Department of Computer Science

2007/2008
08101 Programming I
Practical 1

Laboratory Work in the Department

Welcome to our laboratory sessions. You will have a lab session each week during the taught part of the programming course. *It is best if you prepare for the laboratory in advance.* You can view the lab material by visiting the web site for this course. Details of the address of the web site are given at the end of this document.

Remember that these practical sessions will **not** teach you how to program. They will help you *learn* how to do it. You have to put effort into the process yourself, just like you had to work hard to learn to ride a bicycle! (note that you can still learn to program even if you cannot ride a bike)

In this session you are going to find out how to start using the computers, how to create a session and also how to create and run programs written using the C# programming language.

The Laboratory Documents

Each week you will be given a document which describes what you have to do. Some of the work will be assessed, either in the laboratory itself, or as a submitted piece of work. In these lab sheets the work you have to do is indicated as follows:

This indicates an activity you should perform in the laboratory, at this point in the text. You may be given precise instructions, or you may have to work something out for yourself.

This indicates an investigation that you will have to perform. You should write up what you discover as this material may form the basis of discussion in your tutorial group meetings.

The University Network

All of the workstations you are going to use are connected to the university campus network, which operates using Microsoft networking. The workstations themselves run the Microsoft Windows XP operating system.

There are a large number of workstations around the campus. Some are in the Computer Centre, others are in the library and we have systems in laboratories which are specifically set aside for Computer Science and Engineering students. Most of your practical sessions will take place in the Fenner Computer Suite.

It does not matter which particular workstation you sit down at. A number of user settings, called your *profile,* are held centrally and used to configure the workstation that you log in to. However, you may notice that some systems provide a different range of programs. This is because we have customised "our" workstations with software which is not generally available across the campus.

Registration and Usernames

Each user of the campus network is uniquely identified by a user number. A valid number allows you to make use of computer systems managed by the university. Your number also gives you

access to your own area on a central file store and a print quota for the laser printers. You will also need your number to send and receive email using your university email address. All of the items that you create on the campus network are marked as being "owned" by you and you are the only person allowed to use them.

Usernames and Passwords

You have been allocated a user number and password as part of your registration. Your welcome pack contains details of your unique user identity, which is a six digit number. This number is unique to you and will stay with you throughout your university career. Your password is generated randomly. You will have to give your password each time you "log in" to start a session on the computer. A few things about passwords:

1. If you forget your password or user number you will be unable to use the computer systems. You will have to go to Computer Reception and ask for help. You will be required to present proof of identity.
2. The password you are initially given is printed on your registration form. In case anyone has seen this form you should change this password the very first time you use the computer systems. Try to choose a memorable password, but be careful that you do not choose one which is easy to guess. The system may reject passwords that it regards as insecure. I suggest that you run two words together, and perhaps include numbers as well as text. In the past I have used things like "s0mechance" (please do not use this). *Do not to pick a password that you have to write down!*
3. You will be held liable for any actions performed whilst logged in with your user number. It is **very** important that you keep your password private. Do *not* divulge it to anyone else.
4. If you suspect that someone is using your user number and password to gain access to the computer systems you should inform the Computer Centre as a matter of urgency.

You change your password using the **`setpass`** command. You will use this command later in this laboratory.

Computer Regulations

The regulations governing computer use in the university are printed on the back of your registration form. If you fail to abide by these you may find your username disabled, which might make it impossible to complete mandatory coursework, and lead to you failing modules.

Sessions at a Workstation

At the start of a session you introduce yourself to a vacant workstation by "logging on". When you log on you identify yourself with your user number and password. This combination is checked to ensure that you are who you say you are. If the user number and password are valid the Windows XP operating system then sets up a number of system preferences for you, connects to your file storage are and allows you to use the workstation.

At the end of a session you say goodbye by "logging off". At this point the link with your file storage is broken, and the workstation waits for the next person to log on.

Logging off properly is at least as important as logging on. If you do not log off you leave the workstation connected to your name, and anyone else can do what they like *in your name*; including deleting all your programs, reading and sending email etc.

Logging on to a workstation

To log in to the system find a vacant machine. If the screen is completely blank you may have to move the mouse to stop the screensaver. Hold down the Ctrl and Alt keys and then press the Delete

key (sometimes marked Del and located next to the 0 key in the numeric keypad at the right of the keyboard). This will cause the following screen (or one a bit like it) to appear:

Figure 1 Login Dialogue

This is the login screen. Use the mouse to move the pointer into the text box next to the Username label and press the left hand button on the mouse. This is called *clicking* in an item on the screen. Use the keyboard to enter the username which you have been given. This should be in its full form as given on your registration form, with the leading dot. If there is any other text in the text box, delete it. Then click inside the password text box and enter your password, *exactly as given.* The text of your password will be replaced with * characters as you type them in.

Once you have entered your username and password use the mouse to click on the OK button. This starts the login process. A number of screens will appear. Half way through the login process you will be invited to press a key to continue. When the login process is complete a Windows NT desktop will be displayed.

If the login process fails make sure that you have *not* used CAPITAL LETTERS where you should have used lower case. Windows XP treats A and a as different characters in both the username and the password. Also make sure that the Workstation only tick box is empty. If anything goes wrong, make a note of the error and then report it to a laboratory demonstrator.

Logging out from a Workstation

When you have finished working on the workstation you *must* log out. To log out you use the mouse to press the Start button. This will cause the Start Menu to appear. Press the Shut Down button at the bottom of this menu. This will cause the Shut Down screen to appear. Select the option to log off.

You should not turn off the power from the computer. The workstations are left running at all times.

Once the logout has completed the system should display a prompt for the next user.

Before you go any further; perform the following:

1. Log on to a workstation using your username and password.

The Command Prompt

In the first part of the programming course you are going to use the Command Prompt to tell Windows XP what to do. The command prompt is actually a program which you run. The command prompt program understands a number of "built in" commands. It will also load and start other programs for you, such as the Notepad editor and the C# compiler.

All versions of the Windows operating system provide a command prompt. The commands that are understood by it are based on those used by the original MS-DOS operating system, which is why it is sometimes referred to as the MS-DOS prompt.

Getting a Visual Studio Command Prompt

For the first few weeks we are going to create our programs at the *command prompt*. Later we will start to use the Visual Studio environment. Once you have logged into the system you can find the Visual Studio Command tools by traversing the program menu as shown below:

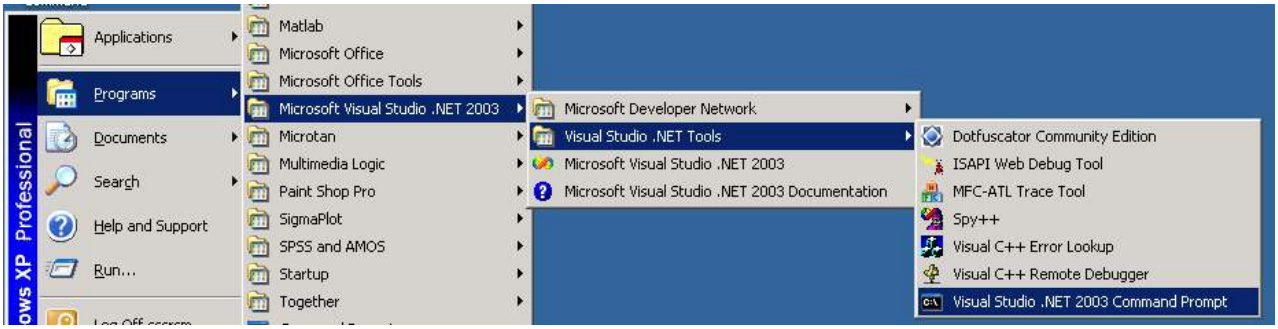

Figure 2 Visual Studio 2003 Command Prompt path

Note: You will be using Visual Studio 2005, the path to the program is the same, but the version of the program is different.

This should produce a special command prompt where you can use the Visual Studio commands.

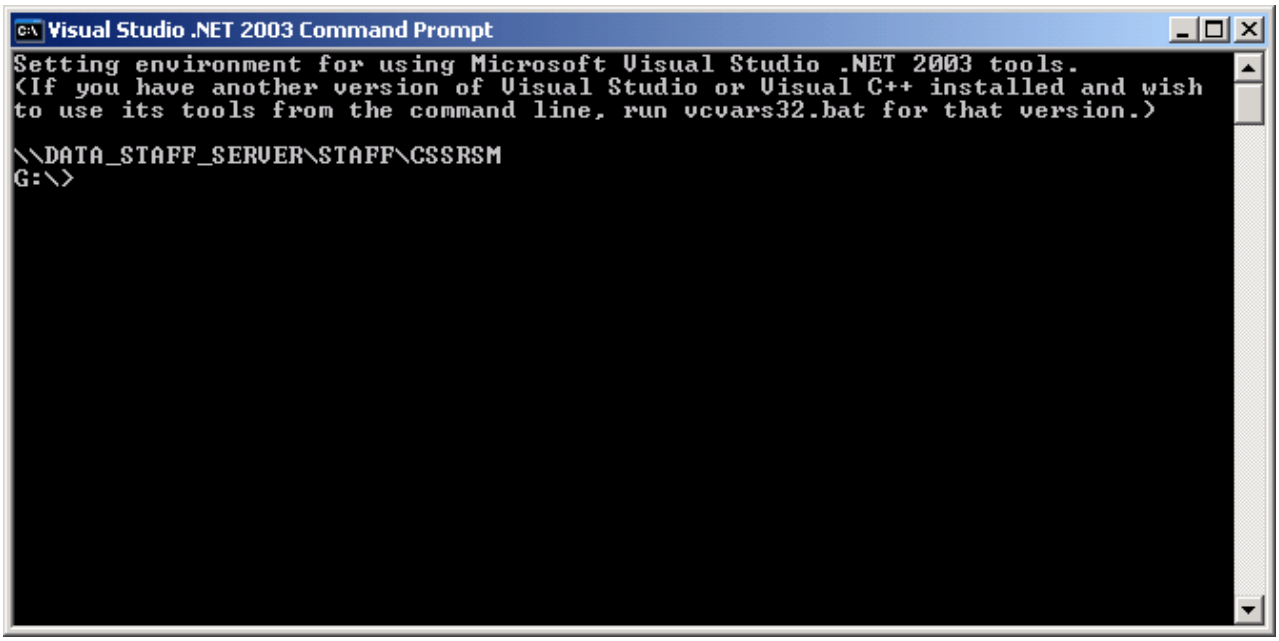

Figure 3 Visual Studio Command Prompt

You type commands into the window and press the Return key (sometimes marked as Enter or with a left pointing arrow on it) to get the commands obeyed. I call this *issuing* a command.

You can close the command window by typing the command **`exit`** and pressing the enter key. The window will then close. You can have several command prompt windows active at one time by starting multiple versions. Two may be useful, but a very large number of windows can be confusing. You can select a window by clicking somewhere inside it.

You do not need to close all your command prompt windows when you log out, as Windows XP will shut them automatically.

Before you go any further; perform the following:

2. Start a Visual Studio command prompt window.

Managing your File space

A computer using the Microsoft Windows operating system organises its file space in terms of *disk drives*. Each drive attached to a particular system is identified by a given letter of the alphabet. By convention the drive identified by the letter A is the *floppy disk* drive. The internal *hard* disk is usually identified by the letter C.

The Netware system in use on the university campus creates other, *network*, drives which are also identified by letters. When you store information on a network drive you are actually storing it on a server somewhere in the Computer Centre. The area of storage space which you are allocated is connected to network drive G. This drive is selected automatically when you start using the command prompt. You can select the drive you wish to use by issuing a command which is just the drive letter, followed by a colon character:

```
G:
```

- would select drive **G**. Note that, as with all storage media, the amount of storage space you are allowed to use is limited to a particular amount. If you have a good, academic, reason why you need more file space you can request this from the Computer Centre. Note also that you will not be allowed access to files on some drives as these are part of the Windows system itself.

Directories and Folders

A typical computer has many thousands of files on it. If these were all held in one place on the disk drive it would be impossible to find anything. On a particular drive you can create *directories* or *folders* which can organise a set of files together. The terms folder and directory mean the same thing. You will use your university account for a variety of purposes, it is sensible to create directories to keep your work together.

We are going to perform all the creation and management of our directories from the command prompt. First we need to create a directory to hold our programs. We do this with the **md** (short for *Make Directory* command). You follow the **md** command with the name of the directory you wish to make.

```
md 08101lab01
```

- would create a directory which has the name **08101lab01**. Note that I have not created any files, merely a place where I can store them.

We can then move *into* a directory using the **cd** (short for *Change Directory* command). Think of a folder (or directory) as a room. When you are inside the room you can directly refer to any resource in that room. If you leave the room and enter another, you can directly refer to resources in the room you have just entered. When you start the command prompt you are at the top level of your file space, where the drive starts from. Think of this as the hall in your house. You then issue **cd** commands to move from one directory to another:

```
cd kitchen
```

- would *move* me into a directory called **kitchen** (which must exist otherwise Windows will complain). Certain characters have special meaning where the **cd** command is concerned. The character . (the full stop) means *the current directory.* The command:

```
cd .
```

- would take me precisely nowhere. However the sequence **..** (two full stops) means the directory *above* my present directory, i.e. the directory that I came from to get here. This means that if I am in the kitchen, and I issue the command:

```
cd ..
```

- this would take me "up" a level, to the directory that contains my current directory. If I was in the **kitchen** directory this would take be back to the hall.

You can ask the command prompt to tell you the names of all the files in your current directory. You use the command **dir** to do this:

```
dir
```

Just as what you can see depends on the room you are in, the output from the **dir** command will tell you the contents of your current directory.

Before you go any further; perform the following:

3. At the command prompt issue the commands:

   ```
   G:
   md 081011ab01
   ```

   This will move you to drive **G:** and create a directory called **081011ab01**.
4. Now issue the following command:

   ```
   dir
   ```

   This will show you all the files in your drive **G**, including the **081011ab01** directory. Do not worry about, or mess with, the other files, they are there to keep the system working for you.
5. Now issue the command:

   ```
   cd 081011ab01
   ```

   This will move you into the **081011ab01** directory.
6. Now issue the command:

   ```
   dir
   ```

   This will show you the contents of the **081011ab01**directory, which should be empty.
7. Now issue the command:

   ```
   cd ..
   ```

   This will take you out of the **081011ab01**directory and back to the top level.
8. Now issue the command:

   ```
   cd 081011ab01
   ```

   This will take you back into your **081011ab01**directory again. Note how the prompt changes as you issue these commands.

The Notepad editor

We will use the Windows notepad editor to create and manipulate our programs. Notepad is a very simple editor which lets you work on files of text. You can start the notepad program by issuing

the command **notepad** at the command prompt. This will start up notepad with an empty file. If you want to edit an existing file, or create a file with a particular name, you can follow the command with a filename:

```
notepad letter.txt
```

This would start notepad and ask it to open a file called **letter.txt**. If the file does not exist the notepad program will ask if you want to create one.

Before you go any further; perform the following:

9. At the command prompt type the following:

   ```
   notepad Hello.cs
   ```

   The notepad program will ask if you want to make a file called **Hello.cs**. Accept by clicking on Yes.

10. Enter the C# program below into the notepad text window:

    ```
    using System;
    class Hello
    {
       static void Main()  {
            Console.WriteLine("Hello World");
       }
    }
    ```

11. Ensure that the program is typed in **exactly** as above, with correct use of curly **{**, ordinary **(** and square **[** brackets where appropriate. Use the cursor keys or the mouse to move around and make any corrections by overtyping and using the Del key to remove characters.

12. Click on the **File** command on the menu bar at the top of the Notepad window. Select Save from the menu which appears. This will save a copy of the file. Leave Notepad running so that you can make changes to the file and save them again later.

Compiling a C# program

If you have followed the sequence above you now should have a C# program in your **08101lab01** directory. We must now use the C# compiler to convert the text that you have written into a set of low level instructions that the .NET runtime can understand and obey. The C# compiler is a program which acts on C# source files. You run the compiler from the command prompt using the command **csc**. The command to start the compiler **must** be followed by the name of the text file which holds the C# source you want to have compiled:

```
csc Hello.cs
```

This would ask the C# compiler to take a look at the C# program in the file **Hello.cs**. If the C# compiler likes the file that it sees, it creates an *executable* file which can be run as a program. This file has the letters **.exe** on the end of the name. These letters on the end of the name are known as the filename *extensions.*

It is perfectly OK to have two files with the same name and different extensions. The extension part of the name identifies the type of data in the file. To run a program file you simply give the name of the file containing the program:

```
Hello
```

The program in the file **Hello.exe** is loaded and run by the Windows XP.

Before you go any further; perform the following:

13. At the command prompt type the following:

    csc Hello.cs

    The compiler will run and, hopefully, produce no errors.
14. If you get any errors you will have to use notepad to edit the file **Hello.cs** and make sure that it matches the sample text *exactly*. Remember to save the edited program before trying to compile it again.
15. If the compiler has worked correctly issue the **dir** command to find out what the compiler has done. You will notice a new file has appeared.
16. Issue the command:

    Hello

    This should result in your program running.

If your program does not compile make sure that the text in the file exactly matches the text in the example above. Pay special attention to the type of the brackets and the use of CAPITALS and lower case characters.

Copying files

The command **copy** can be used to copy a file from one location to another. This can be used to save you entering a lot of text if you are making a new program based on an earlier one.

```
copy Fred.cs Jim.cs
```

- would make a new file called **Jim.cs** which contains a copy of **Fred.cs**.

Reading input text from the user

The first program that we wrote just printed a message. It did not receive any input from the user. The next program we write is going to read some user input.

When you want a C# program to read a message from the keyboard you have to set up a place where the message can be stored. This message will be a string of text, and so we use a **string** data type to store it in, as in the program below:

```
using System;

class Greeter
{
      static void Main()
      {
            Console.Write("Enter your name : ");
            string name = Console.ReadLine();
            Console.WriteLine("Hello " + name );
      }
}
```

The program reads a name as a string of text and then prints out a greeting. At the moment you will have to just enter the program text as given above.

Before you go any further; perform the following:

17. Now issue the copy command make a new file called **`Greeter.cs`** which is a copy of **`Hello.cs`**.

```
copy Hello.cs Greeter.cs
```

18. Use notepad to edit **`Greeter.cs`** so that it contains the program above, save it, compile it and run it.
19. Change the program so that it adds a complement after it has said hello:

```
Hello Rob
Gosh, you are handsome!
```

20. Experiment with **`Write`** and **`WriteLine`**. See if you can describe the difference between the two.

Reading input numbers from the user

We are now going to write a program which can read numbers from the user and perform calculations. C# regards strings of text, for example **`"Hello world"`**, as quite different from numeric values.

If you want your program to read a number from the user you have to read a string of text and then convert that string into a value. C# has something called **`"Parse"`** which takes a string and converts it into an integer value. It is something that the **`int`** class can do for us. In the next program we are going to use this to enable us to write a program which reads in two numbers and then prints out their sum.

```
using System;

class Sums
{
	static void Main()
	{
		Console.Write("First Number : ");
		string number1Text = Console.ReadLine();
		int number1 = int.Parse(number1Text);
		Console.Write("Second Number : ");
		string number2Text = Console.ReadLine();
		int number2 = int.Parse(number1Text);
		int result = number1 + number2;
		Console.WriteLine("Sum is : " + result );
	}
}
```

Again, we will cover what each line of the program does in more detail later in the course. Note that I have added comment lines so that you can see what each part of the program does.

Before you go any further; perform the following:

21. Use Notepad to make a new file called **`Sums.cs`**. Type in the program above and run it.
22. Enter the program above, compile it and run it. Does it work?

Debugging and Testing

If the program that you have typed in seems to work OK, then you must have not copied it exactly. The text above has one bug in it. A *bug* is a mistake in a program which causes it to do the wrong thing. In the case of the above program it sometimes gets the answers wrong.

Before you go any further; perform the following:

23. Make sure that your program is *exactly* the same as the one above.
24. Compile and run it. Devise some test data. Some of the test data should cause the program to fail. Other values should cause the program to appear to work.
25. Fix the bug in the program and test it with the data you have devised.
26. Send an Email to Rob Miles describing the fault and how you fixed it. First email wins a Mars Bar.

Whenever we write a program from now on you will be required to create test data for the program. In fact, you may be asked to create the tests *before* you create the program itself!

RSM 27/09/07