Laboratio de exercicios 0

Luis Eduardo Castro Mendes de Lima

22 de janeiro de 2024

0.1 davi

0.2 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 sub-programas de grandes programas complexos.

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

0.3 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.

Programação Estruturada
É basicamente um subconjunto de programas procedimentais.
Nesse caso, os programadores podem codificar um programa simplesmente dividindo-o em módulos ou unidades menores.
Suas vantagens incluem reduzir a complexidade, facilitar a depuração, aumentar a produtividade dos programas do programador, etc.
Esses programas não permitem a duplicação de código.
Ele não oferece total liberdade aos programadores para programarem como quiserem.

Programação não Estruturada
É basicamente um programa de procedimentos.
Nesse caso, os programadores não têm permissão para dividir o código dos programas em unidades pequenas. Em vez disso, o programa deve ser escrito como um único bloco contínuo sem nenhuma quebra.
Suas vantagens incluem sua velocidade.
Esses programas permitem a duplicação de código.
Ele fornece total liberdade aos programadores para programar como quiserem.

0.4 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.

0.4.1 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.

0.4.2 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.

0.4.3 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.

0.4.4 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.

0.4.5 Vantagens do Interpretador

- 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.

0.4.6 Desvantagens do Interpretador

- O interpretador só pode executar o programa interpretado correspondente.

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

0.5 Diferença entre Compilador e Interpretador

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