Lesson 2

Selection

In this lesson, you will learn about:

- More C operators
- The syntax of the `if()` statement
- The syntax of the `if() – else` statement
- Boolean operators

Important Notes:

- **You should type in all the programs in this handout, and run them more than once with different data**
- **You should read, and understand everything in this handout, the material in it forms the basis of the quizzes**
- **If you don't understand something; ask me to explain.**

More C operators

Below is a table containing some C operators, and how they are “read”.

Operator	Read as
&	“Address of”
=	“Takes the value of”
==	“Equal to”
!=	“Not equal to”
>	“Greater than”
>=	“Greater than, or equal to”
<	“Less than”
<=	“Less than, or equal to”

Assignment operator

It is important that you know these operators, and how to read them.

Notice that the *equality operator* is ==, not =, which is the *assignment operator*.

The `if()` statement

The syntax of the `if()` statement, follows the form:

```
if ( <expression is true> )
{
  <execute these statements>
}
```

Most of time, the *expression* contains one of the comparison operators, and will be used to compare the values held in variables, such as:

```
if (first_number == second_number)   // both the same

if (first_number != second_number)   // both different

if (first_number > second_number)   // first is largest

if (first_number < second_number)   // second is largest
```

If the expressions between the `()` are true, then the statements between the following `{}` get executed.

The `if() - else` statement

The `else` is an optional extra of the `if()` statement. The syntax of the `if() - else` follows the form:

```
if ( <expression true> )
{
  <execute these statements>
}
else
{
  <execute these statements>
}
```

The `else` directs the program to execute the statements between the `{}` following the `else`, when the `if()` expression is not true. A simple example follows in program `age1.c` on the next page.

Program (age1.c)

```
/*
  Program "age1.c"
  Written by: Joe Dorward
  Date: 05/24/00

  This program demonstrates the basic use of the if - else
  statement.  It asks the user to enter a customer's age
*/

#include <stdio.h>

void main(void)
{
int customer_age = 0;

  printf("\n Please enter the customer's age: ");
  scanf("%d",&customer_age);

  if (customer_age < 21)   // Customer younger than 21 years
  {
    printf("\n Not OK - ");
    printf("The customer is too young to buy alcohol. \n");
  }
  else
  {
    printf("\n OK - ");
    printf("The customer may buy alcohol. \n");
  }
}
```

Program (pin.c)

```
/*
  Program "pin.c"
  Written by: Joe Dorward
  Date: 05/16/00

  This simulates an ATM asking for a PIN, and uses the test condition
  in an if() statement to choose the course of action
*/

#include <stdio.h>

void main(void)
{
const int good_pin = 1234;

int entered_pin;

  printf("\n ** PIN REQUIRED TO ACCESS THIS MACHINE ** \n");

  printf("\n Please enter your PIN: ");
  scanf("%d",&entered_pin);

  if (entered_pin == good_pin)
  {
    printf("\n Welcome to the Wells Fargo ATM server.");
    printf("\n You may now transfer other people's money ");
    printf("into your own account. \n\n");
  }
  else
  {
    printf("\n ** ACCESS DENIED **");
    printf("\n I know who you are, and I'm calling the cops now. \n\n");
  }
}
```

Program (pos_neg1.c)

If a number is less than zero it must be a negative number. In some programs, you'd want to check input numbers for that value.

This program demonstrates the use of the `if() - else` statement to check for a negative number.

```
/*
  Program "pos_neg1.c"
  Written by: Joe Dorward
  Date: 03/18/00

  This program demonstrates the basic use of the if() - else statement,
  by testing for a negative number.
*/

#include <stdio.h>

void main(void)
{
int the_number;

  printf("\n Please enter an integer: ");
  scanf("%d",&the_number);

  if (the_number < 0)   /* it's a negative number */
  {
    printf("\n Not OK - The number: %d is negative. \n",the_number);
  }
  else
  {
    printf("\n OK - The number: %d is positive. \n",the_number);
  }
}
```

Program (high_low1.c)

This program asks the user for two numbers, it compares them, and prints them to the screen in order.

```
/*
  Program "high_low1.c"
  Written by: Joe Dorward
  Date: 05/10/00

  This program reads in two integers
  It compares their value, and prints them to the screen
  in order, the lowest first
*/

#include <stdio.h>

void main(void)
{
int first_number,
    second_number;


  // Ask for a number
  printf("\nPlease enter an integer: ");
  scanf("%d",&first_number);

  // Ask for a number
  printf("\nPlease enter an integer: ");
  scanf("%d",&second_number);

  // Test the numbers, and choose the message
  if (first_number < second_number)
  {
    printf("\n The first number: %d is lower than",first_number);
    printf(" the second number %d \n",second_number);
  }
  else
  {
    printf("\n The second number: %d is lower than",second_number);
    printf(" the first number %d \n",first_number);
  }
}
```

This program assumes that the numbers are different.

Program (numbers1.c)

This program asks the user for a whole number, then prints the value of that number to the screen in English.

```
/*
  Program "numbers1.c"
  Written by: Joe Dorward
  Date: 05/15/00

  This program asks the user for a whole-number in the range 0 - 2,
  then prints out the number in English.
*/

#include <stdio.h>

void main(void)
{
int the_number;

  printf("\n Please enter a whole-number (0 - 2): ");
  scanf("%d",&the_number);


  if (the_number == 0)
  {
    printf("\n Zero. \n");
  }
  else if (the_number == 1)
  {
    printf("\n One. \n");
  }
  else if (the_number == 2)
  {
    printf("\n Two. \n");
  }
}
```

Program (`add_test.c`)

This program asks the user to enter two integers, then challenges the user to add them. It tests the user's answer in the `if()` statement, and prints out the appropriate message.

```
/*
  Program "add_test.c"
  written by: Joe Dorward
  Date: 05/10/00

  This program asks the user for two numbers.
  It then asks the user what they add up to.
  It then checks the answer in an if() statement,
  and prints out a right/wrong message.
*/

#include <stdio.h>

void main(void)
{
int first_number,
    second_number,
    the_answer;


  // Ask for a number
  printf("\n Please enter an integer: ");
  scanf("%d",&first_number);

  // Ask for a number
  printf("\n Please enter an integer: ");
  scanf("%d",&second_number);

  // Ask the question
  printf("\n What does %d + %d = ",first_number,second_number);
  scanf("%d",&the_answer);

  // Test the answer, and choose a message
  if (first_number + second_number == the_answer)
  {
    printf("\n Hey, you got it right! \n");
  }
  else
  {
    printf("\n Boy did you get it wrong! \n");
  }
}
```

More on ASCII characters, and their values

Looking at the ASCII table, you can see that the characters (and their values) fall into groups:

digits	'0' = 48	'9' = 57
uppercase letters	'A' = 65	'Z' = 90
lowercase letters	'a' = 97	'z' = 122

This tells you that if a character has a value in the range:

48 - 57	that it is a digit
65 - 90	that it is an uppercase letter
97 - 122	that it is a lowercase letter

Those values are something we can test for inside an `if()` statement.

You will also notice that the difference in values between the uppercase letters, and their lowercase versions is 32.

If you add 32 to the value of `'A'` (65) you get 97, the value of `'a'`.

You'll be using that fact to write a program later.