Programação de Computadores

Instituto de Computação UFF
Departamento de Ciência da Computação

Otton Teixeira da Silveira Filho

Conteúdo

- Alguns Conceitos sobre Linguagens

  Paradigmas para linguagens de Programação

  Linguagens de alto, médio e baixo nível de abstração

  Processo de resolução de problemas

- Conceito de Algoritmo

Paradigmas de Programação

Paradigmas de Linguagens de Programação

Linguagem de programação imperativa

Linguagem de programação estruturada

Linguagem de programação Orientada a Objetos

Linguagem de programação Indutiva

Linguagem de programação Orientada por Semântica

Etc.

Ideias Gerais sobre Linguagens

Estes paradigmas de linguagens programação não são mutuamente excludentes

Podemos ter linguagens multiparadigmáticas

Ideias Gerais sobre Linguagens

Programação Imperativa

As linguagens de uso mais comum trabalham sobre o paradigma denominado Programação Imperativa

A Programação Imperativa descreve o processo de computação como ações, enunciados ou comandos que mudam as **variáveis (formas organizadas de informação)** definidas num programa.

São exemplos de linguagens imperativas

- Fortran, C, C++, Python, Java, Lua

Variáveis

Variáveis

São espaços reservados na memória do computador que serão lidos e escritos de uma forma pré-estabelecida

Ideias Gerais sobre Linguagens

Programação Estruturada

É uma forma de programação na qual os programas serão construídos usando três estruturas:

- de sequência: uma tarefa é executada após a outra, linearmente

- de decisão: partindo de testes lógicos, segmentos do código serão executados ou não

- de Iteração: partindo de testes lógicos, segmentos do código são repetidos um número finito de vezes

- Quase todas as linguagens atuais são estruturadas

Ideias Gerais sobre Linguagens

As linguagens podem ser

- Compiladas

  São linguagens que fazem uso de um programa chamado **compilador** que transforma o código fonte escrito numa linguagem em outro código equivalente em outra linguagem

- Interpretadas

  São linguagens que fazem uso de um programa chamado **interpretador** que permite que o sistema operacional ou o processador execute diretamente o código fonte escrito

Ideias Gerais sobre Linguagens

As linguagens podem ser

- Compiladas

  Exigem pelo menos um passo intermediário para a execução

  A execução é mais rápida que com as interpretadas

  São boas onde é necessário velocidade de execução

- Interpretadas

  Resultam numa aplicação mais direta

  São boas na construção de protótipos e ações de controle de processos

Ideias Gerais sobre Linguagens

As linguagens podem ser

- Compiladas

  Algol, Fortran, Pascal, C, C++

- Interpretadas

  Java, Lisp, Forth, Lua, PHP, Ruby, Python, R

Ideias Gerais sobre Linguagens

As linguagens podem ser

- Compiladas

  Algol, Fortran, Pascal, C, C++

- Interpretadas

  Java, Lisp, Forth, Lua, PHP, Ruby, Python, R

- Existem linguagens que podem usar recursos da compilação e da interpretação conjuntamente como Forth

Ideias Gerais sobre Linguagens

As linguagens podem ser

- De baixo nível

  São linguagens que acessam diretamente as características da arquitetura de um computador usando as instruções do processador. Dizemos que estamos escrevendo programas em linguagem de máquina e com **baixo nível de abstração**

- De alto nível

  Faz uso de estruturas das linguagens naturais facilitando a programação. Tem um **nível** mais **alto de abstração**

- De “médio” nível

  Conjuga o acesso à arquitetura com estruturas de alto nível

Ideias Gerais sobre Linguagens

As linguagens podem ser

- De baixo nível

  São os códigos de cada processador ou os chamados montadores (**Assemblers**)

- De alto nível

  Algol, Fortran, Pascal, Java, Lisp, Lua, PHP, Ruby, Python, R

- De “médio” nível

  C, C++, Forth

Ideias Gerais sobre Linguagens

As linguagens podem ser

- De baixo nível

  Resultam em programas altamente eficientes em termos de memória e tempo de execução
  Programação difícil exigindo muito treinamento

- De alto nível e “médio” nível

  São menos eficientes em termos de tempo de execução e uso de memória que as de baixo nível
  São de programação mais simples que as de baixo nível

Ideias Gerais sobre Linguagens

As linguagens podem ser

- De baixo nível

  Resultam em programas altamente eficientes em termos de memória e tempo de execução
  Programação difícil exigindo muito treinamento

- De alto nível e “médio” nível

  São menos eficientes em termos de tempo de execução e uso de memória que as de baixo nível
  São de programação mais simples que as de baixo nível

- Não se esqueça que, assim como todo e qualquer trabalho, pode se programar mal em qualquer linguagem

Algoritmo

- É um conjunto de operações, finitas em número, necessárias à resolução do problema

- Observe que pela definição o algoritmo é independente da linguagem de programação

- Em geral, a parte que exige maior esforço é a concepção do algoritmo

Algoritmo

- Podem existir problemas para os quais não existe algoritmo
- Em alguns casos isto se dá por não ser possível achar o que se pretende em um número finito de passos

Algoritmo

Vamos a um exemplo de algoritmo

Sequência de Fibonacci

A sequência de Fibonacci é gerada da seguinte forma:

- Seja $F_1 = 1$ e $F_2 = 1$
- Então $F_n = F_{n-1} + F_{n-2}$

Assim teremos a sequência: 1, 1, 2, 3, 5, 8, 13...

Sequência de Fibonacci

Vejamos alguns programas que geram a sequência de Fibonacci em várias representações

Sequência de Fibonacci

- Representado em binário

1000 1011 0101 0100 0010 0100 0000 1000 1000 0011 1111 1010
0000 0000 0111 0111 0000 0110 1011 1000 0000 0000 0000 0000
0000 0000 0000 0000 1100 0011 1000 0011 1111 1010 0000 0010
0111 0111 0000 0110 1011 1000 0000 0001 0000 0000 0000 0000
0000 0000 1100 0011 0101 0011 1011 1011 0000 0001 0000 0000
0000 0000 0000 0000 1011 1001 0000 0001 0000 0000 0000 0000
0000 0000 1000 1101 0000 0100 0001 1001 1000 0011 1111 1010
0000 0011 0111 0110 0000 0111 1000 1011 1101 1001 1000 1001
1100 0001 0100 1010 1110 1011 1111 0001 0101 1011 1100 0011

Sequência de Fibonacci

- Representado em hexadecimal

  8B542408 83FA0077 06B80000 0000C383 FA027706 B8010000 00C353BB 01000000 B9010000 008D0419 83FA0376 078BD989 C14AEBF1 5BC3

Sequência de Fibonacci

- Representado em Assembly

```
x86
https://en.wikipedia.org/wiki/Low-level_programming_language


fib:
    mov edx, [esp+8]
    cmp edx, 0
    ja @f
    mov eax, 0
    ret

    @@:
    cmp edx, 2
    ja @f
    mov eax, 1
    ret

    @@:
    push ebx
    mov ebx, 1
    mov ecx, 1

    @@:
        lea eax, [ebx+ecx]
        cmp edx, 3
        jbe @f
        mov ebx, ecx
        mov ecx, eax
        dec edx
    jmp @b

    @@:
    pop ebx
    ret
```

Sequência de Fibonacci

- Representado em C

```c
#include <stdio.h>
#include <math.h>
#include <stdlib.h>
#include <float.h>

int main(void)
{
int n, a, b, c, fibo;

a = 1;
b = 1;

printf("Entre com n ");
scanf("%d", &n);

printf("%d", n);

if (n <= 0) printf("\n n negativo\n");
else
    {
    if (n <= 2) fibo = 1;
    else
        {
        while (n > 2)
            {
            c = a + b;
            a = b;
            b = c;
            n--;
            }
        fibo = c;
        }
    printf("\n O n-esimo termo da sequencia de Fibonacci e' %d\n", fibo);
    }
}
```

Sequência de Fibonacci

- Representado em Python

```
def main():

    a = 1
    b = 1

    n = int(input('Entre com n '))

    if n <= 0 :
        print('n negativo')

    elif n <= 2 :
        fibo = 1
    else :
        while n > 2 :
            c = a + b
            a = b
            b = c
            n = n - 1

        fibo = c

        print('O n-esimo termo da sequencia de Fibonacci: ', fibo)
main()
```

Sequência de Fibonacci

- Representado em Python

  (Versão menos legível)

```
n = int(input('Digite n '))

a, b = 1, 1

print (b)
for i in range(1, n):
    a, b = b, a+b
    print (b)
```

Processo de Resolução de Problemas

-Compreender o problema para poder fazer o programa desejado:
Definindo dos requisitos do problema

Entradas

Cálculos

Decisões e repetições

Saídas

-Estabelecer um plano para resolver de maneira correta o problema:
Desenvolvendo do algoritmo da solução

Fluxograma

Pseudocódigo

Processo de Resolução de Problemas

-Codificar o programa:

Python no nosso caso

-Testar o programa. Detectar:

Defeitos na codificação

Defeitos na concepção da solução

Processo de Resolução de Problemas

Python é uma linguagem:

- Interpretada
- De alto nível
- Multiparadigmática (estruturada, imperativa, orientada a objeto, funcional)

Resumo

- Crie um algoritmo partindo da necessidade de resolver um problema
- Escolha a linguagem mais adequada
- Codifique o algoritmo nesta linguagem
- Teste o código escrito na linguagem:
  - Parte do teste são correções de escrita do código
  - Parte do teste é a avaliação dos resultados e a compatibilidade deles com o esperado

- O processo de detecção e correção de erros é chamado de **Depuração**

Tipos de erros

- Sintaxe:

A transcrição do algoritmo para a linguagem não seguiu as regras da mesma

- Lógico:

Os resultados não são os esperados podendo inclusive provocar paradas inesperadas do programa