Programming Languages Exercises

Questions

Q1: Why programs written with Assembly languages are machine dependent?

Q2: What is the difference between a *compiler* and an *interpreter*? Give an example for both.

Q3: What are the four programming paradigms? Give an example for each of them.

Q4: What is the difference between *pass by value* and *pass by reference*?

Q5: What is called to a variable defined in the function?

Q6: What is the purpose of the *lexical analyzer*?

Q7: Which of the following allocates memory space when created: object or classes?

Q8: What is encapsulation?

Q9: What is inheritance?

Q10: What is Abstraction?

Q11: What is a function?

Answers

A1: Because Assembly language is different for every microprocessor family. That is why, for the microprocessor you want to run your code, you should write the program in the Assembly language for that microprocessor.

A2: Compilers translate programs into executable form. Interpreters do not create an executable copy of the code, it executes the program as they are translated. C++ is an example of a compiled language and Python is an example of an interpreted language.

A3: Imperative (procedural), Object Oriented, Functional and Declarative. Imperative: *C, Python, Perl*; Object Oriented: *Java, C++*, Functional: *LISP, Haskell, Scheme*; Declarative: *Prolog, SQL, AMPL.*

A4: In *pass by value*, any changes done in by the invoked function does not effect the original value. In *pass by reference*, when you call a function and make some changes; it will change the original value. (Check the slides and examples for better understanding. Try to write the example codes and learn them by trying, editing them.)

A5: Local variable.

A6: The purpose of the lexical analyzer is that it converts the source code into higher level tokens so that a parser can understand them.

A7: When we create classes, no memory allocation happens. However, when we create objects, it allocates memory space.

A8: Encapsulation prevents direct access and it hides the values or state of an object from others. It is not unique to Object Oriented languages.

A9: Inheritance is the mechanism of basing an object or class upon another object or class. So, if you have an animal class, you can create a dog class and say that it is derived from the Animal class. That means that every method the Animal class have, the *dog* class will also have. Inheritance lets us inherit attributes and methods from another class.

A10: Abstraction is the process of hiding the implementation details and showing only functionality to the user. It shows only the essential things.

A11: Function is a set of instructions for performing a task that can be used by other program units. It divides a large program into managable units.