Structured programming

This type of programming is called structured because follows an order in how the actions are done, also, structured programming has the characteristic of be readable and elements of codes can be reused.

In a code we used a language understandable for humans, but this code must be read by the computer, so it must be converted from a programming language to a language that the computer understands. This is called a compilation, and it has six different stages.

The first stage is the **lexical analysis,** here the compiler transforms the high-level programming in a serie of tokens, a token is a sequence of characters which the programming language reads as a unit.

The next step is the **symbol table,** where the compiler stores information, such as variables, constants, procedures, functions and more things.

The **syntax analysis** checks the tokens and shows us if there is a syntax error or not, the tokens must coincide with the language with which you are working.

Then the **semantic analysis** makes sure the program is semantically correct, for this, the symbol table and the information of the syntax analysis are used to compare the consistency of the code.

In **code generation stage**, the high-level code turns into a low-level code which the computer can understand.

Finally, the code **optimization** makes the program faster and uses fewer resources.

Levels of programming

Machine language

This is the lowest level of language; it consists in instructions in binary code (0,1). This language is the understood by the computer, but for humans is really hard to understand and to write.

```
01001000 01100101 01101100 01101100 01101111 00100000 01010111
01101111 01110010 01101100 01100100
```

Figure 1. Example of machine language for the text "Hello World."

Assembly language

This language is a middle point between the machine language and the high-level language, since this language has some words in English but is simpler than the high-level language.

In addition, this language only can be used for a specific family of processors, because each family of processors have their own set of instructions.

```
global  _main
    extern  _printf
    section .text
_main:
    push    message
    call    _printf
    add     esp, 4
    ret
message:
    db  'Hello, World!', 10, 0
```

Figure 2. "Hello, World" written for a 32-bit Intel processor

High-level language

This type of language allows the programmers make programs without restriction of a particular computer or processor; they are closer to the human language, for this they are called language of high-level, the logic of the process must be given by the programmer and a compiler is used to change from high-level to machine level language.

```
while count < 10:
#While the value of count is less than ten

      number = int(input("Type in a number"))
      #Input a number

      total = total + number
      #Add the number to the total

      count = count + 1
      #Add one to the value of count

print("The total is ", total)

#Print out the total
```

Figure 3. High-level example

References

Nolle. T. (n.d.). Structured programming (modular programming). TechTarget. Retrieved from https://searchsoftwarequality.techtarget.com/definition/structured-programming-modular-programming

Computer Hope. (April 26, 2017). Compilation. Retrieved from https://www.computerhope.com/jargon/c/compilat.htm#:~:text=Compilation%20is%20the%20process%20the,language%2C%20Machine%20language%2C%20Programming%20terms

Bitesize. (n.d). Program construction. BBC Retrieved from https://www.bbc.co.uk/bitesize/guides/zmthsrd/revision/3

Geeks for Geeks. (June 09, 2020). Introduction of Lexical Analysis. Retrieved from https://www.geeksforgeeks.org/introduction-of-lexical-analysis/

Geeks for Geeks. (January 31, 2019). Symbol table in compiler. Retrieved from https://www.geeksforgeeks.org/symbol-table-compiler/

Geeks for Geeks. (November 21, 2019). Introduction to Syntax Analysis in compiler design. Retrieved from https://www.geeksforgeeks.org/introduction-to-syntax-analysis-in-compiler-design/

Geeks for Geeks. (April 22, 2020). Semantic Analysis in compiler design. Retrieved from https://www.geeksforgeeks.org/semantic-analysis-in-compiler-design/

Tutorialspoint (n.d.). Assembly-Introduction. Retrieved from https://www.tutorialspoint.com/assembly_programming/assembly_introduction.htm

JavaTpoint (n.d.). What is a programming language? Retrieved from https://www.javatpoint.com/classification-of-programming-languages

Computer Hope. (June 30, 2019). Machine language. Retrieved from https://www.computerhope.com/jargon/m/machlang.htm

Computer Hope. (October 07, 2019). Assembly language. Retrieved from https://www.computerhope.com/jargon/a/al.htm

Bitesize. (n.d.). Types of programming language. BBC. Retrieved from https://www.bbc.co.uk/bitesize/guides/z4cck2p/revision/1