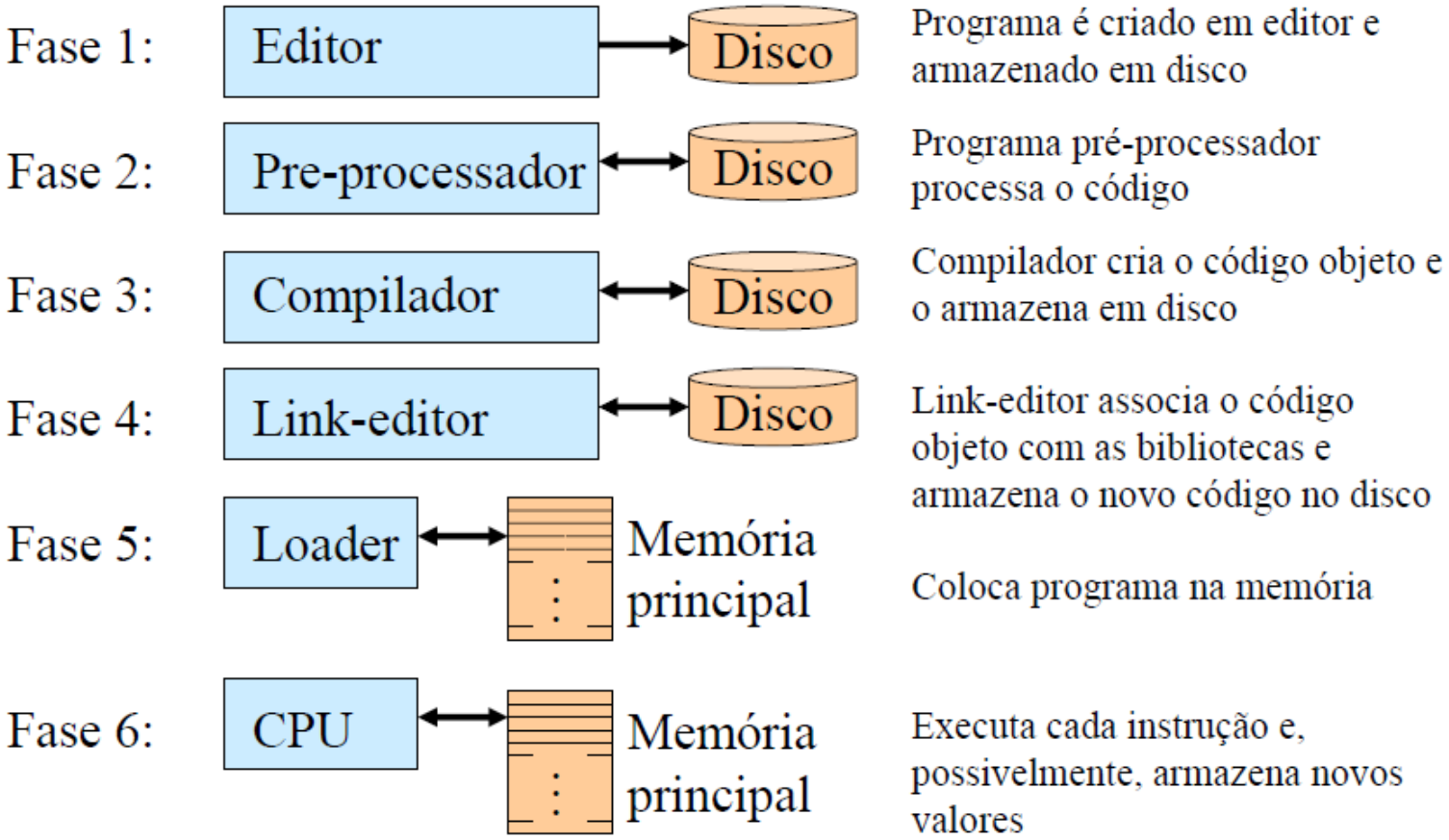
Estrutura de Dados 2023

PROF. FELIPE PELLISON

FELIPE.CARVALHO@BARAODEMAUA.BR

Criação de Programas em C++



Erros de Programação

- **Muitas vezes, os programadores cometem erros ao escrever seus programas**

- **Em geral, um programa pode apresentar 4 categorias de erros**
  - Erros sintáticos ou de compilação
  - Erros de composição
  - Erros de execução
  - Erros semânticos

Erros de Programação

- **Erros sintáticos ou de compilação**

Código não obedece às regras de sintaxe da linguagem – não compila

»Sintaxe de uma linguagem é a sua “gramática”

Erros são indicados através de mensagens de erros resumidas na tela do computador

»Mensagens de erros mais detalhadas podem ser encontradas em manuais da linguagem

»Feitas as correções necessárias, a compilação pode ser reinicializada

Erros de Programação

- **Erros de composição**

Ocorrem na fase de link-edição

Geralmente estão relacionados a erros na chamada de funções em outras bibliotecas

»Exemplo: chamar a função de saída de rand() sem importar a bibliotecas usadas

Erros são indicados através de mensagens de erros resumidas na tela do computador

Erros de Programação

- **Erros de execução**

Código obedece às regras de linguagem

»programa é compilado

Mas ao ser executado, código executa operações não permitidas

»Exemplos

- dividir por zero
- raiz quadrado de número negativo

São mais difíceis de descobrir e interpretar

Erros de Programação

- **Erros semânticos**

Semântica = significado

Erros no projeto lógico do programa

»Forma utilizada para resolver o problema

Provoca erros indesejáveis no programa

»Mensagens indicativas dos erros não são apresentadas

»Sua correção exige mudanças na concepção do programa

São os erros de detecção mais difícil

Operadores

Standard algebraic equality or relational operator	C++ equality or relational operator	Sample C++ condition	Meaning of C++ condition
Relational operators			
>	>	x > y	x is greater than y
<	<	x < y	x is less than y
≥	>=	x >= y	x is greater than or equal to y
≤	<=	x <= y	x is less than or equal to y
Equality operators			
=	==	x == y	x is equal to y
≠	!=	x != y	x is not equal to y

Operadores

Operators				Associativity	Type
*	/	%		left to right	multiplicative
+	-			left to right	additive
<<	>>			left to right	stream insertion/extraction
<	<=	>	>=	left to right	relational
==	!=			left to right	equality
=				right to left	assignment

Condicionais

- **São estruturas de controle condicionais**

Muitas vezes, a avaliação de uma condição define a execução ou não de um comando (execução condicional)

A maneira mais fácil de fazer isso é através do comando *If*

```
if (condicao)
        comando;
```

```
if (condicao)
        comando1 ;
else
        comando2;
```

Comandos podem ser simples ou compostos (bloco)

Condicionais

- **Operadores lógicos podem ser utilizados na equação de condição**

–Exemplos:

```
...
if ( (a>5) && (b==-1) ) {
    ...
```

```
...
if ( (a>1) || !(b==2) ) {
    ...
```

Repetições

- **Permitem a execução de partes de um programa mais de uma vez**

- **São comandos iterativos da linguagem C++**

–Comando *while*

–Comando *do while*

–Comando *for*

Repetições (while)

- **Estrutura de repetição mais simples**
- **Executa um comando repetidamente até uma condição se tornar falsa**

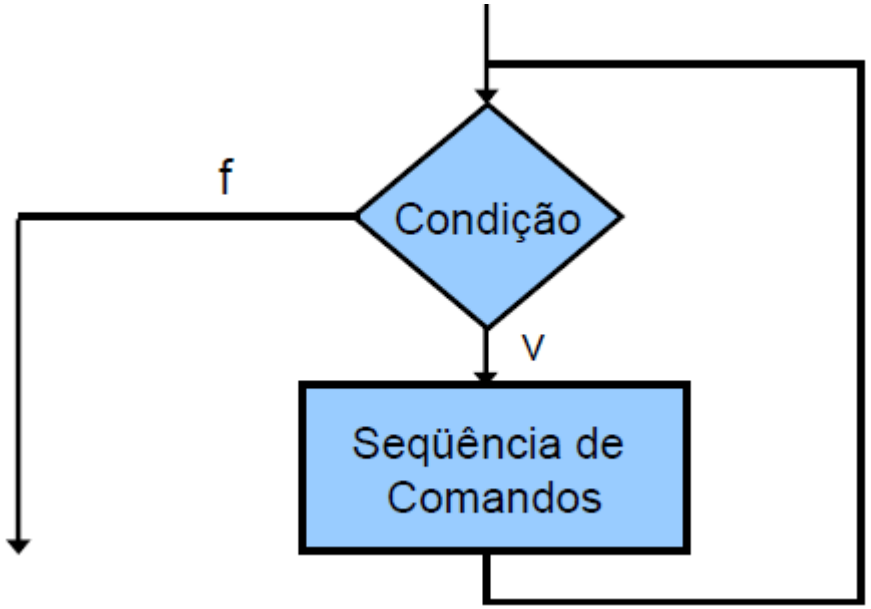
```
while (expressão condicional)
        comando;
```

Comandos podem ser simples ou compostos (bloco)

Repetições (while)

- **Teste condicional é executado antes de cada ciclo do *loop***

–Se o primeiro teste resultar no valor 0 (Falso), o corpo do loop não é executado



Repetições (while)

- **Utilizado onde existe uma condição de teste que possa ser aplicada antes da execução do corpo**

- **Vários problemas de programação não se encaixam na estrutura do comando *while***

  – As vezes o teste seria mais natural em algum lugar no meio do loop

- Ex.: leitura de dados do usuário até o recebimento de um valor especial (sentinela)

Repetições (while)

- **Utilizado onde existe uma condição de teste que possa ser aplicada antes da execução do corpo**
- **Vários problemas de programação não se encaixam na estrutura do comando *while***

–As vezes o teste seria mais natural em algum lugar no meio do loop

- Ex.: leitura de dados do usuário até o recebimento de um valor especial (sentinela)
- ***Loop* baseado em sentinela**
- Ler um valor
- Se o valor é igual ao sentinela, sair do *loop*
- Senão, executar o processamento requerido por este valor

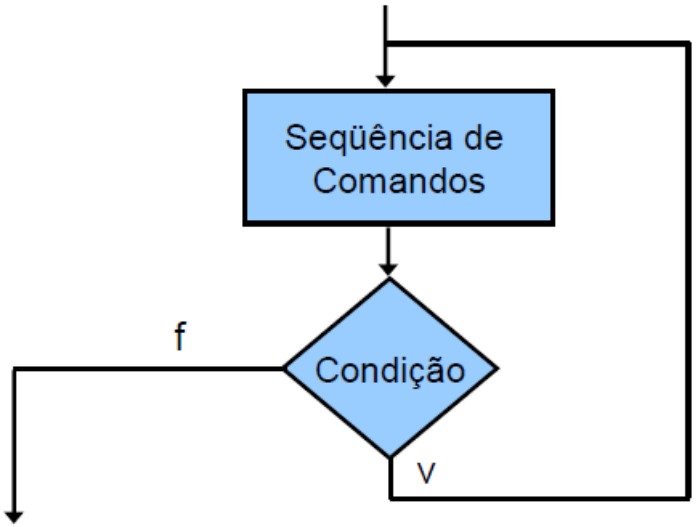
Repetições (while)

- **Executa um comando repetidamente até uma condição se tornar falsa**

do {

comandos;

} while (expressao condicional);



- **Semelhante ao comando *while***

–Exceto que a expressão é avaliada no final

Repetições (for)

•O comando *for* é um comando de repetição determinado pelas expressões:

–inicial

–teste

–passo

```
for (inicial; teste; passo) {
        comandos;
}
```

Repetições (for)

- **Inicial**

–Expressão que indica como o *loop* do comando *for* deve ser inicializado

–Executado uma única vez no início do *loop*

–Define o valor inicial da variável de indexação (contador)

- Ex.: for (i = 0; .... for (i = -7; ...

Repetições (for)

- **Teste**

–Expressão que indica quando o *loop* do comando *for* deve parar

–Funciona como a condição de teste do *while*

–Expressão é avaliada no topo de cada interação do *loop*

- Compara valor do contador com um valor final
  - –Quando o resultado da avaliação é *FALSO*, o *loop* termina
  - –Enquanto teste é VERDADEIRO, o *loop* continua
- Ex.: for (i = 0; (i < n); i++){

Repetições (for)

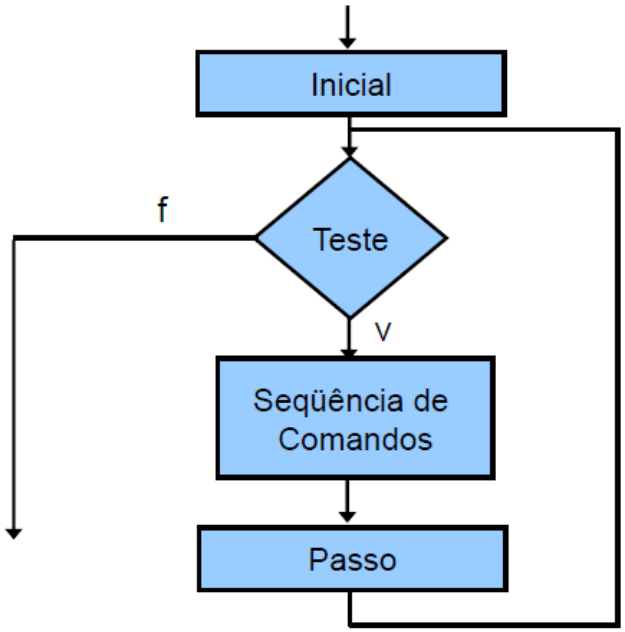
- **Passo**

–Expressão chamada a cada interação do *loop* para atualizar o valor do contador

–Indica quanto muda o valor da variável de indexação de um ciclo para outro

–Especificações de passo mais comuns

- *index++, index--*



Repetições (for)

- **Expressões inicial, teste e passo são opcionais**

–Os separadores (;) devem aparecer

–Falta de valor inicial -> não é feita nenhuma inicialização do indexador

–Falta de condição de teste -> ela é assumida como sempre VERDADEIRA

–Falta de passo -> indexador não é alterado entre ciclos do loop

- Mas pode ser alterado dentro do ciclo