Accredited

A LEVEL

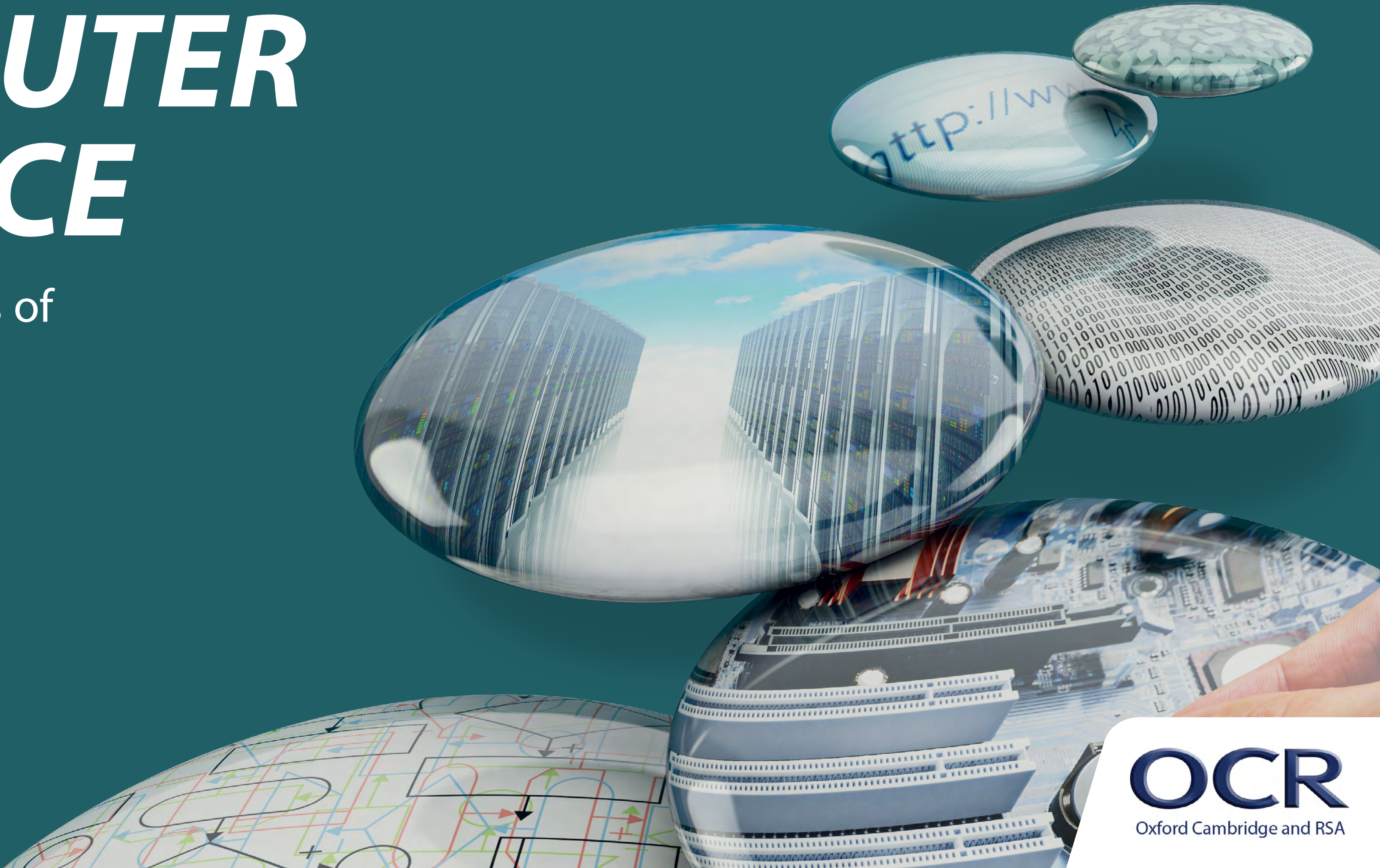
Delivery Guide

H446

COMPUTER SCIENCE

Theme: 1.2.4 Types of Programming

June 2015



OCR
Oxford Cambridge and RSA

CONTENTS

Introduction

Delivery guides are designed to represent a body of knowledge about teaching a particular topic and contain:

- Content: a clear outline of the content covered by the delivery guide;
- Thinking Conceptually: expert guidance on the key concepts involved, common difficulties students may have, approaches to teaching that can help students understand these concepts and how this topic links conceptually to other areas of the subject;
- Thinking Contextually: a range of suggested teaching activities using a variety of themes so that different activities can be selected that best suit particular classes, learning styles or teaching approaches.

If you have any feedback on this Delivery Guide or suggestions for other resources you would like OCR to develop, please email resources.feedback@ocr.org.uk.

KEY

🗎	Click to view associated resources within this document.
▶ Click here	Click to view external resources
AS Level only	AS Level content only

Curriculum Content

a) Need for and characteristics of a variety of programming paradigms.

b) Procedural languages.

c) Assembly language (including following and writing simple programs with the Little Man Computer instruction set). See appendix 5d.

d) Modes of addressing memory (immediate, direct, indirect and indexed).

e) Object-oriented languages (see appendix 5d for pseudocode style) with an understanding of classes, objects, methods, attributes, inheritance, encapsulation and polymorphism.

Thinking Conceptually

Need for and characteristics of a variety of programming paradigms	Resources
There are many different types of programming language. Until students have tried a few different ones they will not necessarily appreciate the differences between them, but for the course it is important to know some of the basic differences they may encounter. Students can use the resources given as activities at the bottom of this document to support their learning. There is a great set of posters available that can be put up around the classroom and covers many different types of programming language, found at: http://community.computingatschool.org.uk/resources/1872 http://www.teach-ict.com/as_as_computing/ocr/H447/F453/3_3_6/types_language/miniweb/index.htm	▶ Click here ▶ Click here
Procedural languages	**Resources**
Procedural programming uses a list of instructions to tell the computer what to do step-by-step. A procedural language uses blocks of code called procedures or routines. It is also referred to as imperative programming. It is intuitive because it is similar to how you would expect a program to work, and most early programming languages followed this approach since OOP had not been introduced yet. A very good video that introduces the ideas of procedural languages vs object-oriented can be found here: http://education-portal.com/academy/lesson/object-oriented-programming-vs-procedural-programming.html (You are limited to the first five minutes but this is enough to cover the basic ground.) http://www.teach-ict.com/as_as_computing/ocr/H447/F453/3_3_6/types_language/miniweb/pg3.htm	▶ Click here ▶ Click here

Assembly language (including following and writing simple programs with the Little Man Computer instruction set) See appendix 5d	Resources
Assembly language is a low level programming language. This means that it provides very close control of the CPU. However, assembly language, being a low level language, is hard to understand, and programmers need to have a good understanding of the way the CPU works, such as the fetch decode execute cycle. The Little Man Computer is a free piece of software that models a Von Neumann architecture computer designed for educational purposes. Students may already be familiar with this from GCSE. It can be programmed in machine or assembly code. This does require software to be installed from http://www.yorku.ca/sychen/research/LMC/ Some of the commands are as follows:	Click here

INP	Input to the accumulator. The user can type a number
STA VariableName	Stores the current value of the accumulator into 'VariableName'
LDA VariableName	Loads 'VariableName' into the accumulator
ADD VariableName	Adds 'VariableName' to the current value of the accumulator

http://en.wikipedia.org/wiki/Assembly_language

Click here

Thinking Conceptually

Modes of addressing memory (immediate, direct, indirect and indexed)

An addressing mode refers to how you are addressing memory in a given location.

It would be useful to teach this part of the specification using the websites listed below.

Immediate addressing is called that because the value to be stored in memory immediately follows the operation code in memory. For example, the instruction: MOV A,#30h

This instruction uses immediate addressing because the accumulator will be loaded with the value that immediately follows; in this case 30 (hexadecimal).

Immediate addressing is very fast since the value to be loaded is included in the instruction. However, since the value to be loaded is fixed at compile-time it is not very flexible.

Direct addressing is called this because the value to be stored in memory is obtained by directly retrieving it from another memory location. For example: MOV A,30h

This instruction will read the data out of RAM address 30 (hexadecimal) and store it in the accumulator. Direct addressing is slightly slower than immediate addressing; although the value to be loaded isn't included in the instruction, it is quickly accessible since it is stored in RAM. It is much more flexible than immediate addressing since the value to be loaded is whatever is found at the given address, which may vary.

Indirect addressing uses a second register (known as the base register) which holds the actual memory address that the program is interested in.

A block of memory called a 'vector table' is used by the loader to store the address of every subroutine in the library. The CPU gets the data by referring to a location in the vector table and then retrieving the memory location stored there. This address is then used to fetch the data.

The instruction MOVA,@4000 looks to this location for an address. Location 7000 is returned. Looking up in this location returns the value 300.

Indexed addressing uses a base address to a block of data, then an index is used from this address to access various parts of the block.

http://en.wikipedia.org/wiki/Addressing_mode

http://www.teach-ict.com/as_as_computing/ocr/H447/F453/3_3_8/lowlevel/miniweb/index.htm

Resources

Click here

Click here

Object-oriented languages (see appendix 5d for pseudocode style) with an understanding of classes, objects, methods, attributes, inheritance, encapsulation and polymorphism	**Resources**
Object Oriented Programming (OOP) is an approach to creating programs that makes use of objects Objects are components of an OOP program that performs certain actions and knows how to interact with other parts of a program. It is important that students understand clearly the terminology associated with OOP. Classes are the blueprints of an object that can be reused. If we take an example of a person, we could say that a person has a name, can do things like speaking and walking. This would be our blueprint. We can create multiple objects of this class by instantiating them, a bit like jelly coming out of a jelly mould. When an object is created, it has a method in the class called the constructor that sets up all the properties a newly created object should have. Methods and attributes can be part of a class. Taking the person example, Name and Date of Birth would be attributes, whilst we would create methods (actions) to make the person speak or move. Inheritance uses the ideas of parent and child (super- and sub-classes) to abstract certain behaviours and attributes for similar classes. For instance, if our person class was in a program with a lion and a penguin, we might have a parent 'Animal' class with the attributes Name and Date of Birth, as well as a Move method in order to inherit these directly into the subclasses Person, Lion and Penguin. Therefore if we were to change the general attributes in the parent class, these would all be applied to the subclasses too. Encapsulation is a way of ensuring that methods and attributes have the right permissions set for accessing and altering data. This is sometimes called information hiding. These are usually termed as public and private and are usually used to stop data from being changed accidentally. Polymorphism is a property of OOP that allows the programmer to make a program accept any data that they want into a method and it will be able to cope with it. For instance, if we had a base class called shape, our area method should be able to accept circles, triangles etc and the area should still be calculated correctly. This usually uses a process called 'Overloading'. **Object oriented programming concepts (Naresh Proddaturi)** http://community.computingatschool.org.uk/resources/970 A pdf file containing an explanation of OO concepts.	▶ Click here

Thinking Conceptually

Common misconceptions or difficulties students may have
Students will find assembly language using LMC quite difficult at first, but there are now some very good resources out there to help students gain familiarity and confidence with the subject matter. Of course, some students may have already covered this at GCSE level.

It would be beneficial for students to keep a glossary of key terms. The activities given at the end of the document should help with this.

Conceptual links to other areas of the specification – useful ways to approach this topic to set students up for topics later in the course
A student may choose to use one of the languages discussed in this part of the specification for use in class, but it is usual for centres to steer students to something that has been taught. Students may end up using some features of object-oriented programming in their final projects.

Modes of addressing memory links with 1.2.1 – section b, which covers memory management.

Activities	Resource
Activity 1 – Different types of programming language *Part 1* Students are to be given the statements on the cards and are required to sort them into which type of programming language they belong to. This could be done individually by students or in groups. *Part 2* Each student should find one fact about that particular type of programming language that has not been listed.	Learner Resource 1
Activity 2 – Object oriented programming *Part 1* Complete the quiz given in Learner Resource 2. If students do not know the answers, give them. *Part 2* Taboo is a fun game where one person has to sit at the front of their group and they have a set of things that they must get the rest of their group to guess without using any of the words in the answer. For example, for Objects you might say "The abbreviation for this is OOP". You could divide students up into teams, and if anyone is a bit reluctant they could always be the judge/scorekeeper. Cards are given in Learner Resource 3 of keywords that you might want students to cover. This is a good way to see if they fully understand key terminology.	Learner Resource 2 Learner Resource 3
Activity 3 – Little Man Computer CPD pack (Mark Clarkson) http://community.computingatschool.org.uk/resources/1573 Great CPD pack that contains all that is needed to get going with LMC at KS4/5 including a practical booklet with simple activities that can be given to every student and then used later as a revision guide if needed. Topics covered are getting input from the user, addition, branching and loops.	Click here

Learner Resource 1 Activity 1 cards

See page 11

Low level	**Object oriented**
'Mnemonics' are used as programming code such as ADD or MOV	Makes use of the idea of classes, objects and the ideas of encapsulation, polymorphism and inheritance
Excellent for close control of the CPU (sometimes called bare metal programming). Many device drivers are coded in assembly language	Classes can be treated as 'black boxes'. Other coders do not need to know how the class works internally. They just need to know what it is supposed to do
Machine code	Code portability is a useful feature of these languages. You need the right compiler for the target CPU and the code can be run on a different hardware platform
Assembly language	JAVA
They are specific to the CPU you are using, making direct use of internal registers	C++
Need to know a lot of detail about the internal structure of the CPU such as its memory management and registers	'Design patterns' are available that solve common programming tasks. A coder can use an existing design pattern for a user interface and begin coding straight away

Declarative	Procedural
Commonly used in artificial intelligence systems	You code specific instructions for the computer to carry out. This can be thought of as the 'do-this, then-this, then-this' style of programming
The software will seek an answer by interrogating a database containing Facts and Rules	C
Prolog	Pascal
	FORTRAN

Functional
These languages are mostly concerned with providing answers to problems purely through applying calculations to input data
These types of languages are used a lot in highly mathematical areas such as engineering and finance
LISP
Mathematica

Learner Resource 2 Activity 2: Part 1 – quiz

See page 11

1. **Which of the following is not a part of OOP?**
   a) Encapsulation
   b) Multitasking
   c) Polymorphism
   d) Information hiding

2. **What is the template of an object called?**
   a) Stencil
   b) Class
   c) Object template
   d) Blueprint

3. **OOP promotes a way of programming that allows programmers to think in terms of:**
   a) Data
   b) Procedures
   c) Objects
   d) People

4. **Information Hiding can also be termed as:**
   a) Cloaking
   b) Inheritance
   c) Moulding
   d) Encapsulation

5. **The process by which one object can acquire the properties of another object is:**
   a) Inheritance
   b) Encapsulation
   c) Polymorphism
   d) Blueprints

6. **A big advantage of OOP over traditional programming is:**
   a) The objects are all declared private
   b) The ability to reuse classes
   c) The convenience of giving all objects in a project the same name
   d) All of the above

7. **Constructors are used to:**
   a) Build a GUI
   b) Free memory
   c) Initialize a newly created object
   d) Create a subclass

8. **Two or more methods with the same name in the same class with different arguments is called:**
   a) Method overriding
   b) Method over handling
   c) Method overloading
   d) Method overcompensating

Teacher Answers Activity 2: Part 1 – quiz

	Answer
1. Which of the following is not a part of OOP? a) Encapsulation b) Multitasking c) Polymorphism d) Information hiding	B
2. What is the template of an object called? a) Stencil b) Class c) Object template d) Blueprint	B
3. OOP promotes a way of programming that allows programmers to think in terms of: a) Data b) Procedures c) Objects d) People	C
4. Information Hiding can also be termed as: a) Cloaking b) Inheritance c) Moulding d) Encapsulation	D
5. The process by which one object can acquire the properties of another object is: a) Inheritance b) Encapsulation c) Polymorphism d) Blueprints	A
6. A big advantage of OOP over traditional programming is: a) The objects are all declared private b) The ability to reuse classes c) The convenience of giving all objects in a project the same name d) All of the above	B
7. Constructors are used to: a) Build a GUI b) Free memory c) Initialize a newly created object d) Create a subclass	C
8. Two or more methods with the same name in the same class with different arguments is called: a) Method overriding b) Method over handling c) Method overloading d) Method overcompensating	C

Learner Resource 3 Activity 2: Part 2 – cards

See page 11

Object	Class	Attribute	Method
Inheritance	Encapsulation	Polymorphism	Parent (super) class
Child (sub) class	Information hiding	Overloading	Private
Public	Constructor		

OCR customer contact centre

General qualifications

Telephone 01223 553998

Facsimile 01223 552627

Email general.qualifications@ocr.org.uk

A DIVISION OF
CAMBRIDGE ASSESSMENT