UNIP

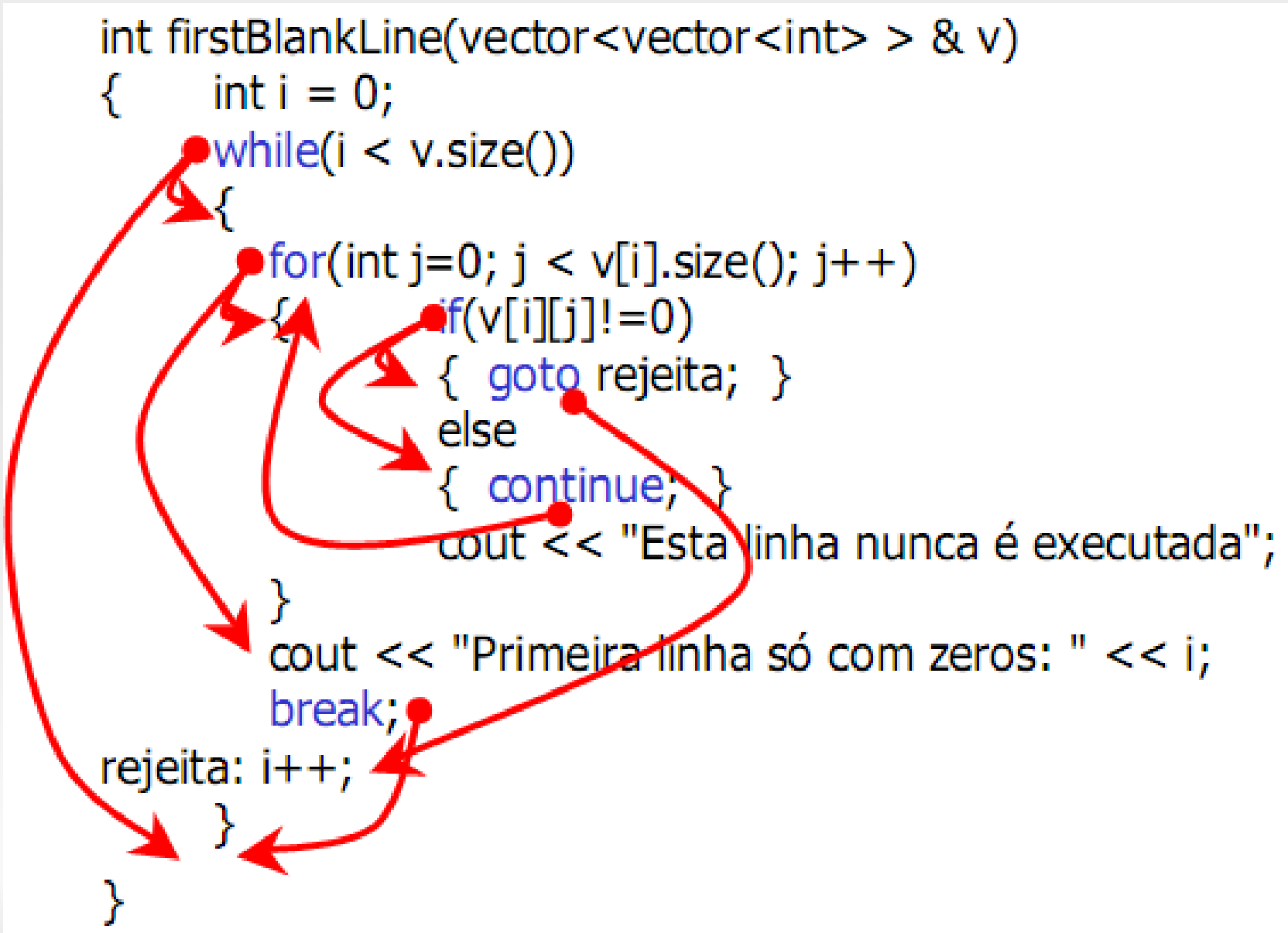
UNIVERSIDADE PAULISTA

Paradigmas de Linguagens

Aula 4: Estruturas de Controle

Professora Sheila Cáceres

Motivação



Não é um bom exemplo

Introdução

- Além das instruções de atribuição, são necessários pelo menos dois mecanismos linguísticos adicionais:
  - Selecionar entre caminhos
  - Execução repetida de instruções

  } **Instruções de Controle**
- Sem elas só haveria uma maneira do programa ser executado: de cima para baixo, comando por comando
- As estruturas de controle são fundamentais para qualquer linguagem de programação

Fluxo de controle ao nível de instrução

- ***Estrutura de controle*** *é* uma instrução de controle e a sua coleção de comandos cuja execução ela controla

- Níveis de fluxo de controle:
  - Entre unidades de programa
  - **Entre instruções do programa**
  - Dentro de expressões
    - Regra de Associatividade
    - Regra de Precedência

Instruções Compostas

- Método para abstrair uma coleção de instruções como uma única instrução
- Foi introduzido no ALGOL 60:

```
begin
    comando 1
    ...
    comando n
end
```

- Um bloco é uma Instrução Composta que pode definir variáveis locais.

Instruções de Seleção

- Permite escolher entre dois ou mais caminhos de execução em um programa.
- Duas categorias:
  - Seleção bidirecional
  - Seleção n-dimensional ou múltipla

Instruções de Seleção Bidirecional

- Permitem escolher entre dois ou mais caminhos de execução no programa.
- Forma geral:
  - if (expressão booleana)
    - instrução
  - else
    - instrução
- Alguns aspectos para serem considerados:
  - Tipo de expressão que controla o seletor (quem seleciona caminhos)
  - Quantas instruções podem ser selecionadas (permitir sequencias de instruções foi um grande passo na evolução das instruções – instruções únicas acarretavam dependência de gotos)
  - Possibilidade de adicionar seletores aninhados

Instrução de Seleção Bidirecional

- Exemplo em FORTRAN IV (unidirecional):
  IF (expressão booleana) instrução
- Problema:
  - seleciona somente uma único instrução.
  - Para selecionar mais instruções, deve ser utilizado um "goto".
- Exemplo: queremos inicializar as variáveis I e J somente se FLAG =1

```
IF (.NOT. condição) GOTO 20
    I  = 1
    J = 2
20 CONTINUE
```

lógica negativa (prejudicial ->legibilidade)

Instrução de Seleção Bidirecional

- ALGOL 60 introduziu uma instrução de seleção **unidirecional** para executar uma única instrução ou uma instrução composta (foi adotada por linguagens posteriores)

```
if (expressão booleana) then
begin
instrução1
…..
instrução n
end
```

- ALGOL 60 introduziu o primeiro seletor **bidireccional.**

```
if (expressão booleana) then
   instrução
else
    instrução
```

  - Onde instrução pode ser simples ou composta.

Instrução de Seleção Bidirecional

Nas linguagens C,C++, C#, Java usamos:

- A expressão do lado de if é avaliada. Se ela for verdadeira (diferente de 0), o comando ou bloco que forma o corpo do if é executado; Caso contrário, o comando ou bloco que é o corpo do else (se existir) é executado.
- Apenas o código associado ao **if** ou o código associado ao **else** será executado, nunca ambos;
- A forma geral da sentença if é:

```
if(expressão)
    { comando; }
else
    { comando; }
```

- A cláusula else é opcional.

Aninhando Seletores

- **Exemplo em Java**

```
if (sum == 0)
    if (count == 0)
        result = 0;
else result = 1;
```

- O **else** pertence a qual dos ifs?
- Java possui uma regra semântica estática
  - O else solto é associado ao if mais próximo

Aninhando Seletores

- Exemplo em Pascal (aninhamento direto de seletores):

```
if ... then
   if ... then
      ...
   else ...
```

- Qual then associa-se com else?
- Regra do Pascal:
  - else associa-se com o then mais próximo.

Aninhando Seletores

- Para forçar uma semântica alternativa, uma forma sintática diferente é necessária, na qual o if-else é colocado em uma instrução composta usando chaves { }.
- Exemplo original em java

```
if (sum == 0)
   if (count == 0)
      result = 0;
else result = 1;
```

- Exemplo usando instruções compostas { }

Else do segundo if

```
if (sum == 0){
     if (count == 0)
            result = 0;
     else result = 1;
 }
```

Else do primeiro if

```
if (sum == 0){
   if (count == 0)
      result = 0;
}
else result = 1;
```

- A solução acima é usada em C, C++ e C#, Perl requer que todas as cláusulas then e else sejam compostas

- Ex. em ALGOL 60:
  - Não permite aninhamento direto
  - É preciso ser colocado em uma instrução composta

Aninhado ao 2º if

```
if ... then
   begin
      if ... then
         ...
      else …
   end
```

Aninhado ao 1º if

```
if ... then
   begin
      if ... then
         …
   end
else …
```

Construções de Seleção Múltipla

- Permite a seleção de uma instrução, dentre qualquer número de instruções ou de grupos de instruções (generalizañ{ao de um seletor).
- Suas origens pertencem ao FORTRAN.

- Questões de Projeto:
  - Qual é a forma e o tipo da expressão que controla que controla a seleção?
  - Que instruções são selecionadas (simples, compostas, seqüências de instruções)?
  - O fluxo de execução através da estrutura limita-se a incluir apenas um único segmento selecionável?
  - O que é acontece quando o valor da expressão não está representado?

Seletores Múltiplos Antigos

FORTRAN - IF aritmético (**seletor tridirecional**)

- Escolhe entre três caminhos de desvio baseando-se no valor de uma expressão aritmética
- Forma : O desvio baseia-se no resultado da expressão aritmética considerando que ela pode ser > 0 , = 0, < 0 (nessa ordem)

  IF (expressão aritmética) N1, N2, N3: .

  - Exemplo – Determinar se o valor numérico é

```
IF (expressão) 10,  20,  30
10 ...
   ...
GO TO 40
20 ...
   ...
GO TO 40
30 ...
   ...
40 ...
```

  - Aspectos negativos

    Segmentos requerem GOTOs. Os segmentos selecionáveis podem estar em qualquer lugar no código

Seletores Múltiplos Modernos (switch)

- O switch é um comando de seleção múltipla
- O switch testa sucessivamente o valor de uma expressão contra uma lista de constantes inteiras ou de caracteres.
- Existe na linguagem C,C++, C#, java, etc

```
switch(expressão){
case constante1:
    seqüência de comandos
    break;
case constante2:
    seqüência de comandos
    break;
case constante3:
    seqüência de comandos
    break;
...
default:
    seqüência de comandos
}
```

- O valor da expressão é testado, na ordem, contra os valores das constantes especificadas nos comandos case;
- Quando uma coincidência for encontrada, a seqüência de comando associada àquele case será executada até que o comando break ou o fim do comando switch seja alcançado;
- O comando default é executado se se nenhuma coincidência for detectada;
- O default é opcional.

Seletores Múltiplos Modernos (switch)

```c
#include <stdio.h>

int main() {

    int num;

    printf("\n ##########################################");
    printf("\n # exemplo do comando switch #");
    printf("\n ##########################################");

    printf("\n\n Digite um numero ");
    scanf("%d", &num);


    switch( num )
    {
        case 9:
            printf("\n O numero eh igual a 9");
            break;

        case 10:
            printf("\n O numero eh igual a 10");
            break;

        default:
            printf("\n O numero nao eh 9 nem 10");

    }


}
```

Seletores Múltiplos usando if

- Seletores múltiplos podem ser construídos como extensões diretas de seletores bidirecionais, usando cláusulas else-if, por exemplo em Ada:

```
if ...
   then ...
elsif ...
   then ...
elseif …
   then …
else ….
end if
```

Instruções Iterativas (repetição)

- Execução de zero, uma ou mais vezes de uma instrução ou bloco de instruções

  Realizada através de:

  - Iteração
  - recursão

Instruções Iterativas

- Tipos de Instruções Iterativas:
  - Laços controlados por contador;
  - Laços controlados logicamente;
  - Laços controlados pelo usuário.

Instruções Iterativas

- Considerações de projeto (para iteração):
  - Como a iteração é controlada?
    - Contador;
    - Expressão lógica (booleana) ou relacional.
  - Onde colocar a condição de controle do laço?
    - No início do laço (laço pré-testado);
    - No fim do laço (laço pós-testado).

Laços controlados por contador

- Uma instrução iterativa de contagem possui uma **variável do laço**, na qual o valor da variável é mantido.
- Essa instrução possui meios de especificar os valores **inicial** e **terminal** da variável, e o **tamanho do passo**
- Instrução for do Pascal
  - **for** variavel := valor-inicial **(to | downto)** valor-final **do**

    instrução

Laços controlados por contador

- Instrução for do C, do C++ e do Java

```
for(inicialização; condição; incremento)
{
    comandos;
}
```

Exemplo

```
for (int cont = 0; cont < comp; cont++) {…}
```

  - Todas as variáveis envolvidas podem ser alteradas no corpo do laço
  - A primeira expressão é avaliada apenas uma vez, mas as outras duas são avaliadas em cada iteração

Comando For

- Inicialização é, geralmente, um comando de atribuição que é usado para colocar um valor na variável de controle do laço;
- A condição é uma expressão relacional que determina quando o laço acaba;
- O incremento define como a variável de controle do laço varia cada vez que o laço é repetido;

Exemplo

```c
#include <stdio.h>

int main() {

    int x;

    for(x=1; x<100; x++)
    {
        printf("%d", x);
    }

    printf("Saimos do laco");


}
```

```
Administrator: C:\Windows\system32\cmd.exe

c:\programasC>exfor.exe
12345678910111213141516171819202122232425262728293031323334353637383940414243444546474849505152535455565758596061626364656667686970717273747576777879808182838485868788899091929394959697989 9Saimos do laco
c:\programasC>
```

Laços controlados por contador

RESUMO:

Linguagem	Tipo Var. de controle do laço	Teste do laço	Valor da Var. no fim do laço	Var. laço pode ser alterada?
FORTRAN I	int	pós	mais recente	não
FORTRAN 77	int, float, double	pós	mais recente	não
Algol 60	int, float	pré	mais recente	não
Pascal	int, enumerado	pré	mais recente	sim
Ada	int,enumerado	pré	eliminado	não
C/C++/Java	n/ tem var.	pré	mais recente	sim

Laços Controlados Logicamente

- O controle da repetição baseia-se em uma expressão booleana e não em um contador

- Exemplos em Pascal:
  - Pascal possui instruções separadas para
    - pré-teste

```
while(condição=true) do
begin
      ...
end
```

    - pós-teste

```
repeat
      ...
until(condição=false)
```

Laços Controlados Logicamente

- C e C++ também possuem

```
while (condição == true)
   corpo do laço;
```

```
do
   corpo do laço;
while (condição == true);
```

Comando while

- A segunda estrutura de repetição em C é o laço while. A sua forma geral é:

```
while(condição)
{
        comando;
}
```

- *comando* é um comando vazio, um comando simples ou um bloco de comandos;
- A *condição* pode ser qualquer expressão, e verdadeiro é qualquer valor não-zero;
- O laço se repete quando a condição for verdadeira. Quando a condição é falsa, o controle do programa passa para a linha após o ódigo do laço

Exemplo while

```c
#include <stdio.h>

int main() {

    int num1, num2;

    char ch = '\0';

    printf("Este programa soma dois numeros inteiros ate que o usuario digite N ou n  no flag ");


    while( ch != 'N' && ch != 'n' )
    {
        printf("\n\nDigite o primeiro numero");
        scanf("%d", &num1);

        printf("\n\nDigite o segundo numero");
        scanf("%d", &num2);

        printf("A soma de %d + %d eh %d", num1, num2, (num1+num2));

        printf("\n\nDeseja realizar uma nova soma? (S/N)");

        ch = getche();


    }

    printf("Numeros pares listados acima ");


}
```

Comando do ... while

- Ao contrário dos laços for e while, que testam a condição do laço no começo, o laço do-while verifica a condição ao final do laço;
- Portanto, o laço do-while será executado ao menos uma vez;
- Forma geral:

```
do{
        comando;
} while(condição);
```

- O laço do-while repete até que a condição se torne falsa.

while versus do … while

- Vejamos a principal diferença entre o laço do-while e o laço while:

```
int num = 101;
do{
   scanf("%d", &num);
} while(num<100);
```

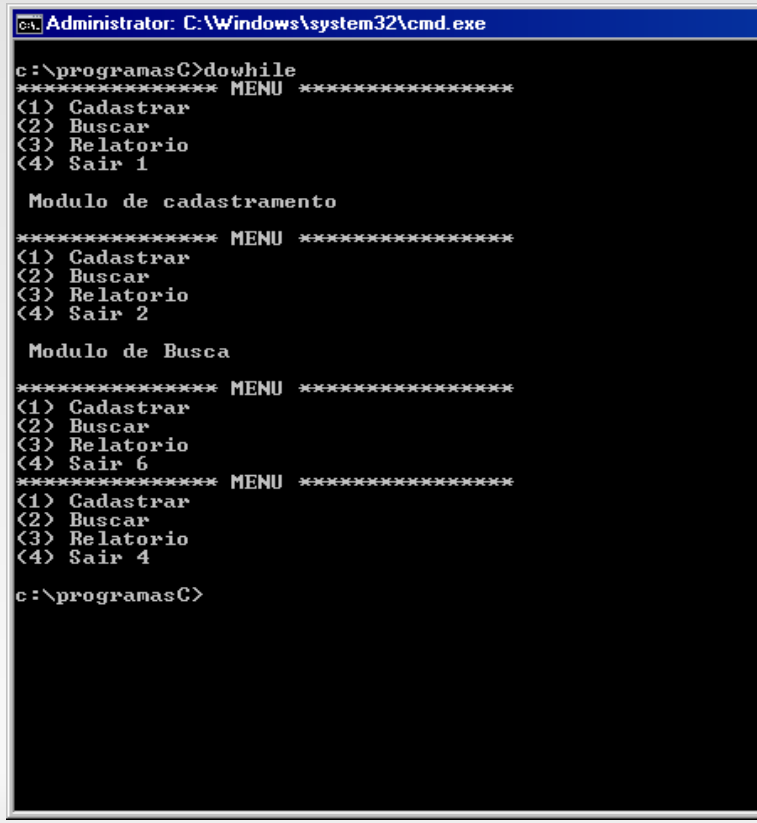
```
int num = 101;
while(num<100)
{
   scanf("%d", &num);
}
```

- do-while executa pelo menos uma vez.

Comando do ... while

- Um uso mais comum do laço do-while é uma rotina de seleção por menu, vejamos:

```
int main()
{
    int opt;
    do
    {
        do
        {
            printf("**** Menu ****");
            printf("\n(1) Cadastrar ");
            printf("\n(2) Buscar ");
            printf("\n(3) Relatorio ");
            printf("\n(4) Sair ");
            printf("\nDigite sua opcao: ");
            scanf("%d", &opt);
        } while ((opt<1) || (opt>4));
        switch(opt)
        {
                    case 1:
                         printf("\nModulo de cadastramento (III UNidade)\n\n");
                         break;
                    case 2:
                         printf("\nModulo de busca (III UNidade)\n\n");
                         break;
                    case 3:
                         printf("\nModulo de relatorios (III UNidade)\n\n");
                         break;
        }
    } while(opt != 4);
    system("pause");
    return 0;
}
```



Laços Controlados Logicamente

- Ex. do Ada:
  - Possui a versão de pré-teste mas não a pós-teste.
- Ex. do FORTRAN 77 e 90:
  - Não possui este tipo de instruções (laços lógicos).

Mecanismo de Controle de Laços pelo Usuário

- Programador deve escolher a localização do controle do laço (outra diferente do inicio ou fim).
- C , C++ e Java: **break**
  - incondicional;
  - para qualquer laço ou **switch;** um nível apenas
- Exemplo da instrução **"break":**

```
while ( true)
{    getnext(valor); // le um valor ingressado pelo usuário
     if (valor < 0 ) break;
      soma += valor
}
```

se valor for negativo o laço é finalizado.

- Java e C# possuem uma instrução break rotulada. O controle é transferido para o rótulo
- Uma alternativa para não sair do laço: **continue**. Ela pula o resto das instruções da iteração, mas não sai do laço

Iteração baseada em Estruturas de Dados

- A iteração está associada a uma estrutura de dados;
- A ordem dos elementos depende do iterador;
- O mecanismo de controle é uma chamada de função que retorna o próximo elemento, em alguma ordem, desde que exista elementos, caso contrário o laço termina.

Iteração baseada em Estruturas de Dados

- A instrução **foreach** do **C#** itera nos elementos de vetores:

Strings[] = strList = {“Bob”, “Carol”, “Ted”};

  - Instruções iterativas: Iteração baseada em Estruturas de Dados

```
foreach (Strings name in strList){
                Console.WriteLine (“Name: {0}”, name);
}
```

Iteração baseada em Estruturas de Dados

- Ex. em **Perl** (mostrar todos os elementos de um array):
  - Perl possui um iterador implícito para arrays e hashes.

```
$nomes = {"José","João","Joca"};

foreach $nome (@nomes)
{
 print $nome
}
```

Iteração baseada em Estruturas de Dados

- Ex. em **C++** (mostrar todos os elementos de um array):
  - A instrução "for" do C++ pode ser utilizada para percorrer toda a estrutura de dados.

```
char * nomes[]={"José","João","Joca",0};

for(char ** p = nomes; *p!= 0; p++)
{
                    cout << *p << end;
}
```

Iteração baseada em Estruturas de Dados

- O **for** do C, do C++ e do Java pode ser usado para simular uma instrução de Instruções iterativas:

```
for (p=root; p==NULL; traverse(p)){
   ...
}
```

Desvio Incondicional

- Uma instrução de desvio incondicional transfere o controle para outra posição do programa.

- Mecanismo mais conhecido:
  - instrução **goto**

- Problema -> Legibilidade

Desvio Incondicional

- Algumas linguagens não têm instruções de desvio incondicional
  - Modula-2
  - Java.
- Linguagem que permitem "goto" mas desaconselham a sua utilização.
  - Algol
  - Pascal
  - C
  - C++

Desvio Incondicional

- Exemplo de "goto" em C:

```
printf("Enter m for mesg, or e to end:");
scanf("%c",&letter);
if(letter=='m')
    goto A;
else
    goto B;
A: printf("\nHello!, you pressed m");
    goto FIM;
B: printf("\nBye!, ending program");
FIM:
```

Referências Bibliográficas

O material para a realização desta apresentação foram extraidos de:

* Sebesta, Linguagens de programação, 4ta edição.

* VAREJÃO, Flavio. Linguagens de Programação, 2004.

* Slides da Prof. Gláucya Carreiro Boechat