Conceitos de Linguagens de Programação

Aula 01 – Introdução

Edirlei Soares de Lima
<edirlei@iprj.uerj.br>

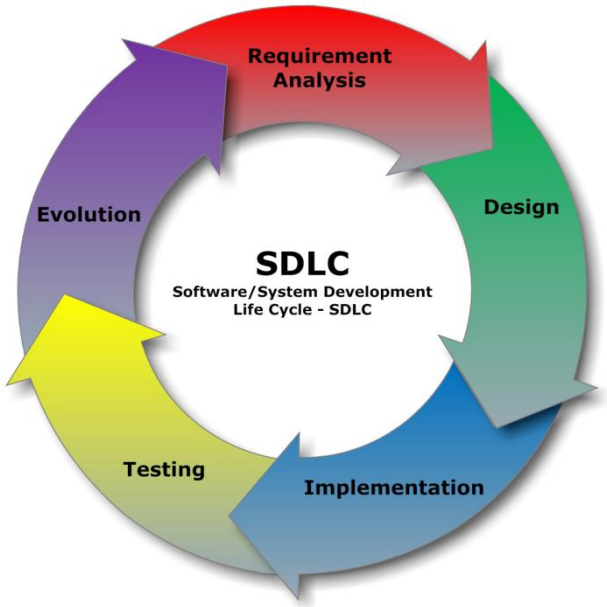
O que é uma Linguagem de Programação?

- Na programação de computadores, uma linguagem de programação serve como **meio de comunicação** entre o indivíduo que deseja resolver um determinado problema e o computador.

- A linguagem de programação deve fazer a ligação entre o **pensamento humano** (muitas vezes de natureza não estruturada) e a **precisão requerida para o processamento** pelo computador.

O que é uma Linguagem de Programação?

- Uma linguagem de programação auxilia o programador no processo de desenvolvimento de software:
  - Projeto;
  - Implementação;
  - Teste;
  - Verificação;
  - Manutenção do software;



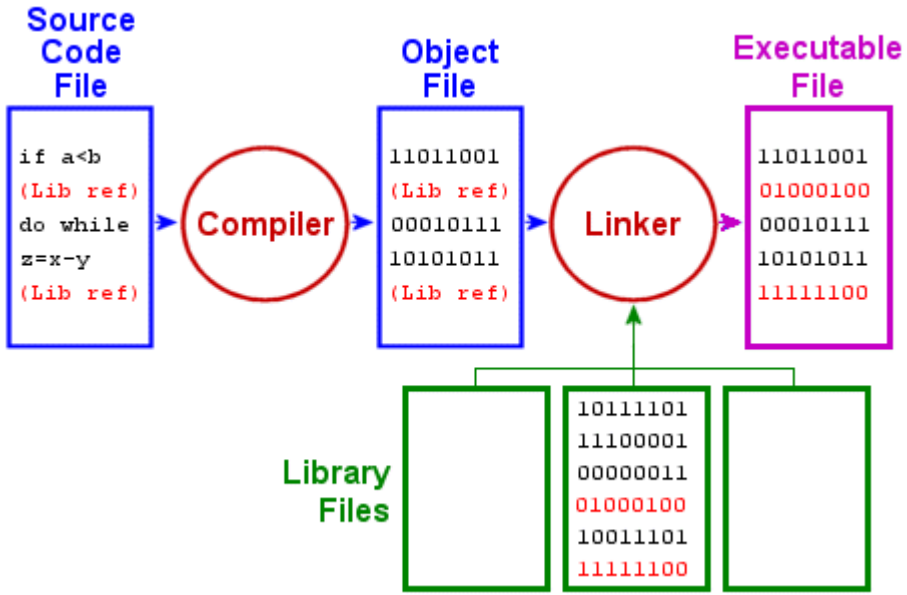
O que é uma Linguagem de Programação?

- Uma linguagem de programação é uma **linguagem** destinada para ser usada por uma pessoa para **expressar um processo** através do qual um computador possa resolver um problema.

- Os **modelos/paradigmas** de linguagens de programação correspondem a diferentes **pontos de vista** dos quais processos podem ser expressados.
  - Exemplos: Imperativo, orientado a objeto, funcional, lógico

O que é uma Linguagem de Programação?

- Para que se tornem operacionais, os programas escritos em **linguagens de alto nível** devem ser traduzidos para **linguagem de máquina**.

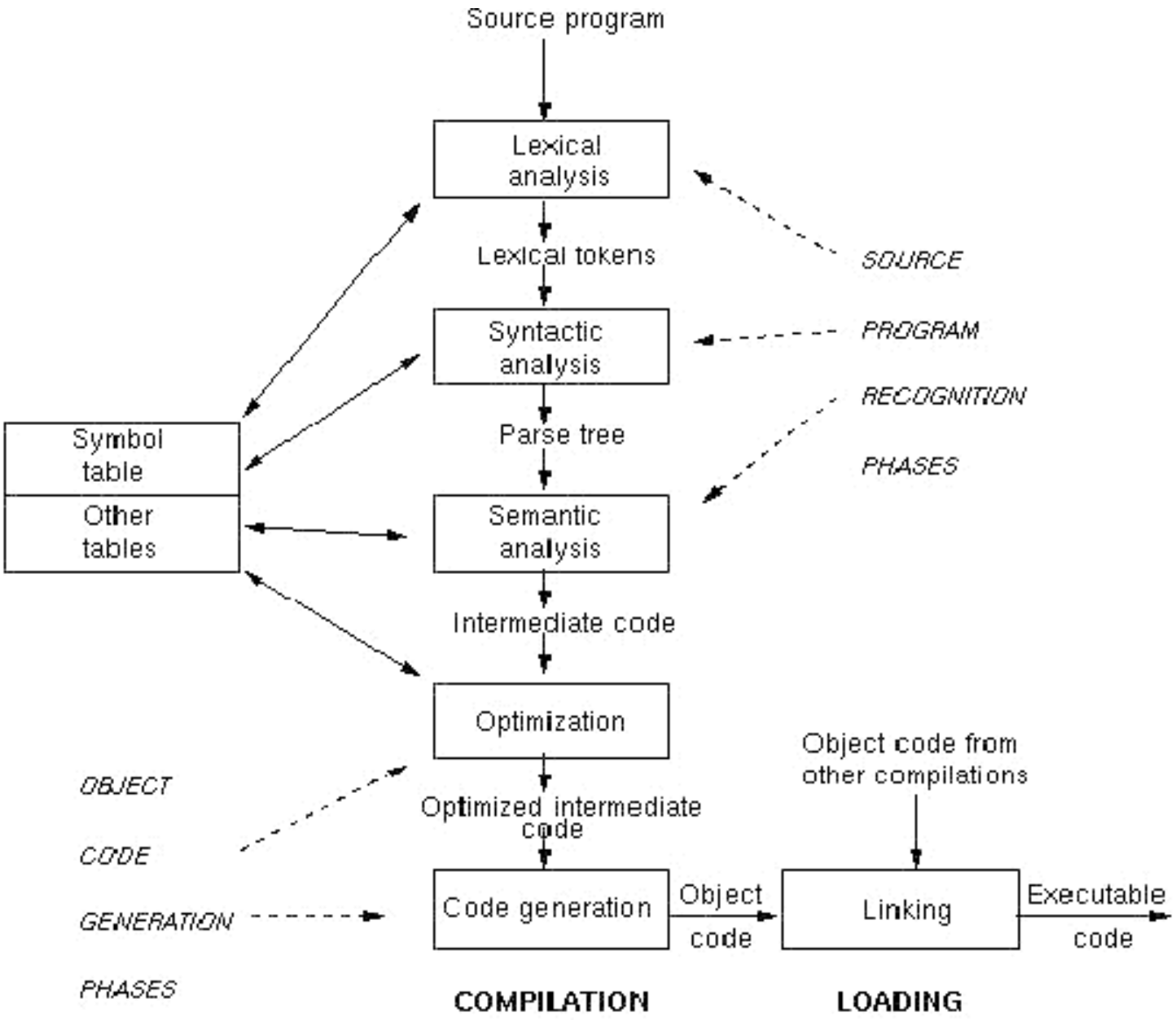


O que é uma Linguagem de Programação?

- A conversão de um código em **linguagem alto nível para linguagem de máquina** é realizada através de sistemas especializados:

Compiladores ou **Interpretadores**

- Esses sistemas recebem como entrada uma representação textual da solução de um problema (expresso em uma **linguagem fonte**) e produzem uma representação do mesmo algoritmo expresso em uma **linguagem de máquina**.


Source program
Lexical analysis
Lexical tokens
Syntactic analysis
Parse tree
Semantic analysis
Intermediate code
Optimization
Optimized intermediate code
Code generation
Object code
Linking
Executable code
Object code from other compilations
Symbol table
Other tables
SOURCE PROGRAM RECOGNITION PHASES
OBJECT CODE GENERATION PHASES
COMPILATION
LOADING

Por que estudar Linguagens de Programação?

- **Aumentar a capacidade de expressar ideias:**
  - Conhecimento amplo dos **recursos de linguagem** reduz as limitações no desenvolvimento de software;
  - A melhor compreensão das funções e implementação das estruturas de uma linguagem de programação nos leva a usar a linguagem de modo a **extrair o máximo** de sua funcionalidade e eficiência;
  - Recursos ou facilidades podem ser simulados.

Por que estudar Linguagens de Programação?

- **Maior conhecimento para escolha de linguagens apropriadas:**
  - Algumas linguagens são mais apropriadas para resolver determinados problemas;
  - Escolher a melhor linguagem para um problema específico devido ao conhecimento de novos recursos é difícil para:
    - Programadores antigos;
    - Desenvolvedores sem educação formal;

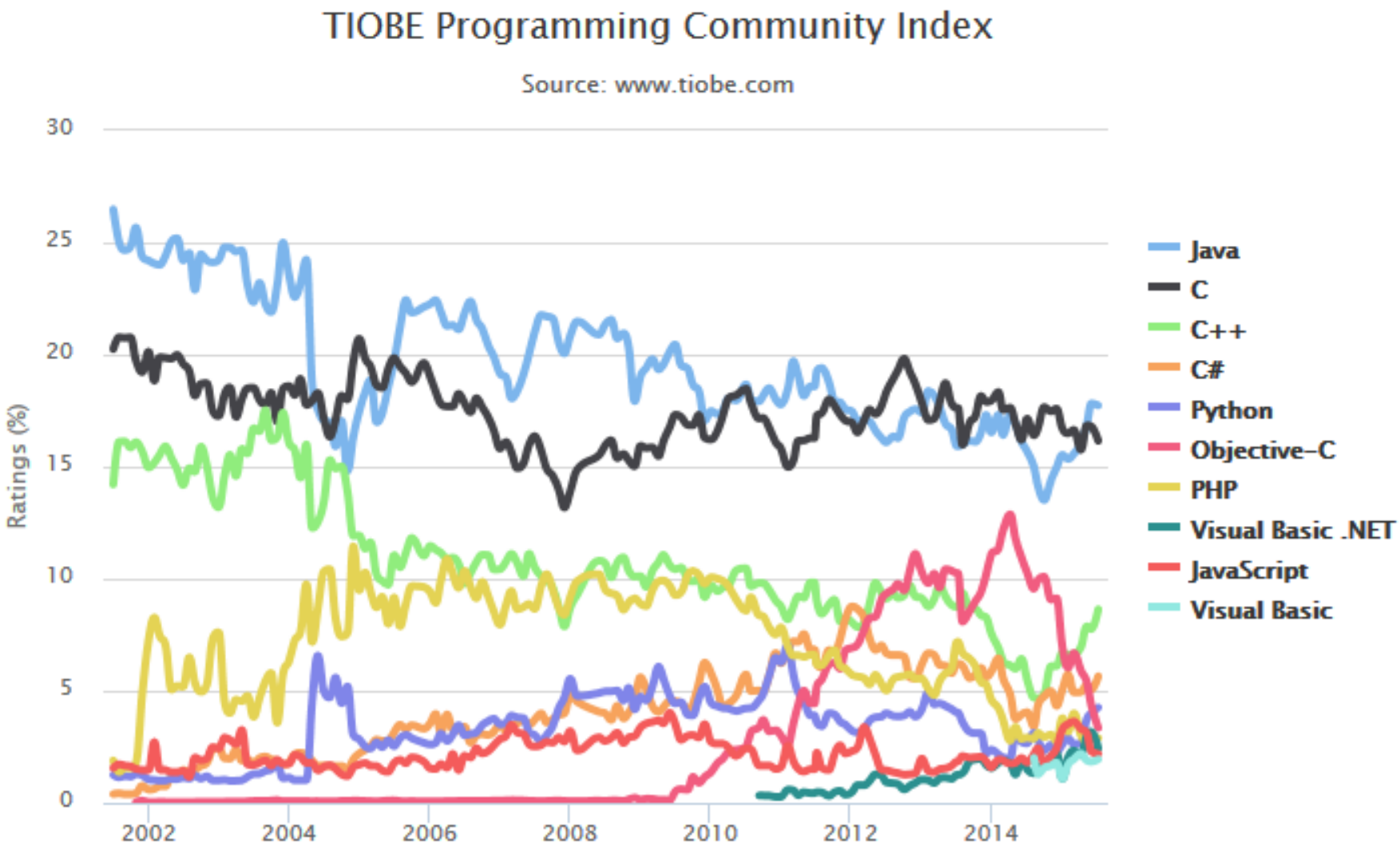
Por que estudar Linguagens de Programação?

- **Entender melhor a importância da implementação:**
  - Leva a um entendimento do **porquê** das linguagens serem projetadas de determinada maneira;
  - Melhora as escolhas que podemos fazer entre as linguagens de programação e as consequências das opções;
  - Nos permite desenvolver programas mais eficientes;

Por que estudar Linguagens de Programação?

- **Maior capacidade para aprender novas linguagens:**
  - Na computação, o aprendizado contínuo é fundamental;
  - Compreender os conceitos gerais das linguagens torna mais fácil entender como eles são incorporados na linguagem que está sendo aprendida;
  - TIOBE Index for July 2015: http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html

Jul 2015	Jul 2014	Change	Programming Language	Ratings	Change
1	2	↑	Java	17.728%	+2.04%
2	1	↓	C	16.147%	-1.00%
3	4	↑	C++	8.641%	+3.12%
4	6	↑	C#	5.652%	+1.60%
5	8	↑	Python	4.257%	+1.60%
6	3	↓	Objective-C	3.344%	-6.95%
7	7		PHP	2.893%	-0.02%
8	12	⇈	Visual Basic .NET	2.423%	+0.93%
9	9		JavaScript	2.194%	+0.39%
10	-	⇈	Visual Basic	1.946%	+1.95%
11	11		Perl	1.812%	+0.18%
12	20	⇈	Assembly language	1.535%	+0.76%
13	17	⇈	Delphi/Object Pascal	1.480%	+0.45%
14	33	⇈	ABAP	1.389%	+1.02%
15	14	↓	Ruby	1.378%	+0.31%


TIOBE Programming Community Index
Source: www.tiobe.com
Ratings (%)
30
25
20
15
10
5
0
2002
2004
2006
2008
2010
2012
2014
Java
C
C++
C#
Python
Objective-C
PHP
Visual Basic .NET
JavaScript
Visual Basic

Por que estudar Linguagens de Programação?

- **Avanço global da computação:**
  - Nem sempre as linguagens mais populares são melhores, por quê?
    - Imposição!
  - Por que existem várias linguagens de programação?
    - Resolução específica de problemas.

Domínios de Programação

- Computadores têm sido aplicados a uma infinidade de áreas...

Domínios de Programação Aplicações Cientificas

- Os primeiros computadores que surgiram na década de 40 foram projetados e utilizados para **aplicações cientificas**.

- Nesta categoria se enquadram todos os problemas que necessitam um grande volume de processamento, com operações geralmente feitas em ponto flutuante, e com poucas exigências de entrada e saída.
  - Uma da preocupações primarias neste tipo de aplicação é a **eficiência**.

- As aplicações científicas incentivaram a criação de algumas linguagens de alto nível, como por exemplo o **Fortran**.

Fortran – Exemplo

```
program teste
  real a, b, s
  read *, a, b
  s = a + b
  print *, a, ' + ' , b
  print *, ' = ' , s
end
```

Compilador Online: http://www.tutorialspoint.com/compile_fortran_online.php

Domínios de Programação Aplicações Comerciais

- O desenvolvimento de aplicações comerciais teve inicio na década de 50.

- A primeira linguagem bem sucedida para o desenvolvimento de aplicações comerciais foi o **COBOL** (em 1960).

- As linguagens de programação comerciais se caracterizam pela facilidade de elaborar relatórios e armazenar números decimais e dados de caracteres.

COBOL– Exemplo

```
IDENTIFICATION DIVISION.
PROGRAM-ID. Teste.
DATA DIVISION.
WORKING-STORAGE SECTION.
01  Num1            PIC 9.
01  Num2            PIC 9.
01  Result          PIC 99.
01  Operador        PIC X.

PROCEDURE DIVISION.
Calcula.
    PERFORM 3 TIMES
       DISPLAY "Numero 1: "
       ACCEPT Num1
       DISPLAY "Numero 2: "
       ACCEPT Num2
       DISPLAY "Operador (+ ou *): "
       ACCEPT Operador
       IF Operador = "+" THEN
          ADD Num1, Num2 GIVING Result
       END-IF
       IF Operador = "*" THEN
          MULTIPLY Num1 BY Num2 GIVING Result
       END-IF
       DISPLAY "Result is = ", Result
    END-PERFORM.
    STOP RUN.
```

http://www.tutorialspoint.com/compile_cobol_online.php

Domínios de Programação Inteligência Artificial

- O desenvolvimento de aplicações para inteligência artificial teve inicio no final da década de 50.

- Essas aplicações caracterizam-se pelo uso de computações simbólicas em vez de numéricas (são manipulados nomes e não números);

- A primeira linguagem desenvolvida para IA foi a funcional **LISP** (1959).

- No início dos anos 70 surge a programação lógica: **Prolog**.

LISP – Exemplo

```
(defun fatorial (num)
   (cond ((zerop num) 1)
      (t (* num (fatorial (- num 1))))
   )
)
(setq n 6)
(format t "Fatorial ~d = ~d" n (fatorial n))
```

Compilador Online: http://www.tutorialspoint.com/execute_lisp_online.php

Prolog – Exemplo

```
pai(fred, marcos).
pai(ricardo, pedro).
pai(pedro, paulo).
avo(X,Y) :- pai(X, Z), pai(Z, Y).
```

```
?- avo(X, paulo).
```

Compilador Online: http://www.tutorialspoint.com/execute_prolog_online.php

Domínios de Programação
Software Básico

- O software básico (sistema operacional) deve possuir eficiência na execução por propiciar suporte a execução de outros aplicativos.

- As linguagens de programação para este tipo de sistema devem oferecer execução rápida e ter recursos de baixo nível que permitam ao software fazer interface com os dispositivos externos.

- O sistema operacional UNIX foi desenvolvido quase inteiramente em C (tornando-o fácil de portar para diferentes máquinas).

Critérios de Avaliação de Linguagens

- Um dos critérios mais importantes para julgar uma linguagem de programação é a facilidade com que os programas são **lidos e entendidos**.

- Antes dos anos 70: linguagens foram criadas pensado em termos de escrita de código.
  - Eficiência e legibilidade da máquina;
  - As linguagens foram projetadas mais do ponto de vista do computador do que do usuário.

- Na década de 70 foi desenvolvido o conceito de ciclo de vida de software: **manutenção**.

Critérios de Avaliação: Legibilidade

- A facilidade de manutenção é determinada em grande parte, pela **legibilidade** dos programas, dessa forma ela se tornou uma medida importante da qualidade dos programas e das linguagens.

- A legibilidade deve ser considerada no contexto do **domínio do problema**.
  - Um programa escrito em uma linguagem não apropriada para o domínio do problema se mostra antinatural, "enrolado" e difícil de ser lido.

Critérios de Avaliação: Legibilidade

1. **Simplicidade geral:**
   - A simplicidade geral de uma linguagem de programação afeta fortemente sua legibilidade;
   - Uma linguagem com um **grande número de componentes básicos** é mais difícil de ser manipulada do que uma com poucos desses componentes.
     - Os programadores que precisam usar uma linguagem grande tendem a aprender um subconjunto dela e ignorar seus outros recursos.
     - Isso pode ser um problema quando o leitor do programa aprende um conjunto diferente de recursos daquele que o autor aplicou em seu programa.

Critérios de Avaliação: Legibilidade

1. **Simplicidade geral:**
   - Uma segunda característica que complica a legibilidade é a multiplicidade de recursos (mais que uma maneira de realizar uma operação particular);
   - Exemplo em C:

```
cont = cont + 1;
cont += 1;
cont++;
++cont;
```

Mesmo significado quando usadas em expressões separadas!

Critérios de Avaliação: Legibilidade

1. **Simplicidade geral:**
   - Um terceiro problema é a **sobrecarga de operadores**, na qual um único símbolo tem mais que um significado.
   - Apesar de ser um recurso útil, pode ser prejudicial a legibilidade se for permitido aos usuários criar suas próprias sobrecargas.
     - Exemplo: sobrecarregar o + para adicionar inteiros, reais, concatenar strings, somar vetores...

Critérios de Avaliação: Legibilidade

1. **Simplicidade geral:**
   - A simplicidade de linguagens, no entanto pode ser levada ao extremo, por exemplo a forma e o significado da maioria das **instruções em Assembly são modelos de simplicidade**, entretanto torna os programas em Assembly menos legíveis.

   - Falta instruções de controle mais complexas, o que torna necessário o uso de mais comandos para expressar problemas do que os necessário em linguagens de alto nível.

Assembly – Exemplo

```
section .text
   global _start
_start:
   mov  eax,'3'
   sub  eax,'0'
   mov  ebx,'4'
   sub  ebx,'0'
   add  eax,ebx
   add  eax,'0'
   mov  [sum],eax
   mov  ecx,msg
   mov  edx,len
   mov  ebx,1
   mov  eax,4
   int  0x80
   mov  ecx,sum
   mov  edx,1
   mov  ebx,1
   mov  eax,4
   int  0x80
   mov  eax,1
   int  0x80
section .data
   msg db "Resultado:",0xA,0xD
   len equ $ - msg
   segment .bss
   sum resb 1
```

http://www.tutorialspoint.com/compile_assembly_online.php

Critérios de Avaliação: Legibilidade

2. **Ortogonalidade:**
   - A ortogonalidade diz respeito a **possibilidade de combinar** entre si, sem restrições, os **componentes básicos** da linguagem para construir estruturas de controle e dados.
   - **Exemplo:** permitir combinações de estruturas de dados, como arrays de estruturas;
   - **Contra exemplo:** não permitir que um array seja usado como parâmetro para uma função;

Critérios de Avaliação: Legibilidade

2. **Ortogonalidade:**
   - A linguagem C possui dois tipos de dados estruturados, arrays e registros (struct), sendo que:
     - Registros podem ser retornados de funções, arrays não.
     - Parâmetros são passados por valor, a menos que sejam arrays – que obrigatoriamente são passados por referência.

Critérios de Avaliação: Legibilidade

3. **Instruções de controle:**
   - A revolução da programação estruturada da década de 70 foi, em parte, uma reação à má legibilidade causada pelas limitadas instruções de controle das linguagens das décadas de 50 e 60.
   - Reconheceu-se que o uso indiscriminado de instruções `goto` reduz criticamente a legibilidade de programas.
   - Em certas linguagens, entretanto, instruções `goto` que se ramificam para cima, às vezes, são necessárias;
     - Exemplo: `goto` é usado na construção de laços while em FORTRAN.

Critérios de Avaliação: Legibilidade

3. **Instruções de controle:**
   - Restringir instruções `goto` das seguintes maneiras pode tornar os programas mais legíveis:
     - As instruções goto devem preceder seus alvos, exceto quando usadas para formar laços;
     - Os seus alvos nunca devem estar tão distantes;
     - O número de usos deve ser limitado;
   - A partir do final da década de 60, as linguagens projetadas passaram a ter instruções de controle suficientes e portanto a necessidade da instrução `goto` foi quase eliminada.

Critérios de Avaliação: Legibilidade

4. **Tipos de dados e estruturas:**
   - A presença de facilidades adequadas para definir tipos de dados e estruturas de dados em uma linguagem é outro auxílio significativo para a legibilidade.

   - **Exemplo:** supondo que um tipo numérico seja usado para um sinalizador porque não há nenhum tipo booleano na linguagem:
     - `terminou = 1`, não é tão claro como `terminou = true`

   - **Outro avanço:** tipos enumerados.

Critérios de Avaliação: Legibilidade

5. **Considerações sobre sintaxe:**
   - A sintaxe ou a forma dos elementos de uma linguagem tem um efeito significativo sobre a legibilidade dos programas.

   - Exemplos de opções de projeto sintático que afetam a legibilidade:
     - **Formas identificadoras:** restringir os identificadores a tamanhos muitos pequenos prejudica a legibilidade, impedindo que variáveis sejam nomeadas com nomes conotativos. Exemplos:
       - FORTRAN 77: máximo 6 caracteres;
       - BASIC ANSI: uma letra ou uma letra e um número;

Critérios de Avaliação: Legibilidade

5. **Considerações sobre sintaxe:**
   - Exemplos de opções de projeto sintático que afetam a legibilidade:
     - **Palavras especiais:** a aparência de um programa e sua consequente legibilidade são fortemente influenciadas pelas formas das palavras especiais de uma linguagem (begin, end, for...);
     - **Exemplos:** Pascal exige pares de begin/end para formar grupos em todas as construções de controle, a linguagem C usa chaves;
       - Ambas as linguagens sofrem porque os grupos de instruções são sempre encerrados da mesma maneira, o que torna difícil determinar qual grupo está sendo finalizado quando um end ou } aparece;
       - O FORTAN 90 e o ADA tornam isso mais claro, usando uma sintaxe de fechamento distinta para cada tipo de grupo de instrução. (if...end if / loop...end loop);

Critérios de Avaliação: Capacidade de Escrita

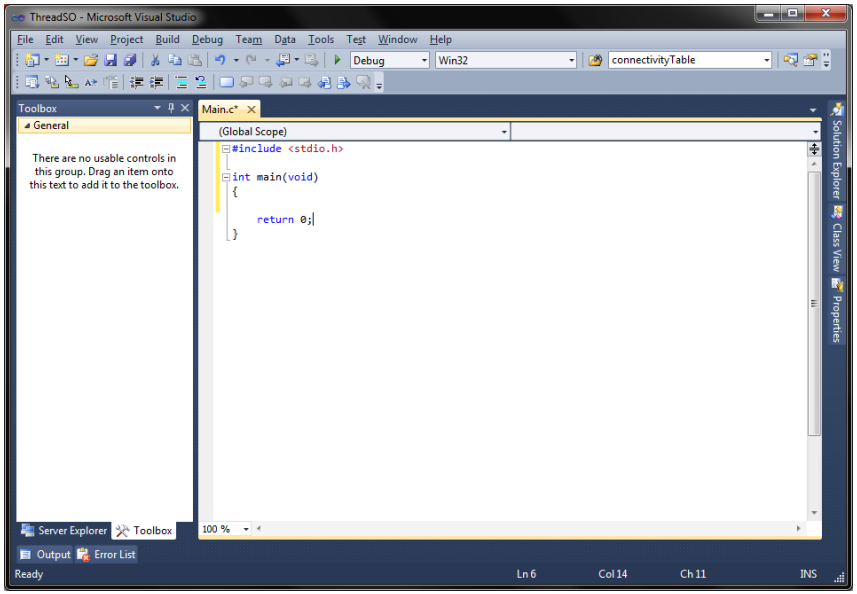
- A **capacidade de escrita** é a medida da facilidade em que uma linguagem pode ser usada para criar programas para um domínio de problema escolhido;

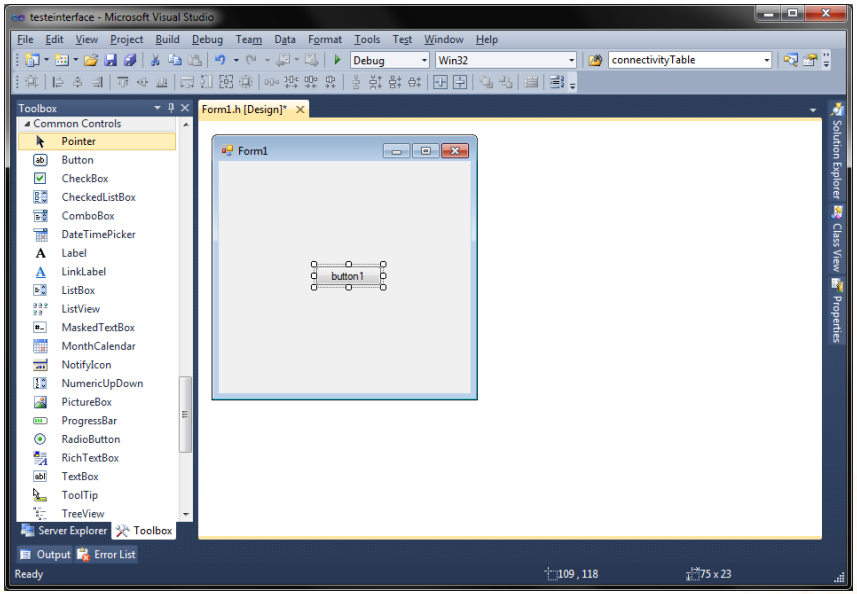
- A maioria das características da linguagem que afetam a legibilidade também afetam a capacidade de escrita;

- Deve ser considerada no contexto do domínio de problema-alvo da linguagem.

Critérios de Avaliação: Capacidade de Escrita

- **Exemplo:** Construção de uma interface gráfica





C **versus** **C++**

Critérios de Avaliação: Capacidade de Escrita

1. **Simplicidade e Ortogonalidade:**
   - Se uma linguagem de programação tem um grande número de construções, alguns programadores não estarão familiarizados com todas;
   - Pode acarretar o uso incorreto de alguns recursos e uma utilização escassa de outros que podem ser mais elegantes ou eficientes do que os usados;
   - Podem ocorrer usos de recursos desconhecidos com resultados não esperados.

Critérios de Avaliação: Capacidade de Escrita

2. **Suporte para abstração:**

   - **Abstração:** capacidade de definir e, depois usar estruturas ou operações complicadas de uma maneira que permita ignorar muito dos detalhes.
     - Exemplo: uso de funções provenientes de bibliotecas;

   - **Tipos de Abstração:**
     - Abstração de Processo: algoritmos em geral;
     - Abstração de Dados: tipos de dados e estruturas de dados.

Critérios de Avaliação: Capacidade de Escrita

3. **Expressividade:**
   - Formas convenientes de especificar computações, onde uma expressão representa muitas computações.

   - Exemplos:
     - `i++`, ao invés de `i = i + 1;`
     - `for` ao invés do `while;`
     - `cin` do C++ ao invés de `nextInt` do Java

```
int v;
cin >> v;
```

```
int v;
Scanner entrada;
entrada = new Scanner(System.in);
v = entrada.nextInt();
```

Critérios de Avaliação: Confiabilidade

- Um programa é **confiável** se ele se comportar de acordo com suas especificações sob todas as condições.

- Recursos que afetam a confiabilidade:
  - Verificação de Tipos;
  - Manipulação de Exceções;
  - Apelidos (Aliasing);
  - Legibilidade e Facilidade de Escrita;

Critérios de Avaliação: Confiabilidade

1. **Verificação de Tipos:**
   - Visa testar se existem erros de tipos de dados no programa por meio do compilador ou durante a execução do programa;
   - A verificação de tipos durante a compilação é a mais indicada. Quanto antes for detectado o erro, menos caro é fazer todos os reparos necessários.
     - Exemplo: Java
   - A verificação de tipos em C é bastante fraca:

```
int vet[50];
vet[100] = 8.5;
```

Critérios de Avaliação: Confiabilidade

2. **Manipulação de Exceções:**
   - Capacidade de um programa de interceptar erros em tempo de execução, pôr em prática medidas corretivas e, depois, prosseguir.
   - Exemplos em C++ e Java:

```
try
{
   ...
}
catch (Exception &exception)
{
   ...
}
```

```
try
{
   ...
}
catch(Exception e)
{
   ...
}
```

Critérios de Avaliação: Confiabilidade

3. **Apelidos (Aliasing):**
   - Consiste em ter um ou mais métodos, ou nomes, distintos para fazer referência à mesma área de memória.
   - Exemplos em C:

```
char c = 'x';
char *p;
p = &c;
*p = 'z';
```

```
union Data{
   int i;
   float f;
   char  str[20];
};
union Data data;
data.i = 10;
data.f = 220.5;
strcpy(data.str, "Programa");
```

Critérios de Avaliação: Confiabilidade

4. **Legibilidade e Facilidade de Escrita:**

   – Tanto a legibilidade como a facilidade de escrita influenciam a confiabilidade.

   – Quanto mais fácil é escrever um programa, mais probabilidade ele tem de estar correto.

   – Programas de difícil leitura complicam também sua escrita e sua modificação.

Critérios de Avaliação: Custo

- Custo final de uma linguagem de programação é uma função de muitas de suas características.

- **Custo de Treinamento:** cresce em função da simplicidade e da ortogonalidade da linguagem e da experiência dos programadores;

- **Custo da Escrita:** Os esforços originais para projetar e implementar linguagens de alto nível foram motivados pelos desejos de diminuir os custos para criar software;

Critérios de Avaliação: Custo

- **Custo do Sistema de Implementação:** uma linguagem de programação cujo sistema de implementação seja caro, ou rode somente em hardware caro, terá muito menos chance de tornar-se popular;

- **Custo do Projeto da Linguagem: s**e uma linguagem de programação exigir muitas verificações de tipos durante a execução, proibirá a execução rápida do código;

- **Custo de Compilação:** problema amenizado com o surgimento de compiladores otimizados e de processadores mais rápidos.

Critérios de Avaliação: Custo

- **Custo da Má confiabilidade:** falhas podem ocasionar insucesso do software e ações judiciais;

- **Custo de Manutenção:** depende principalmente da legibilidade. O custo de manutenção pode atingir de duas a quatro vezes o custo de desenvolvimento;

Como Escolher uma Linguagem?

- **Implementação:**
  - Disponibilidade quanto à plataforma;
  - Eficiência: velocidade de execução do programa objeto;
- **Competência:**
  - Experiência do programador;
  - Competência do grupo envolvido;
- **Portabilidade:**
  - Necessidade de executar em várias máquinas diferentes;

Como Escolher uma Linguagem?

- **Sintaxe:**
  - Certos tipos de aplicação acomodam-se melhor em certas sintaxes;
- **Semântica:**
  - Algumas linguagens são mais adequadas para processamento concorrente, outras são mais otimizadas para recursividade;

Como Escolher uma Linguagem?

- **Ambiente de programação:**
  - Ferramentas para desenvolvimento de software diminuem o esforço de programação;
  - Existência de bibliotecas;

- **Modelo de computação**
  - O Paradigma Lógico pode ser mais adequado para algumas aplicações de inteligência artificial, enquanto que o Paradigma Orientado a Objeto é mais adequado para aplicações comerciais;

Trabalho 1

- Em dupla, pesquise sobre a **linguagem de programação** sorteada para a sua dupla. Mostre o histórico da linguagem, características, importância, estrutura, vantagens/desvantagens, versões e classificação (nível, geração e paradigma). Apresente exemplos de código-fonte.

- **A dupla deverá:**
  - Apresentar a pesquisa em aula (15 minutos para cada dupla);
  - Entregar um relatório (em PDF);

- **Data de entrega e apresentação:** 10/09

Leitura Complementar

- Sebesta, Robert W. **Conceitos de Linguagens de Programação**. Editora Bookman, 2011.

- **Capítulo 1: Aspectos Preliminares**