FACTFILE:
GCE DIGITAL TECHNOLOGY

AS1: APPROACHES TO SYSTEMS DEVELOPMENT: PROGRAMMING

Rewarding Learning

The Programming Environment

Learning Outcomes

Students should be able to:
- describe the purpose of a computer program
- describe the main features of an integrated development environment (IDE)
- explain the process of translation

Content in The Programming Environment Fact File
- The purpose of a computer program
- The main features of an integrated development environment (IDE)
- The process of translating a program

The purpose of a computer program
A computer program is a specific set of ordered instructions to be performed by a computer. The program represents a solution to a problem. A computer will usually get one instruction, execute it and then get the next instruction.

Computer programs can be written in high level programming languages such as C#, Java, Visual Basic and Python. They can also be written in low level languages such as machine code and assembly language. A series of language statements are written to solve a problem, this is called the *source code or source program*.

The main features of an IDE
Program code can be created using a text or code editor. The code is then translated into machine code. Source code cannot be executed unless it is fully compiled (*we will return to the translation process later in this document*).

If a text editor is used, the program will have to be compiled using program software. More advanced software known as an Integrated Development Environment (IDE) provides programmers with all of the facilities required to complete the development of an application from coding to testing. This means that an application can be developed fully within one application.

Typically, an IDE consists of the following elements:
- Source code editor
- Compiler
- Debugger
- Graphical User Interface(GUI) builder with an associated toolbox of controls
- Solution Explorer

Source code editor

The basic window for entering source code. Many of these code editors provide:

Clipboard
Where the IDE can remember the last number of items copied. Programmers can rotate through the list of copied items and choose one to paste into the current file.

Code Outlining
Programmers can collapse/expand selected regions of code under their first lines. This means that long programs can be viewed in small logical sections.

Code suggestion IntelliSense
As you enter a function or statement in the Code Editor, its complete syntax and arguments are shown in a ToolTip. When items are needed to complete a statement, IntelliSense provides popup insertion lists of available functions, statements, constants, or values to choose from. This is known as IntelliSense in Microsoft Visual Studio.

Line Numbering
Line numbers help programmers to distinguish between lines in lengthy coding sections.

Syntax error assistance
As code is entered, the Code Editor will place 'wavy lines' beneath code that is incorrect or could cause a problem.

Many IDEs, such as Microsoft Visual Studio provide colour coded error listings. For example, the Error List displays coding problems marked with 'wavy lines'. Programmers can click on any Error List entry to jump to the code where the problem occurs.

- Red lines mark syntax errors, these lines disappear as soon as the marked code is corrected in the Code Editor.
- Blue lines mark semantic errors detected by the compiler, such as a mistyped class name not found in the current context. These disappear after the marked code is corrected and then recompiled.
- Green lines mark warnings. Programmers can review these messages to see if the code needs to be modified.

Compiler
The process of compiling ultimately produces an executable (.exe) file. (In some languages there may be intermediary translation processes). The code written by the programmer is compiled with any additional libraries to create a machine code version of the program. This can then be executed by the processor. Compiling is the core translation process. A program cannot be successfully compiled unless it is error free.

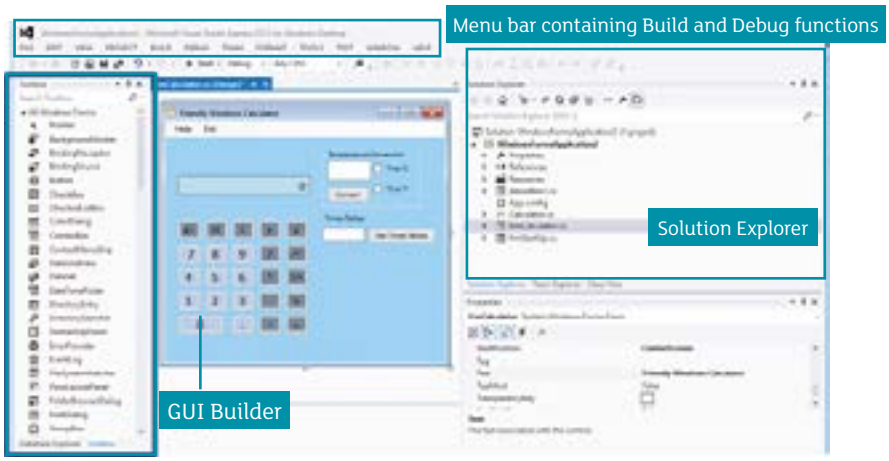
Graphical User Interface (GUI) builder with an associated toolbox of controls
The GUI builder allows a programmer to create windows applications by positioning controls on screen. Controls such as textboxes, combo-boxes and radio buttons can be added to forms by dragging them from a toolbox. There are usually two views: a graphical design view and a code view. Design view allows programmers to specify the location of controls and other items on the user interface.

Solution Explorer
This is a graphical representation of the entire solution. The window displays solutions, their projects, and the items in those projects. Usually, files can be opened for editing, new files can be added and item properties can be viewed.

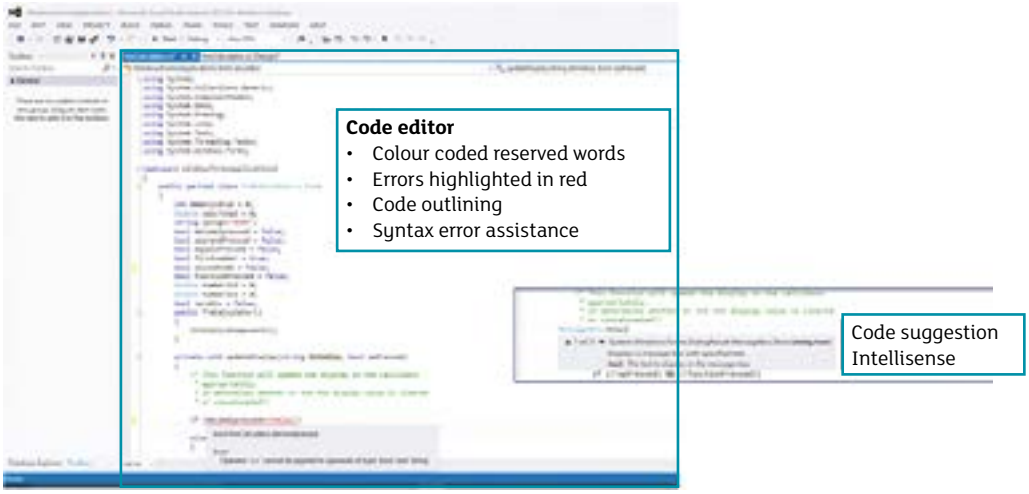
Debugging and break points
Programs can contain logic problems which become apparent at run-time. This is where the code, although syntactically correct, is not providing the correct results. The debugger can be used to help detect and correct such problems. Programmers can set breakpoints where the program will stop during execution and allow the programmer to examine the value of different variables.

An example of a program contained within an IDE

Design View



Code View



The process of translation

Code written in a high level language can be read and understood by humans and requires translation so that the computer can execute the commands in the program.

Translators include compilers, interpreters or assemblers.

An assembler translates assembly language into machine code. Assembly language is a low-level language.

An interpreter translates source code one line at a time. The processor executes a line of code before proceeding to translate the next line of code. No version of the machine code is stored. This is generated each time the program is run.

A compiler will attempt to convert the whole program into machine code before executing it. During the first stage of translating a program, the syntax of each statement is checked. If the statements are not constructed correctly the compiler will generate a list of errors. The source code cannot be fully compiled until all syntax errors are removed from the code.

The source code is "compiled" using a language compiler and the result is called an *object program* (not to be confused with object-oriented programming). The object program contains the string of 0s and 1s called *machine code or native code*. This fully compiled version is also known as an executable version.

A compiler may perform:

- Lexical analysis – converting the incoming source code into fixed length binary code items called tokens;
- Pre-processing – including library code for classes and methods used within the source code;
- Parsing or syntax analysis – checking the statements within the code to ensure they conform to the rules of grammar for the language;
- Semantic analysis – this includes checking to see that variables are declared before being used and type checking;
- Code Generation – creating the machine code version of the source code;
- Code optimisation – making the executable program as efficient as possible.

Many languages have been implemented using both compilers and interpreters. Some language implementations create intermediary versions of code and the machine code is then generated at run-time.

For example c#:

- Compiling translates the source program into common intermediate language (CIL) and generates the required metadata. This is stored as an assembly.
- At execution time, a just-in-time (JIT) compiler translates the CIL into native code. During this compilation, code must pass a verification process that examines the CIL and metadata to find out whether the code can be determined to be type safe.
- The common language runtime (CLR) enables execution to take place. The code to be executed is converted to processor-specific native code.

Python and Java programs are converted into a form of bytecode before they can be executed. Bytecode is computer code that is processed by a program, usually referred to as a virtual machine rather than by the computer's processor. The virtual machine converts each generalized machine instruction into a specific machine instruction that this computer's processor will understand.

Questions

1 A software company is developing an application which will collect details about customers visiting a Shopping Centre.

a) Describe the features of an IDE which make it suitable for developing the application. [6]

b) Describe the features of an IDE which will assist programmers as they undertake the development of the application. [6]

Extension Task:
Investigate an IDE you have used and show (using screenshots or screen recorded animation) how it provides for each of the following:

- Source code editor
- Compiler
- Debugger
- Graphical User Interface(GUI) builder with an associated toolbox of controls
- Solution Explorer

2 a) Describe the purpose of a computer program. [2]

b) Computer programs must undergo a translation process.

(i) Explain why this is necessary. [3]

(ii) Explain the difference between interpreters and compilers. [3]

(iii) Describe the tasks carried out by a compiler during the translation process. [5]

Extension task: With specific reference to a language that you are studying, describe how source code is compiled to object code.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

3 **Practical Task**

a) Complete the following two tasks:

1. Using notepad, create a computer program which will print "hello world" onto the computer screen. Use your chosen language to do this.
2. Using an IDE, create a computer program which will print "hello world" onto the computer screen. Use your chosen language to do this.

b) What differences did you find when using the IDE? What are the advantages?