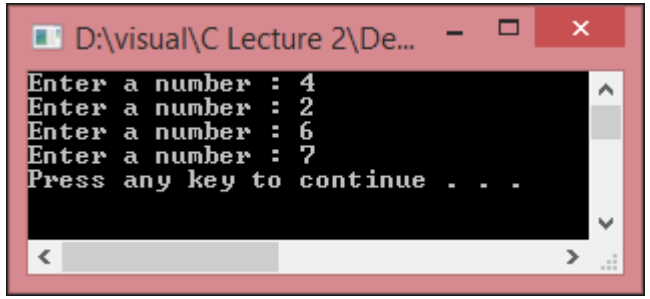
The do-while Statement (continued)

```c
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>

void main() {
    int num;

    do {
        printf("Enter a number : ");
        scanf("%d", &num);
    } while (num % 2 == 0);

    system("pause");
}
```



Check Point #3