Laboratio de exercicios 0

Luis Eduardo Castro Mendes de Lima

8 de janeiro de 2024

0.1 Programação Estruturada

A programação estruturada é um tipo de programação que geralmente converte programas grandes ou complexos em pedaços de código pequenos e mais gerenciáveis.

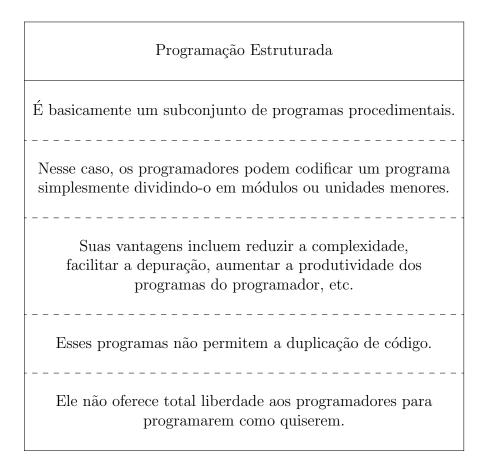
Esses pequenos códigos são geralmente conhecidos como funções ou módulos ou subprogramas de grandes programas complexos.

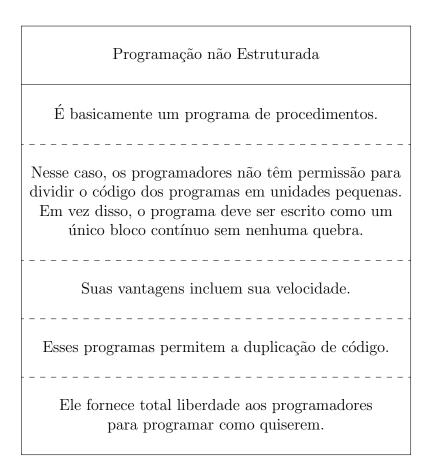
É conhecido como programação modular e minimiza as chances de uma função afetar outra função.

0.2 Programação não Estruturada

A programação não estruturada é um tipo de programação que geralmente executa em ordem sequencial, ou seja, esses programas simplesmente não saltam de nenhuma linha de código e cada linha é executada sequencialmente.

É também conhecida como programação não estruturada, capaz de criar algoritmos completos de torneamento.





0.3 Compilador e Interpretadior

O Compilador e o Interpretador têm trabalhos semelhantes a realizar. Ambos convertem o Código Fonte (HLL) em Código de Máquina (entendido pelo computador). Em geral, os programas de computador existem em Linguagem de Alto Nível que um ser humano pode entender facilmente. Mas os computadores não podem entender a mesma linguagem de alto nível, então precisamos convertê-los para linguagem de máquina e torná-los compreensíveis para os computadores.

0.3.1 O Compilador

O Compilador é um tradutor que recebe a entrada, ou seja, a Linguagem de Alto Nível, e produz uma saída em linguagem de baixo nível, ou seja, linguagem de máquina ou assembly. O trabalho de um compilador é transformar os códigos escritos na linguagem de programação em código de máquina (formato de 0s e 1s) para que os computadores possam entender.

0.3.2 Vantagens do Compilador

- O código compilado é mais rápido em comparação com o código interpretado.
- Os compiladores ajudam a melhorar a segurança das aplicações.
- Os compiladores fornecem ferramentas de depuração, que ajudam a corrigir erros facilmente.

0.3.3 Desvantagens do Compilador

- O compilador só pode detectar erros de sintaxe e alguns erros semânticos.
- A compilação pode levar mais tempo no caso de código volumoso.

0.3.4 Interpretador

Um Interpretador é um programa que traduz uma linguagem de programação para uma linguagem compreensível. O interpretador converte a linguagem de alto nível para uma linguagem intermediária. Ele contém código pré-compilado, código-fonte, etc.

- Ele traduz apenas uma instrução do programa de cada vez.
- Os interpretadores, na maioria das vezes, são menores que os compiladores.

0.3.5 Vantagens do Interpretador

| γ το βουστού μεταιού μ |
|--|
| • Programas escritos em uma linguagem interpretada são mais fáceis de depurar. |
| |
| |
| |
| Os interpretadores permitem a gestão automática da memória, o que reduz os riscos de erros de memória. |
| |
| |
| |
| • A linguagem interpretada é mais flexível do que uma linguagem compilada. |
| |
| |
| |
| 0.3.6 Desvantagens do Interpretador |
| • O interpretador só pode executar o programa interpretado correspondente. |
| |
| |
| |

 \bullet O código interpretado é mais lento em comparação com o código compilado.

0.4 Diferença entre Compilador e Interpretador

| Compilador | Interpretador |
|---|---|
| Diferenças: | Diferenças: |
| - Códigos compilados são mais rápidos | - Códigos interpretados são mais lentos |
| que interpretados. | que compilados. |
| - Modelo de trabalho básico do com- | - Modelo de trabalho básico do Interpre- |
| pilador é o Modelo de Vinculação- | tador é o Modelo de Interpretação. |
| Carregamento. | |
| - Compilador gera uma saída no formato | - Interpretador não gera nenhuma saída. |
| (.exe). | |
| - Qualquer alteração no programa fonte | - Qualquer alteração no programa fonte |
| após a compilação requer recompilar | durante a tradução não requer re- |
| todo o código. | tradução de todo o código. |
| - Erros são exibidos no compilador após | - Erros são exibidos em cada linha. |
| a compilação no momento atual. | |
| - Compilador pode ver o código ante- | - Interpretador funciona linha por linha, |
| cipadamente, o que ajuda na execução | por isso a otimização é um pouco mais |
| mais rápida devido à otimização. | lenta em comparação com compiladores. |
| - Não requer código fonte para execução | - Requer código fonte para execução pos- |
| posterior. | terior. |
| - Compiladores geralmente levam mais | - Em comparação, Interpretadores le- |
| tempo para analisar o código fonte. | vam menos tempo para analisar o código |
| | fonte. |
| - Utilização da CPU é maior no caso de | - Utilização da CPU é menor no caso de |
| um Compilador. | um Interpretador. |
| - Código-objeto é salvo permanente- | - Nenhum código-objeto é salvo para uso |
| mente para uso futuro. | futuro. |
| - Linguagens como C, C++, são basea- | - Linguagens como Python, Ruby, Perl, |
| das em compiladores. | SNOBOL, MATLAB são baseadas em |
| | interpretadores. |