Computer Programming using C
Lecture 4

Dr Julian Miller
Room P/M/101
Ext 2383; E-mail: jfm@ohm
http://www.elec.york.ac.uk/intsys/users/jfm7/

Introduction

• Revision (in the lab last week)
• Answers to exercises
• In the lab this week
In the lab last week (lab 3)

• Iteration
  – The **while** loop repeats instructions every time until the stopping condition is satisfied
  – The **do-while** loop works like the while loop except that the stopping condition is at the end so the body of instructions must be carried out once
  – The **for** loop repeats a group of instructions a specific number of times by using a counter

Revision: **while**

```java
while (carry_on) {
  do_this;
  do_that;
  /* and so on */
}
```

As with if statements the loop continues to repeat while the expression `carry_on` is non-zero. As soon as it is zero the program jumps to the next statement after the while loop.
Lab 3

#include <stdio.h>

int main(void)
{
    int number_entered;
    printf("Enter an integer number: ");
    scanf("%d", &number_entered);
    while (number_entered <= 10)
    {
        printf("%d\n", number_entered);
        number_entered = number_entered + 1;
    }
    return 0;
}

Exercise 3.1

• Change lab3.c so that the program starts with the user entered number and doubles it on every iteration. It should stop when the number is greater than or equal to the square of the user entered number.

#include <stdio.h>

int main(void)
{
    int number, number_entered;
    printf("Enter an integer number: ");
    scanf("%d", &number_entered);
    number = number_entered;
    while (number < number_entered + number_entered)
    {
        printf("%d\n", number);
        number = number * 2;
    }
    return 0;
}
Revision: do-while

The loop executes at least once and continues to repeat while the expression `carry_on` is non-zero. As soon as it is zero the program jumps to the next statement after the `while` condition.

Exercise 3.2 do-while

- Modify exercise 3.1 so that the program asks the user to enter an integer between 1 and 100 and repeats the question until the user complies before continuing.
Exercise 3.2: a solution

```c
#include <stdio.h>
int main(void)
{
    int number, number_entered;
    int number_invalid;
    do
    {
        printf("Enter an integer number between 1 and 100: ");
        scanf("%d", &number_entered);
        number_invalid = (number_entered > 100) || (number_entered < 1);
        if (number_invalid)
            printf("Sorry this number isn't valid\n");
        else
            printf("Thank you\n");
    }
    while (number_invalid);
    number = number_entered;
    while (number < number_entered*number_entered)
    {
        printf("%d\n", number);
        number = number * 2;
    }
    return 0;
}
```

Revision: `for` loop

- **Evaluated, just once at start of loop**
- **Conditional expression evaluated at start of each iteration of the loop**
- **Expression evaluated at end of each iteration of the loop**

```c
for (expression1; expression2; expression3)
{
    /* statements that you want repeated */
}
```
Examples of \texttt{for} loops

```c
int i;
int sum1 = 0;
int number = 1;
double x, sum = 0.0;
for (i = 1; i <= 10; i = i + 1)
  sum1 = sum1 + 1;
for (i = 10; i >= 1; i = i - 1)
  number = number*i;
for (i = 2; i <= 20; i = i + 2)
  {
    x = 1/(i*i*i);
    sum = sum + x;
  }
```

Increment operators

- There are some special operators in C that are shorthand for commonly used statements (often used in \texttt{for} loops)
  ```c
  ++
  --
  ```
- They can only be applied to an \textit{integer variable}. For example
  ```c
  int j = 0;
  int j = 5;
  j++;
  j--;
  ```
Assignment operators

- C allows some special operators to be shorthand for more complex assignments. The format of these is
  \[ \text{variable op= expression} \]
- Which means
  \[ \text{variable = variable op (expression)} \]
- Where op can be +, -, *, /, and a few others. For example

```c
#include <stdio.h>

int main(void)
{
    int k = 0;
    int j = 4;
    double x = 4.2, y = 3.5;
    x = 4.2;
    k += 2;
    j *= k + 3;
    x /= y + 2.5;
    printf("k is %d\n",k);
    printf("j is %d\n",j);
    printf("x is %f\n",x);
    return 0;
}
```

Exercise 3.5: Factorial

a solution

```c
#include <stdio.h>

int main(void)
{
    int number_entered;
    int count;
    int factorial = 1;
    int sign = 1;
    printf("Enter an integer number: ");
    scanf("%d", &number_entered);
    if (number_entered < 0)
    {
        sign = 0;
        number_entered = -number_entered;
    }
    for (count = number_entered; count >= 1; count--)
    {
        factorial = factorial * count;
    }
    if (sign)
        printf("%d! is %d\n", number_entered, factorial);
    else
        printf("-%d! is %d\n", number_entered, factorial);
    return 0;
}
```
In the lab this week (lab 4)

• The C Preprocessor
  – This is invoked whenever a line begins with #
  – #define, #include, #if, #ifdef,
• Debugging techniques
• Running programs without the IDE