Student Name: Federico Watkins

CSIS 1800: Introduction to Computer Science and Information Systems

Chapter number: 9 High Level Programming Languages

Assignment number: 9

1. What is a data type? Define the following data types and give one example per type.
   1. Integer

* A data type which represents the integer range values from smallest to largest values
* A whole number consisting of positive or negative number, or zero. So examples of numbers that are considered integers are: 5, -6, 0 and 983
* Example: int j = 3;
  1. Real
* A data type which represents the range of real numbers from smallest to largest values. Most computers use a rational approximation to real numbers
* The rational numbers stores the numerator and denominator as an integer number. Hence the following float values are considered real: 8.45, 2.14, and 5.1.
* Example: float k = 2.78;
  1. Character
* A data type which represents the characters in the character set
* I may be a character, letter, number, digit, punctuation, or symbol. Hence, ‘w’, ‘7’, and ‘.’ Are considered characters.
* Example: char p = ‘L’;
  1. Boolean
* A data type which represents two values
* The values are designated true or false, sometimes implemented as 0 or 1.
* Example: boolean empty = true;

1. Given Boolean variables one, two, and three*,* write an equation for each of the following questions.
2. Is one greater than both two and three?

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| (one | > | two) | AND | (one | > | three) |

1. Is one greater than two, but less than thre*e*?

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| (one | > | two) | AND | (one | < | three) |

1. Are all three variables greater than zero?

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| (one | > | 0) | AND | (two | > | 0) | AND | (three | > | 0) |

1. Is one less than two or one less than three?

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| (one | < | two) | OR | (one | < | three) |

1. Is *two* greater than one and three less than two?

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| (two | > | one) | AND | (three | < | two) |

1. Describe the process of compiling and running a C++ and Java programs.

* **Compiling**:

The high-level language is converted into bytecode or standard machine code by use of a compiler. The compiler is the translator used to convert the programs in high-level language too machine-level language. Machine code or object code is required to execute a program

* **Execution**:

Byte code and machine code differ based on the type of operating system for any particular machine. With java for example, the java virtual machine has platform-independent execution so that it can run in any environment. Depending on the environment, the JVM interpreter converts the bytecode into machine code and executes it.

1. How do the .WORD and .BLOCK assembler directives in the Pep/8 assembly language differ from the declarations in high-level languages?

* The process or functionality of the assembly language is the same. It allocates the memory for the particular variable and stores the values into it.
* For the assembler directives .WORD and .BLOCK, the programmer knows the actual address of the identifier. In high-level language, the programmer needs not know about actual address of the identifier.

1. *Age*, *weight*, and *height* are three integer variables and a read statement says to input them in that order. Given the following input stream (commas are separators) 03, 118, 80 what is stored in?
2. *Age*?

|  |  |  |
| --- | --- | --- |
| Age | = | 03 |

1. *Weight*?

|  |  |  |
| --- | --- | --- |
| Weight | = | 118 |

1. *Height*?

|  |  |  |
| --- | --- | --- |
| height | = | 80 |

Do the values stored in the variables seem reasonable?

If not, how would you correct the problem?

1. Object-oriented strategy:
2. List the four stage strategy for developing an object-oriented decomposition

* Brainstorming
* Filtering
* Scenarios
* Responsibility algorithms

1. Outline the characteristics of each stage

* Brainstorming, in which we make a first pass at determining the classes in the problem
* Filtering, in which we review the proposed classes looking for duplicates or missing classes. Here functional programming techniques are appropriate. Inheritance relationships are established.
* Scenarios, in which the responsibilities of each class are determined and “what if” questions are asked. Collaborators to a responsibility are assigned. Results in fully developed CRC cards for all classes.
* Responsibility algorithms, in which the algorithms are written for each of the responsibilities listed on a CRC card. Results in the design ready to be translated into a program.

1. Are the stages independent? Explain!

The strategies stages are intricately intertwine as they depend on each other almost as if to be done in order. Each stages output yields the input for the following stage. Hence they are not independent.

1. Compare and contrast an assembler, a compiler, and an interpreter.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Assembler** | **Compiler** | | **Interpreter** | |
| Assembler is a translator used to convert the programs in assembly language to machine language. | Compiler is a translator used to convert the high-level language into machine language or low-level language. | | Interpreter is a translator used to convert the high-level language into machine language or low-level and executes immediately | |
| It produces object code | It produces executable code | | It produces intermediate code | |
| The input of assembler is assembly language which is machine-dependent | The input of compiler is high-level language which is machine-dependent | | The input of interpreter is high-level language which is machine-dependent | |
| The assembler takes each instruction in a program as an input and executes at a time | The compiler takes the entire program as an input and executes at an time | | The interpreter takes each single line instruction as an input and executes each and every line individually in a program | |
| If the assembly code has an error, it may terminate execution | The compiler produces the list of errors after the entire program is executed | | The interpreter immediately stops the execution after it finds the errors and it does not proceed to execute the next instruction before it clears the errors | |
| It is more difficult to understand by human and it produces the efficient execution | | It is human readable code. It does not produce the efficient execution and it is faster to run the program | | It is human readable code. It is useful for testing process and it is slower to run the program |
| Example for assembler:  Assembly Language. | | Example for compiler:  C, C++ languages | | Example for interpreter:  BASIC Language |

1. Match the question with the appropriate translation or execution system
2. Interpreter
3. Assembler
4. Compiler
5. Machine code

|  |  |  |
| --- | --- | --- |
|  | 1. What translates a high-level language into machine code?  |  | | --- | | C | |
|  | 1. What translates a Java program into Bytecode?  |  | | --- | | C | |
|  | 1. What executes Bytecode?  |  | | --- | | A |  1. What executes the Java Virtual Machine?  |  | | --- | | D | |
|  | 1. What translates an assembly language program?  |  | | --- | | B | |
|  | 1. What is the output of an assembler?  |  | | --- | | D | |
|  |  |

1. The obvious way to place an item into unsorted list is different in an array-based and a linked implementation. Explain!

* Array-based:

The item is pushed onto the stack of arbitrary size and placed in the [0] index

* Link-based:

The item is assigned as the head of the link and points to the tail.

1. Which of the following languages is more suitable for solving which specific problems?

* C

|  |
| --- |
| Procedural |

* BASIC

|  |
| --- |
| Procedural |

* FORTRAN

|  |
| --- |
| Procedural |

* COBOL

|  |
| --- |
| Procedural |

* C++

|  |
| --- |
| Procedural with object-oriented features |

* Ruby

|  |
| --- |
| Object oriented with procedural features |

* Python

|  |
| --- |
| Object-oriented |

* PHP

|  |
| --- |
| Scripting |

References:

1. Computer Science Illuminated, Nell Dale and John Lewis, chapter 9.
2. <http://www.umsl.edu/~sauterv/analysis/488_f01_papers/quillin.htm>
3. <https://en.wikipedia.org/wiki/COBOL>
4. <https://en.wikipedia.org/wiki/Ruby_(programming_language)>