Higher Computing Science

# **Types of Languages**

## Recap

1. Describe what is meant by a high-level language. (2)

2. Describe what is meant by a low-level language. (2)

3. Explain the need for a translator. (2)

4. Describe the difference between a compiler and an interpreter. (2)

## Procedural

A procedural language will consist of instructions in a specific order that a program will follow. LiveCode is an example of a procedural language. It makes use of subprograms (such as procedures and functions), control structures and standard arithmetic operations (such as +, -, \*, /, etc). A set of instructions will be written which are executed in order to provide a solution to a problem. The set of instructions will have a definite beginning and end and will describe exactly what the computer must do at each step. These instructions are executed in written order by the computer.

5. Describe the difference between a procedure and a function. (2)

6. Parameters can be passed to a subprogram either by a value or reference. Describe the difference between each. (2)

7. Describe the difference between a formal parameter and an actual parameter. (2)

8. Describe the purpose of the arithmetic operator ^. (1)

## Declarative

Declarative languages make use of facts and rules that are contained within a knowledge base. This knowledge base can be questioned by making use of queries. It is most commonly used in Artificial Intelligence programming as decisions can be made based on these facts and rules.

### Example

The knowledge base shown below consists of facts and rules based on cars.

|  |
| --- |
| person(alan)person(david)drives\_car(alan, peugeot)drives\_car(david, ferrari)drives\_fast\_car(X) :- drives\_car(X, ferrari) |

The first four lines are examples of facts:

* Alan is a person
* David is a person
* Alan drives a peugeot car
* David drives a ferrari car

The last line is an example of a rule:

* Person X drives a fast car if person X drives a ferrari

We could ask the program to tell us whether David drives a fast car by typing the query:

|  |
| --- |
| ?drives\_fast\_car(david) |

This will replace the X in the drives\_fast\_car(X) rule with David, and if a fact is found that matches drives\_car(david, ferrari) then the result True would be returned, otherwise False will be returned.

### Question Set A

|  |
| --- |
| Fact: spouse (john, jane)Fact: spouse (david, mary)Fact: spouse(george, susan)Fact: female (jane)Fact: female (mary)Fact: female (susan)Fact: male (john)Fact: male (david)Fact: male (george)Rule: husband(A,B) IF spouse(A,B) AND male (A)Rule: wife (A,B) IF spouse(A,B) AND female(B) |

9. ?wife(david, mary)

10. ?husband(george, jane)

11. ?male(X) \*Hint: This query will return all possible results for X\*

12. ?spouse (john, X)

13. ?female (stephanie) (5)

### Question Set B

|  |
| --- |
| direct(glasgow, london)direct(london, paris)direct(paris, rome)direct(paris, seville)direct(rome, berlin)fly\_direct(P,Q):-direct(P,Q)one\_stop(X,Y):-fly\_direct(X,Z),fly\_direct(Z,Y) |

14. ? fly\_direct(paris, rome)

15. ? fly\_direct(paris, berlin)

16. ? one\_stop(paris, berlin)

17. ? one\_stop(glasgow, X) (4)

### Question Set C

|  |
| --- |
| warm(fred)warm(joe)warm(jim)warm(jack)fed(fred)fed(francis)fed(justin)cold(jim)cold(justin)happy(person) :- warm(person), fed(person)unhappy(person) :- cold(person) |

18. ?warm(fred)

19. ?happy(jack)

20. ?happy(fred)

21. ?unhappy(X) (4)

## Object-Oriented

Object-oriented programming languages create re-usable blocks of code (called objects) which define data used within that block and how that data can be changed. Data and code are treated as part of the same object. A class acts as a template for an object and contains attributes and states. An attribute (sometimes called fields or instance variables) will describe an object and a state will describe the value of an attribute at any given time.

### Example

|  |
| --- |
| ***Car Class*** |
| **Attribute** | **State** |
| Manufacturer | Peugeot |
| Fuel Type | Diesel |
| Model | 208 |
| Number of Seats | 5 |
| Number of Doors | 5 |

In an example of 2-D vector graphics there would be a class called 'Shape'. Each type of shape within the class is known as a sub class. In the following example 'Shape' is a super class, meaning that there are a large number of different sub classes that could be thought of as shapes. As well as super classes and sub classes, there are also instances shown. An instance is a single member of a sub class. The procedures and functions which can change that data are called its methods.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Class** |  |  |  |  | **Shape** |  |  |  |
|  |  |  |  |  |  |  |  |  |
| **Sub Class** |  |  | **Triangle** |  |  |  | **Line** |  |
|  |  |  |  |  |  |  |  |  |
| **Instance** |  | **Equilateral** |  | **Isosceles** |  | **Straight Line** |  | **Curved Line** |

Objects are closed systems which cannot be altered from outside which makes errors caused by the use of global variables less likely.

This feature is called encapsulation. The benefit of using this technique is that once a class has been defined and tested, sub-classes can then be created which share the main characteristics of the parent class, thus saving testing and development time.

This ability to create a sub-class from a pre-defined class is called inheritance.

Whereas a procedural language program will be built from a number of procedures called from a main procedure, an object oriented program will have a number of methods which are specifically linked to the class within which they are defined, in keeping with the idea of keeping classes and objects self-contained.

Object-oriented table

|  |  |  |
| --- | --- | --- |
| **Object** | **Instance Variables** | **Methods** |
| Button  | Name, Size, Position, Colour | Click, Mouse-over |
| Window | Name, Size, Position, Focus, Border | Maximise, Minimise, Resize, Open, Close |
| Dialogue Box | Contents, Priority | Open, Close, OK, Cancel |

### Questions

22. Give three characteristics of object-oriented programming languages. (3)

23. Explain the concept of inheritance and why it is an important feature of object-oriented programming. (2)

24. Match the words to the correct nodes to complete the inheritance diagram:



(3)

25. Describe the term method. (2)