Fundamentals of Computing

Week 5

What have we done so far

What we have done so far …

Wk	Lecture
1	Introduction, MDF, Module Guide, Assessment Variables
2	C – variables, statements
3, 4	C - Input and Output C - Decision statements

Today …

C – loops,

C - while and do-while loops

Learning Outcomes

1. Learn about loops and their importance.
2. Learn about while and do-while loops
3. Be able to write programs using the while and do-while loops.

Repetition

C provides three types of loops:

- `while` loop
- `do while` loop
- `for` loop

The loops continue to repeat basically until certain conditions are met.

Repetition – `while` loop

The syntax of `while` loop:

```
while (condition) {
  /* block of statements to execute */
}
```

The while loop continues to loop until the condition becomes false. The condition is tested upon entering the loop. Any logical construction can be used in this context.

Repetition – `while` loop

Example 1: Determine the sum of the first 10 positive numbers.

```
#include "stdio.h"
#include "conio.h"

int main() {
  int sum = 0, number = 0;
  while (number <= 10) {
    sum = sum + number;
    number = number + 1;
  }
  printf("sum=%d\n", sum);
  getch();
  return 0;
}
```

Repetition – `while` loop

Example 2: Calculate the sum of the numbers read from the keyboard. Stop the program by entering the number 0.

```
#include "stdio.h"
#include "conio.h"

int main() {
  int number = -1, sum = 0;
  while (number != 0) { //the loop stops when a 0 is entered
     printf("number = ");
     scanf("%d", &number); // read the number
     sum = sum + number; // add the number to the sum
  }
  printf("sum=%d \n", sum);
  getch(); return 0;
}
```

Repetition – `while` loop

Example 3: Print the following sequence of signs of a given length.

+-+-+-+-+-+-+

```
#include "stdio.h"
#include "conio.h"
int main() {
  int length, i = 0;
  printf("length = ");
  scanf("%d", &length); // read the variable length
  while (i < length) {
    if(i % 2 == 0) {
      printf("+");
    } else {
      printf("-");
    }
    i = i + 1;
  }
  getch(); return 0; }
```

Repetition – do-while loop

The syntax of do-while loop:

```
do {
  /* block of statements to execute */
} while (condition);
```

The do-while loop continues to loop until the condition becomes false. Any logical construction can be used in this context.

Repetition – do-while loop

Example 1: Determine the sum of the first 10 positive numbers.

```
#include "stdio.h"
#include "conio.h"

int main() {
  int sum = 0, i = 1;

  do{
    sum += i;
    i++;
  } while(i < 10);

  printf("sum=%d \n", sum);
  getch(); return 0;
}
```

Repetition – `for` loop

The syntax of `for` loop:

```
for (expression1; expression2; expression3) {
     /* block of statements to execute */
 }
```

`expression1` is the **variable initialisation**. It allows you to give a value to one or more existing variables or initialise new ones.

Repetition – `for` loop

The syntax of `for` loop:

```
for (expression1; expression2; expression3) {
     /* block of statements to execute */
 }
```

`expression2` is the **condition**. It tells the program that while the conditional expression is true the loop should continue to repeat itself.

Repetition – `for` loop

The syntax of `for` loop:

```
for (expression1; expression2; expression3) {
     /* block of statements to execute */
 }
```

`expression3` is the **variable update** section. It's the easiest way for a for loop to handle changing of variables.

Repetition – `for` loop

The syntax of `for` loop:

```
for (expression1; expression2; expression3) {
     /* block of statements to execute */
 }
```

A semicolon separates each of these sections, that is important.
Every expression may be empty, though the semicolons still have to be there.
If the condition is empty, it is evaluated as true and the loop will repeat until something else stops it.

Repetition – for loop

Example 1: Determine the sum of the first 10 positive numbers.

```
#include "stdio.h"
#include "conio.h"

int main() {
  int sum = 0, i;
  for (i = 0; i <= 10; i++) {
    sum += i;
  }
  printf("sum=%d \n", sum);
  getch(); return 0;
}
```

Repetition – for loop

Example 2: Calculate the sum of a certain number of numbers read from the keyboard.

```
#include "stdio.h"
#include "conio.h"

int main() {
  int length, i, number, sum = 0;
  printf("length = ");
  scanf("%d", &length); // read the variable  length
  for (i = 0; i < length; i++) {
     printf("number = ");
     scanf("%d ", &number); // read the number
     sum += number; // add the number to the sum
  }
  printf("sum=%d \n", sum);
  getch(); return 0;
}
```

Repetition – `for` loop

Example 3: Print the following sequence of signs of a given length.

+-+-+-+-+-+-+

```
#include "stdio.h"
#include "conio.h"
int main() {
  int length, i;
  printf("length = ");
  scanf("%d", &length); // read the variable length
  for (i = 0; i < length; i++) {
     if(i%2 == 0) printf("+");
     else printf("-");
  }
  getch(); return 0;
}
```

Tasks

1. Write a program containing a loop that runs until told to stop. Use a control variable (`short`) to decide when to stop. *Hint: 1 – true, 0 – false*

2. Extend program #1 to include a menu with 2 options (A, B). Loop should run until option B is selected. Each choice should be fed back to screen.
   - Use a char to record the user's choice.
   - Make use of switch.

3. Transform #2 into a Calculator (+, –, *, /) that runs continuously until an appropriate Exit option is chosen.

Tasks

4.Write a program that offers three options:

a. Sum (unlimited numbers, read until 0 is entered)
- 1 + 3 + 5 + 2 + 5 + 0 = 16
- 1 + 1 + 5 + 0 = 7
- …

b.Average (unlimited numbers, read until 0 is entered)
- (3 + 3 + 2 + 4) / 4 = 3
- (1 + 1 + 1) / 3 = 1
- …

c.Exit