Computer Programming using C
Lecture 3

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 2)

• Getting input from the user
  – the `scanf` function
  – More on the `getch` function
• Making decisions in programs
  – `if` statements and `switch` statements
• Decision expressions
  – Relational operators `<`, `>`, `==`, `!=`, `<=`, `>=`
  – Logical operators `&&`, `||`, `!`
• Compound expressions
  – curly brackets `{ }`

Revision 1: `scanf`

• Getting input from the user using `scanf`. It lets the user type in text until it encounters a return and tries to match the text with the placeholders specified in the format string.

```c
/* lab2: using scanf */
#include <stdio.h>

int main(void)
{
    int number_entered;

    printf("Enter an integer number: ");
    scanf("%d", &number_entered);

    printf("The number you entered was %d\n", number_entered);
    return 0;
}
```
Exercise 2.1

• Change lab2.c so that the program accepts real valued numbers not just integers

```c
#include <stdio.h>
int main(void)
{
    double number_entered;
    printf("Enter a real number: ");
    scanf("%lf", &number_entered);
    printf("The number you entered was \%lf\n", number_entered);
    return 0;
}
```

Simple conditional statements

• A fundamental aspect of all computer languages is the conditional statement. In C there are three kinds of conditional statements: if, if-else, and switch. These can be in simple form or compound form

• if and if-else statements look like this

```c
if (expression)
    do_this=1;

if (expression)
    do_this=1;
else
    do_that=1;
```
What can be in the round brackets in an if statement?

- An expression
  - This is a meaningful combination of constants, variables, operators or function calls

Examples

1 + 2
x + y + z
printf("Hello")
3.142*radius*radius
do_this = 1
x > y
(3 > 4) && (4 > 3)

Statements

- A statement in C is an expression followed by a semicolon. For example

  3.142;
a+b;
if (x==6) y=3;
velocity = 4.3;
i = i + 1;
printf("hello");
More complex if and if-else statements

- if statements can be nested to produce more complex conditional expressions but be careful about their interpretation

```c
if (a == 5)
    if (b == 7)
        c = 2;
if (x > 5.3)
    if (y < 4.7)
        printf("target in range");
    else
        printf("not in vertical range");
if (radius <= 5)
    area = 3.1415*radius*radius;
else
    if (radius <= 10)
        area = radius*radius;
    else
        printf("radius too large");
```

Compound if, if-else statements

- A number of statements can be executed within conditions by using the curly brackets { }. For example:

```c
if (a == 0)
    { if (b == 0) 
      { x = 0; 
        y = 1; 
      } 
      else 
        { x = 1; 
          y = 0; 
        } 
    } 
else 
    { if (b == 0) 
      { x = 1; 
        y = 0; 
      } 
      else 
        { x = 0; 
          y = 1; 
        } }
```
Relational operators expressions

- In C the relational operators are the binary operators that take two expressions as operands and produce either the int values 0 or 1
  
  `<   ` >   ` <=  ` >=

- If a and b are arbitrary arithmetic expressions then the above operators give the following

<table>
<thead>
<tr>
<th>a minus b is?</th>
<th>a &lt; b</th>
<th>a &gt; b</th>
<th>a &lt;= b</th>
<th>a &gt;= b</th>
</tr>
</thead>
<tbody>
<tr>
<td>positive</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>zero</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>negative</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Equality operators expressions

- In C the equality/inequality operators are the binary operators that take two expressions as operands and produce either the int values 0 or 1
  
  `==   ` !=

- If a and b are arbitrary arithmetic expressions then the above operators give the following

<table>
<thead>
<tr>
<th>a minus b is?</th>
<th>a == b</th>
<th>a != b</th>
</tr>
</thead>
<tbody>
<tr>
<td>zero</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>nonzero</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
Logical operators expressions

- In C there are 3 logical operators when applied to expressions they all produce int values 0 or 1
  
  - !
  - &&
  - ||

- If a and b are arbitrary arithmetic expressions then the above operators give the following

| a    | !a |   | a    | b    | a && b | a || b |
|------|----|---|------|------|--------|--------|
| zero | 1  |   | zero | zero | 0      | 0      |
|      |    |   | zero | nonzero | 0    | 1      |
| nonzero | 0 |   | nonzero | zero | 0    | 1      |
|      |    |   | nonzero | nonzero | 1    | 1      |

What gets printed?

```c
int i = 5, j = 6;
if ((i = j) - 6)
    printf("%d is equal to %d\n", i, j);
else
    printf("%d is not equal to %d\n", i, j);
```
Exercise 2.3: answer

- You were asked to modify the lab2.c program so that after the user has entered an integer the program prints out one of the following:
  (a) whether the number is less than or equal to zero
  (b) if the number is between 1 and nine
  (c) if the number is greater than or equal to 10

```c
/* lab2: ex2.3 answer */
#include <stdio.h>

int main(void)
{
    int number_entered;
    printf("Enter an integer number: ");
    scanf("%d", &number_entered);
    printf("The number you entered was %d\n", number_entered);
    if (number_entered <= 0)
        printf("That number is less than or equal to zero\n");
    else if (number_entered <= 9)
        printf("That number is between one and nine\n");
    else
        printf("That number is greater than or equal to ten\n");
    return 0;
}
```

Exercise 2.5

- Write a program had to read in two integers and then print a message when ONE of the following are true.
  (a) Either number is less than 0
  (b) Both numbers are less than 0
  (c) When the first number is greater than 10 and the other is NOT greater than 10
Exercise 2.5: an answer

```c
#include <stdio.h>

int main(void)
{
    int number1, number2;
    int number1_is_less_than_zero;
    int number2_is_less_than_zero;
    printf("Enter two integers: ");
    scanf("%d%d", &number1,&number2);
    printf("The numbers you entered were %d and %d(n", number1,number2);
    number1_is_less_than_zero = number1 < 0;
    number2_is_less_than_zero = number2 < 0;
    if (number1_is_less_than_zero && number2_is_less_than_zero)
        printf("Both numbers are less than zero");
    else if (number1_is_less_than_zero || number2_is_less_than_zero)
        printf("Only one number is less than zero");
    else if ((number1 > 10) && (number2 >10))
        printf("The first number is greater than 10 but the second is not");
    else
        printf("Both numbers are positive
");
    printf("and either the first is less than or equal to 10, \n");
    printf("or the second is greater than 9, or both
");
    return 0;
}
```

Switch and lab2a

In exercise 2.7 you were asked to modify lab 2a so that `getch()` is used to enter a letter and then the months that begin with that letter should be displayed.

```c
#include <stdio.h>

int main(void)
{
    int number_entered;
    printf("Enter an integer number between 1 and 9: ");
    scanf("%d", &number_entered);
    printf("The number you entered was ");
    switch (number_entered)
    {
    case 1:
        printf("January");
        break;
    case 2:
        printf("February");
        break;
    case 3:
        printf("March");
        break;
    /* other cases */
    case 9:
        printf("September");
        break;
    case 10:
        printf("October");
        break;
    default:
        printf("not between one and nine\n");
    }
    return 0;
}
```
Exercise 2.7 solution

```c
#include <stdio.h>
#include <cctype.h>

int main(void)
{
    char character Entered;
    printf("Enter the first letter of the month: "); character Entered = getch();
    printf("The character you entered was ");
    switch (character Entered)
    {
    case 'J': case 'G':
        printf("J: January, June and July\n");
        break;
    case 'F':
        printf("F: February\n");
        break;
    case 'M': case 'N':
        printf("M: March and May\n");
        break;
    case 'A': case 'G':
        printf("A: April and August\n");
        break;
    case 'S': case 'B':
        printf("S: September\n");
        break;
    case 'O': case 'N':
        printf("O: October\n");
        break;
    case 'N':
        printf("N: November\n");
        break;
    case 'D':
        printf("D: December\n");
        break;
    default:
        printf("You did not enter a letter that begins a month\n");
        break;
    }
    return 0;
}
```

ASCII Codes

<table>
<thead>
<tr>
<th>Backspace</th>
<th>Return/Enter</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>13</td>
</tr>
</tbody>
</table>

There are 128 allocated codes (7 bits)

<table>
<thead>
<tr>
<th>! &quot; # ? % &amp; ’ &lt; &gt; * + , - . /</th>
<th>33 34 35 36 37 38 39 40 41 42 43 44 45 46 47</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 1 2 ... 9</td>
<td>48 49 50 ... 57</td>
</tr>
<tr>
<td>: ; &lt; = &gt; ? @</td>
<td>58 59 60 61 62 63 64</td>
</tr>
<tr>
<td>A B C ... Z</td>
<td>[ \ ] ^ _ ‘</td>
</tr>
<tr>
<td>65 66 67 ... 90</td>
<td>91 92 93 94 95 96</td>
</tr>
<tr>
<td>a b c ... z</td>
<td>97 98 99 ... 122</td>
</tr>
</tbody>
</table>
Extended ascii codes: reading arrow keys

- Some characters on modern keyboards didn’t exist when the ASCII codes were decided. So they require 8 bits. To get at these codes you need two calls of `getch`. They occur when the first call is zero.

```c
int number_entered;
printf("Press a key on the keyboard: ");
number_entered = getch();
if (number_entered == 224)
    number_entered = getch();

char character_entered;
printf("Press a key on the keyboard: ");
character_entered = getch();
if (character_entered == -32)
    character_entered = getch();
```

In the lab this week (lab 3)

- Iteration means repeating groups of instructions. Very important feature of programming languages.
- In C there are three ways to iterate
  - `while()`
  - `do {} while ()`
  - `for (i=0;i<end;i++)`
- Incrementation