UNIT 21

Software Engineering

STARTER

1 Put these five stages of programming in the correct sequence.

a Design a solution
b Code the program
c Document and maintain the program
d Clarify the problem
e Test the program

2 To which stage do each of these steps belong?

1 Clarify objectives and users
2 Debug the program
3 Write programmer documentation
4 Do a structured walkthrough
5 Select the appropriate programming language

LISTENING

3 You are going to hear an interview between a systems analyst and a hotel owner who wants to introduce a better computer system. What questions do you think the analyst will ask? Make a list; then compare your list with others in your group.

4 Listen to the recording to compare your list of questions with those asked by the analyst.

5 Listen again to find the answers to these questions:

1 What system does the hotelier have at present?
2 What problem is there with the existing system?
3 What form of output does the hotelier want?
4 Who will use the new system?
5 Which members of staff will require the most training?
6 What concerns has the hotelier about the new system?
7 What kind of hardware will be required?
8 What is the next step?

LANGUAGE WORK Revision: *If X, then Y*

In this section, we will revise structures commonly used in programming. You have met these structures in earlier units but in different contexts.

Study this decision table. It shows the rules that apply when certain conditions occur and what actions to take. Using it, we can make rules like this:

1 *If a* guest stays 3 nights in January and *if one* night is Sunday, *then* charge 2 nights at full price and 1 night at half-price.

2 *If a* guest stays 3 nights and one night is not Sunday and it is not January, *then* charge 3 nights at full price.

CONDITIONS	DECISION RULES	
	1	2
guest stays 3 nights	Y	Y
1 night is Sunday	Y	N
month is January	Y	N
Actions		
charge 3 nights at full price	N	Y
charge 2 nights at full price	Y	N
charge 1 night at half-price	Y	N

6 Now make similar statements about this decision table.

Conditions	Decision Rules					
	1	2	3	4	5	6
guest books bed and breakfast	Y	Y	Y	N	N	N
guest books half-board	N	N	N	Y	Y	N
guest books full-board	N	N	N	N	N	Y
and guest has lunch	N	Y	N	N	Y	–
and guest has dinner	N	N	Y	–	–	–
Actions						
charge rate A	Y	Y	Y	N	N	N
charge rate B	N	N	N	Y	Y	N
charge rate C	N	N	N	N	N	Y
charge menu price less 20%	N	Y	Y	N	Y	N

LANGUAGE WORK Do *until*, do *while*

Study these extracts from a program flowchart. They show iteration or loop structures in which a process is repeated as long as certain conditions remain true.

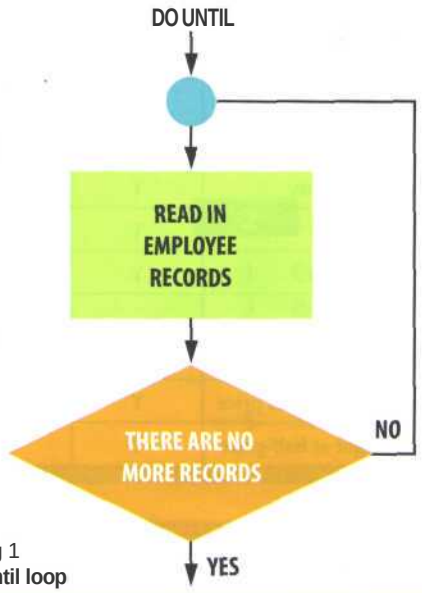


Fig 1
Until loop

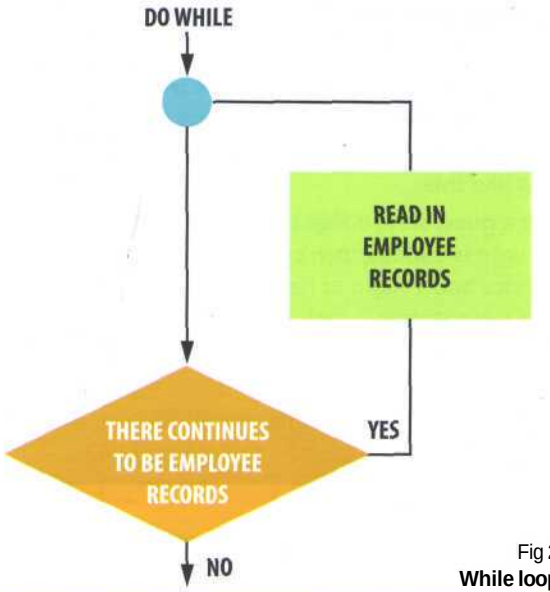


Fig **2**
While loop

We can describe these structures like this:

1 Read in the employee records *until* there are no more employee records.

Note that *until* links an action and the limit of that action.

2 Read in the employee records *while* there continues to be employee records.

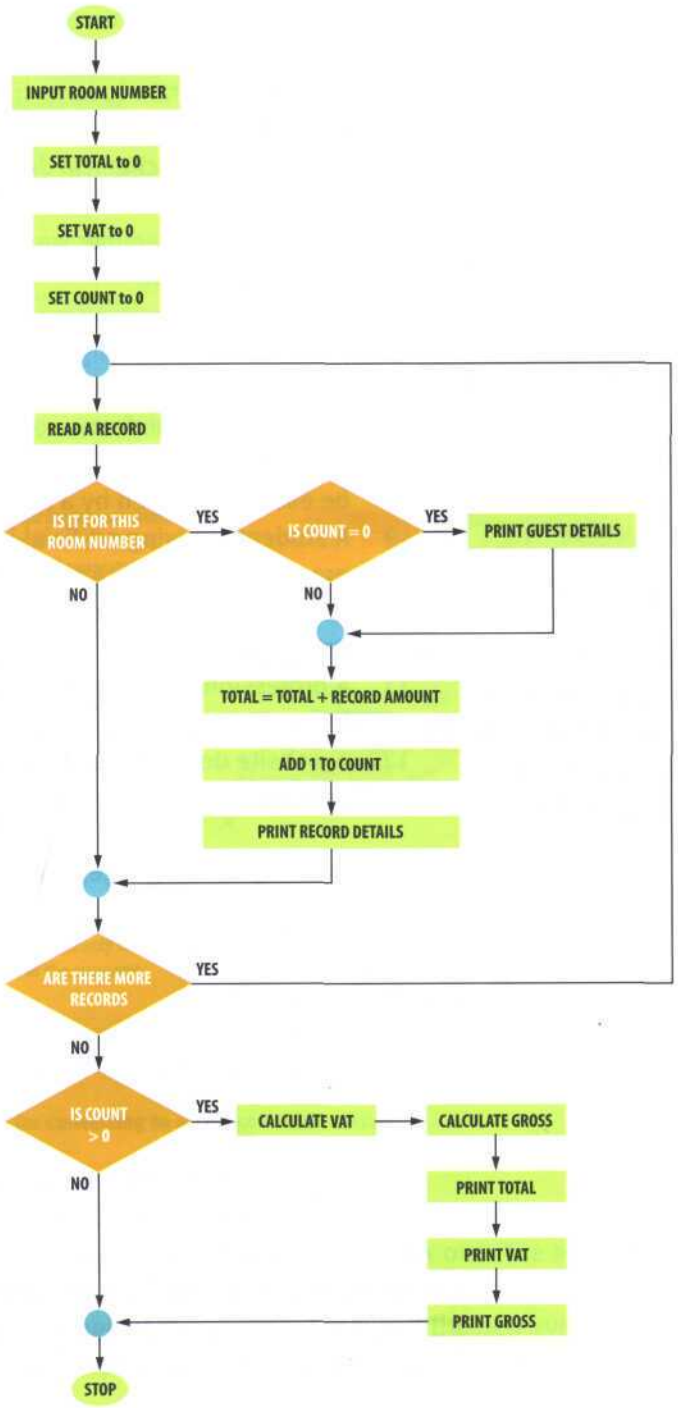
Note that *while* links actions or states happening at the same time.

7 Link these statements with *while* or *until,* whichever is most appropriate.

1 Calculate all sales. There are no more sales.
2 Search for records containing the term. There are still records containing the term.
3 Total extra items. Extra items remain.
4 Search member records. There are no more records.
5 Print all addresses. There are still addresses available.
6 Display client names. There are no names remaining.
7 List all guests. There are no guests left.
8 Total monthly sales. There are no more sales for the current year.

8 Flowcharts are sometimes used for designing parts of programs. Describe this extract from a program flowchart using the structures revised in this unit and the sequence expressions listed in Unit 2, Task 11.

Fig 3
Hotel accommodation invoicing flowchart



SPEAKING **9** Work in pairs, A and B. You each have information about some programming languages. Together decide what would be the most appropriate language to use for each of these situations.

1 A schoolteacher wants his young pupils to learn some basic mathematics by controlling a simple robot.
2 The owner of a small business wants to create a simple database program to keep track of his stock.
3 An engineer wants to develop a program for calculating the stresses in a mechanical device.
4 A student wants to create webpages for a personal website.
5 A systems programmer wants to add some new modules to an operating system.
6 A programmer working for the US army wants to create a program for controlling a new type of weapon.
7 A finance company needs to process data from its branch offices on its mainframe computer.
8 A website designer wants to enable the data on his website to be easily processed by a number of different programs.
9 A student studying artificial intelligence wants to write some programs for a course project.
10 A college lecturer wants his students to learn the principles of programming.
11 A professional programmer wants to create and sell a program for use in language learning.
12 A website designer wants to password-protect a section of a website.

Student A Your languages are on page 188.
Student B Your languages are on page 194.

WRITING **10** **Converting to a new system** Write a paragraph describing each of these strategies for converting to a new computer system. Explain what its advantages and disadvantages are. The first strategy is described for you as an example.

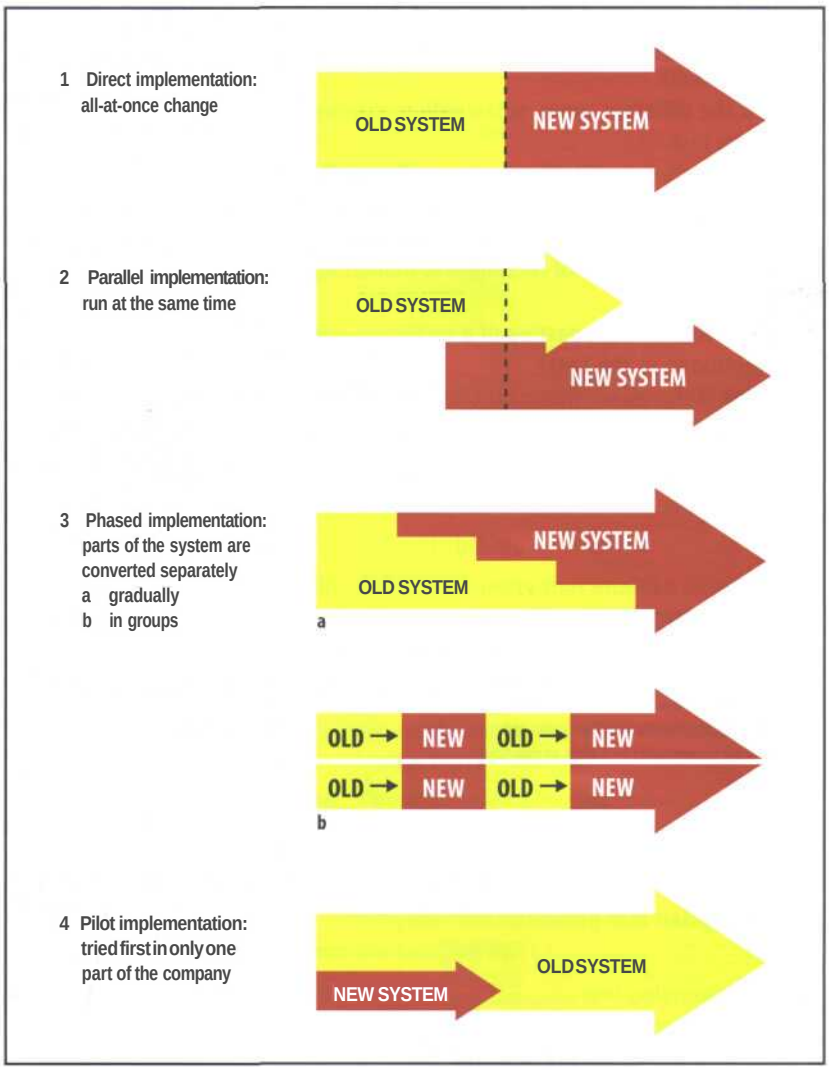


Fig 4
Strategies for converting to a new computer system

1 *Direct implementation:*
Direct implementation means that the user simply stops using the old system and starts using the new one. The advantage is that you do not have to run two systems at the same time. The disadvantage of this approach is that if the new system does not operate properly, there is nothing to fall back on.

SPECIALIST READING

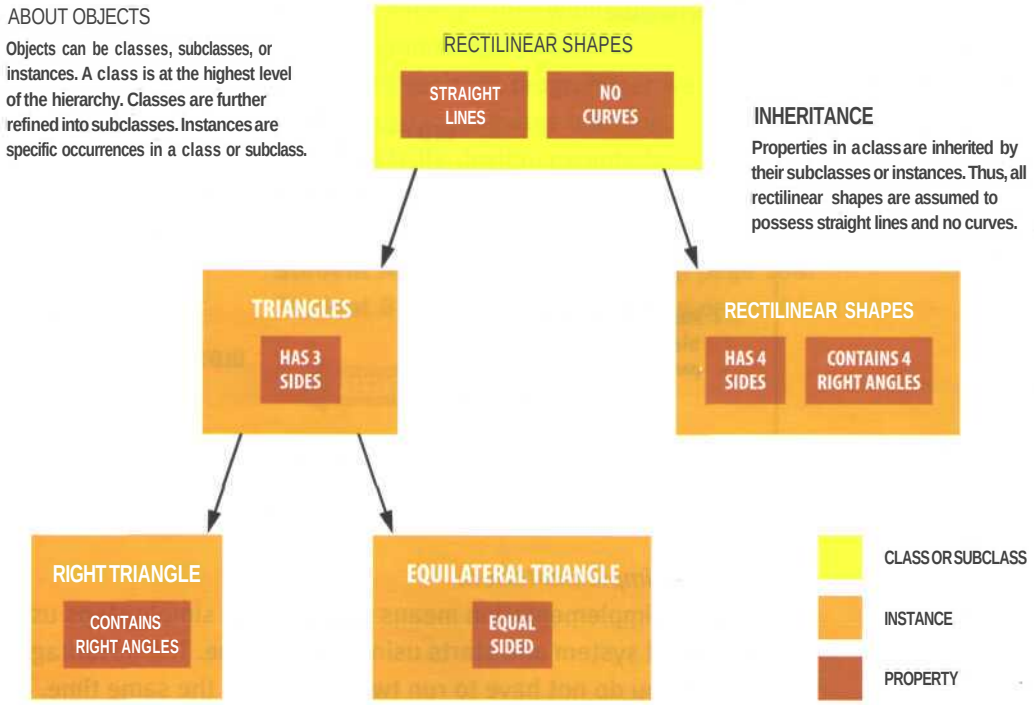
A Find **the answers to these questions in the following text.**

1 What advantages of using object-oriented programming are mentioned in the text?
2 What are the three key features of OOP?
3 What multimedia data types are referred to in the text?
4 List the different types of triangle mentioned in the text.
5 What feature avoids the problem of deciding how each separate type of data is integrated and synchronized into a working whole?
6 What specific type of rectangle is named in the text?
7 What common properties of a rectangle are mentioned in the text?
8 What features are made quicker by code reusability?

OBJECT-ORIENTED PROGRAMMING

One of the principal motivations for using OOP is to handle multimedia applications in which such diverse data types as sound and video can be packaged together into executable modules. Another is writing program code that's more intuitive and reusable; in other words, code that shortens program-development time.

Perhaps the key feature of OOP is encapsulation - bundling data and program instructions into modules called 'objects'. Here's an example of how objects work. An icon on a display screen might be called 'Triangles'. When the user selects the Triangles icon - which is an object composed of the properties of triangles (see fig. below) and other data and instructions - a menu might appear on the screen offering several choices. The choices may be (1) create a new triangle and (2) fetch a triangle already in storage. The menu, too, is an object, as are the choices on it. Each time a user selects an object, instructions inside the object are executed with whatever properties or data the object holds, to get to the next step. For instance, when the user wants to create a



triangle, the application might execute a set of instructions that displays several types of triangles - right, equilateral, isosceles, and so on.

Many industry observers feel that the encapsulation feature of OOP is the natural tool for complex applications in which speech and moving images are integrated with text and graphics. With moving images and voice built into the objects themselves, program developers avoid the sticky problem of deciding how each separate type of data is to be integrated and synchronized into a working whole.

A second key feature of OOP is inheritance. This allows OOP developers to define one class of objects, say 'Rectangles', and a specific instance of this class, say 'Squares' (a rectangle with equal sides). Thus, all properties of rectangles - 'Has 4 sides' and 'Contains 4 right angles' are the two shown here - are automatically inherited by Squares. Inheritance is a useful property in rapidly processing business data. For instance, consider a business that has a class called 'Employees at the Dearborn Plant' and a specific instance of this class, 'Welders'. If employees at the Dearborn plant are eligible for a specific benefits package, welders automatically qualify for the package. If a welder named John Smith is later relocated from Dearborn to Birmingham, Alabama, where a different benefits package is available, revision is simple. An icon representing John Smith - such as John Smith's face - can be selected on the screen and dragged with a mouse to an icon representing the Birmingham plant. He then automatically 'inherits' the Birmingham benefit package.

A third principle behind OOP is polymorphism. This means that different objects can receive the same instructions but deal with them in different ways. For instance, consider again the triangles example. If the user right clicks the mouse on 'Right triangle', a voice clip might explain the properties of right triangles. However, if the mouse is right clicked on 'Equilateral triangle' the voice instead explains properties of equilateral triangles.

The combination of encapsulation, inheritance and polymorphism leads to code reusability. 'Reusable code' means that new programs can easily be copied and pasted together from old programs. All one has to do is access a library of objects and stitch them into a working whole. This eliminates the need to write code from scratch and then debug it. Code reusability makes both program development and program maintenance faster.

[Adapted from 'Understanding Computers Today and Tomorrow', 1998 edition, Charles S. Parker, The Dryden Press]

B Re-read the text to find the answers to these questions.

1 Match the terms in Table A with the statements in Table B.

Table A

a	OOP
b	Encapsulation
c	Object
d	Menu
e	Square
f	Polymorphism
g	Library

Table B

i	An OOP property that allows data and program instructions to be bundled into an object
ii	A list of choices
iii	An OOP property that enables different objects to deal with the same instruction in different ways
iv	A reusable collection of objects
v	A module containing data and program instructions
vi	Object-Oriented Programming
vii	A rectangle with equal sides

2 Complete the following text using words from the reading text:

Encapsulation, ... and polymorphism are key features of ..... programming. Encapsulation allows data and program instructions to be bundled together in ..... called objects. Inheritance means that specific ..... of a class of objects ..... the properties of the class of objects. Polymorphism means that instructions are treated differently by different..... . The combination of these ..... features of OOP means that program code is reusable. This speeds up ..... and ..... of programs.