IDEs (Integrated Development Environments)

A software package containing several features useful for writing code:

Editor
- Allows code to be to written and edited
- Fairly simple with programmer specific features
- Automatic line numbering
- Colour coding
- Auto-correct
- Auto-suggestion
- Auto-indent

Error diagnostics and debuggers
- Help to locate and fix errors.
- Breakpoints allow a program to be paused at certain points so the programmer can then examine different parts of the code or variables.
- Variable tracing shows the changing values of variables as the program runs
- Syntax highlighting shows where syntax errors occur.

Runtime environment
- Software which allows code to run on different platforms to that which they were written on.
- Allows code to be written for specialist hardware without having to have that hardware to hand.
- Creates a virtual machine to run the code in

Translators
- Translates code into a format which the computer can execute.
- Allows code to be run and tested from within the IDE.

2.5 – Programming Languages and Integrated Development Environments

Low Level Languages		High Level Languages	
1st Generation	**2nd Generation**	**3rd Generation**	**4th Generation**
• Also known as Machine Code • Can be executed directly by the CPU • The generation that 'computers understand' • Difficult for humans to understand, write or debug	• Also known as Assembly Code • Uses mnemonics (abbreviations) • Easier for humans to understand and program but still difficult • 1-1 relationship with Machine Code (one Assembly Language instruction translates to one Machine Code Instruction • Must be translated into Machine Code for execution • Commonly used to program device drivers	• Easier still for humans to understand, program and debug • Uses English-Like Keywords • 1-many relationship with machine code (one instruction translates into many machine code instructions) • Examples include Java, Basic and Pascal • Translated using a compiler or interpreter	• Also known as Declarative Languages • Has strict facts and rules for use • Details the computation which should be performed but not how to perform it • Examples include SQL, Expert Systems and Artificial Intelligence

`01010101010100101010100`
`10101010101001111100100`
`0100001010100101`
	`LOAD r1, c`
	`LOAD r2, d`
	`ADD r1, r2`
	`DIV r1, #2`
		`Dim Num1, Num2, Tot as Integer`
		`Num1 = Console.Readline()`
		`Num2 = Console.Readline()`
		`Tot = Num1 + Num2`
			`SELECT * FROM Customers`
			`WHERE Country='Mexico';`

Translators

- Translates a language into a form that the computer can directly execute
- Compilers and interpreters are used for 3rd generation languages

Compilers

- Translates the whole code in one go into Machine Code.
- Optimise the code
- Used at the end of development when code is finished
- Create error reports and object code

Interpreters

- Translate and execute source code
- Work line by line.
- Syntax is checked
- If code is correct it is executed
- If code is incorrect interpreting is stopped.
- Aid debugging

	Compilers	Interpreters
Advantages	• Compiled programs run quickly and without needing additional software. • Compiled programs can be supplied as an executable file which cannot be easily modified. • Optimise code so it runs quickly and uses less memory.	• Instructions are executed as soon as they are translated. • Instructions are not stored for later so less memory is needed. • Errors can be quickly spotted.
Disadvantages	• Because the source code is translated as a whole, more memory is needed. • Requires a temporary working space for the compiler to perform the translation. • Do not spot errors. • Code must be re-compiled every time it changes. • Code compiled on one platform will not run on another.	• The CPU must wait for each instruction to be translated so execution is slower. • Code is translated each time it is run. • Do not produce an executable file that can be distributed • Do not optimise code.