PONTIFÍCIA UNIVERSIDADE CATÓLICA DE SÃO PAULO
Ciência da Computação

Laboratório de Programação 01

Nomes e Variáveis

Prof. Carlos Eduardo de B. Paes
Departamento de Computação
Pontifícia Universidade Católica de São Paulo
carlosp@pucsp.br

1

Nomes

Finalidade: identificar elementos ou entidades de um programa

obs: o ato de identificar elementos, fatos, eventos é natural do ser humano; no âmbito da construção de linguagens/programas esse ato é fundamental - não seria possível construir uma linguagem sem um vocabulário, não seria possível construir um programa sem identificar seus elementos.

IDENTIFICADORES

⇕

ENTIDADES DO PROGRAMA ⇒ **variáveis**

⇒ **instruções ou comandos ou declarações**

2

Nomes

Questões de projeto

Qual o tamanho máximo do nome?

Existem ou não caracteres de conexão?

Distinguir ou não entre maiúscula e minúscula?

Existem palavras especiais?

3

Nomes

Tamanho máximo do nome

Alguns casos:

Um único caractere → Primeiras Linguagens
Até 6 caracteres → FORTRAN I e 77
Até 31 caracteres → FORTRAN 90 e ANSI C
Sem limite → C++ e ADA

Em C, não há limitação em princípio. Algumas implementações especificam tamanho máximo.

Exemplos de nomes (identificadores):

maximo, x,inicio, farh, Auxiliar, Quant1, printf.

4

Nomes — Tamanho máximo do nome

Recomendação - para definição do projeto da linguagem/compilador:

- ***permitir nomes razoavelmente longos e algum caractere de conexão.***

Recomendação - na escolha/construção dos identificadores:

- ***estabelecer relação existente entre identificadores e seus significados (no contexto do programa), não utilizar identificadores muito longos para facilitar a leitura (humana) do programa e sua digitação.***

5

Nomes — distinção maiúsculas/minúsculas

Exemplos:

- **Distinção entre maiúscula e minúscula**
  - ***C, C++, Java, Modula-2***
- **Somente letras maiúsculas**
  - ***FORTRAN anterior a 90***
- **Permissão de minúsculas convertidas para maiúsculas**
  - ***FORTRAN 77 e 90***
- **Sem distinção entre maiúscula e minúscula**
  - ***Pascal***

6

Nomes *distinção maiúsculas/minúsculas*

Fazer a distinção pode dificultar legibilidade (humana) e exige cuidado na escrita (digitação) de nomes pré-definidos.

Exemplo 1 - declarações inadequadas:

```
int vazao_maxima_inicial_da_tubulacao;
int vazao_maxima_Inicial_da_tubulacao;
```

(nomes muito longos e apenas a letra "i" diferencia os nomes)

Exemplo 2 - em Pascal `Var MAX: Real;`
ou `Var max: Real;`
especificam a mesma variável.

Exemplo 3 - `ReadLn(V)` **em Pascal a instrução de leitura pode ser escrita também como** `readln(V)`

Exemplo 4 - em C, a instrução de leitura `scanf("%f",v)` **não pode ser escrita como** `Scanf("%f",v)`

7

Nomes *palavras especiais*

Palavras especiais ***(nomes especiais)***
São associadas a significados específicos na linguagem:

- **especificam ações a executar**
- **especificam algum controle**
- **especificam declarações**
- **separam entidades sintáticas**

Na maioria das linguagens são palavras reservadas (não podem ter o sentido modificado) mas em algumas são apenas palavras-chave (nesse caso os compiladores fazem o reconhecimento pelo contexto).

Exemplo:

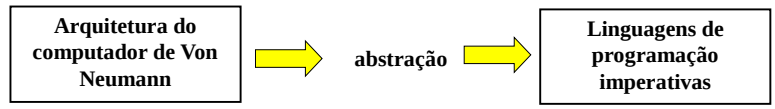
`main` **e** `scanf` **são palavras-chave (em C)**

`int` **é palavra reservada (em C)**

8

Variável

Abstração de um conjunto de células de memória
Finalidade: armazenar dados correspondentes às informações tratadas pelo programa



Arquitetura Von Neumann
Memória:
Armazena dados
Armazena instruções
Processador:
Realiza operações que modificam o conteúdo da memória

Programa (Linguagem Imperativa)
Variável
Armazena dados
Comandos - Instruções
Modificam o conteúdo de variáveis

9

Variável

no início da história (linguagem de máquina): tratamento direto variável – endereço;
depois (linguagens de alto nível): compilador responsável pela associação do nome da variável com células de memória

Atributos de uma variável

- **Nome**
- **Endereço**
- **Tipo**
- **Valor**
- **Escopo**
- **Tempo de vida**

Conceitos relacionados

- **Declarações**
- **Aliases**
- **Vinculação**
- **Tempo de vinculação**
- **Verificação de tipos**
- **Tipificação Forte**
- **Regras de Escopo**
- **Ambientes de Referenciamento**

10

Variável
endereço

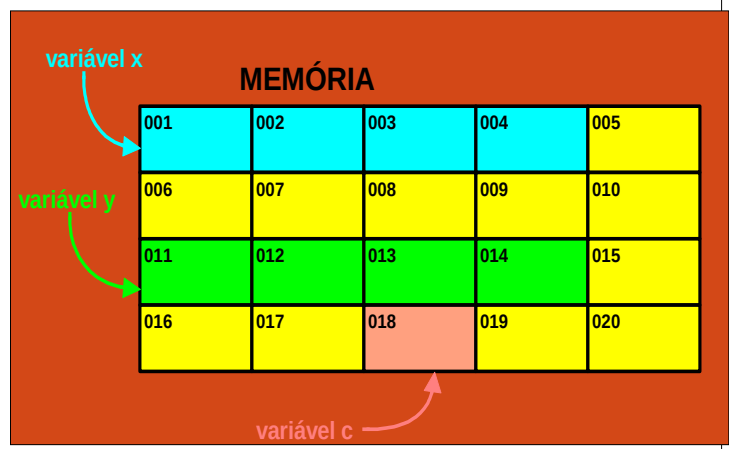
Em princípio, cada variável tem um único endereço, que é o endereço da célula de memória associada à variável.
É possível que um mesmo nome seja associado a diferentes endereços em diferentes lugares no programa e diferentes tempos de execução do programa.

- **O mesmo da célula de memória ao qual está associada a variável;**
- **Importante: o momento da vinculação do nome ao endereço;**
- **Vários nomes podem estar associados ao mesmo endereço (aliases);**
- **Mesmo nome pode estar associado a endereços diferentes (escopo);**

11

Variável endereço

```
Exemplo:
void main(){
 int x,y;
 char c;
 x=10;
 y=30
 x=x+3+y;
 c='F';
  ...
```

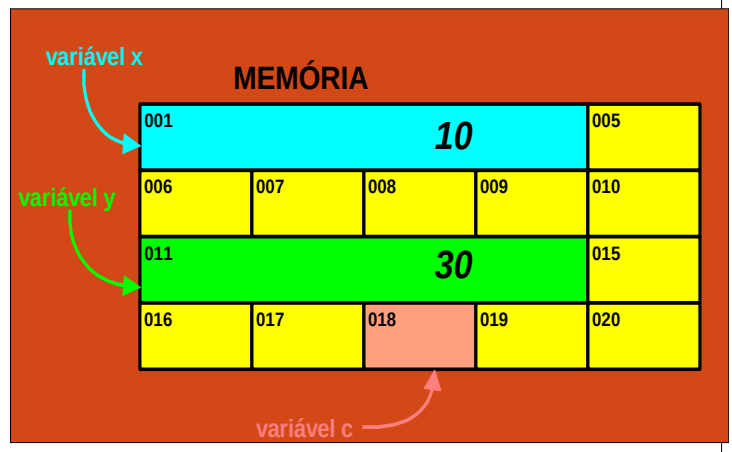


O compilador alocou as células 1 a 4 para x; 11 a 14 para y e 18 para c.;
Nesse exemplo, os endereços de x , y e c são: 001, 011 e 018.

12

Variável endereço

```
Exemplo:
void main(){
 int x,y;
 char c;
 x=10;
 y=30;
 x=x+3+y;
 c='F';
  ...
```

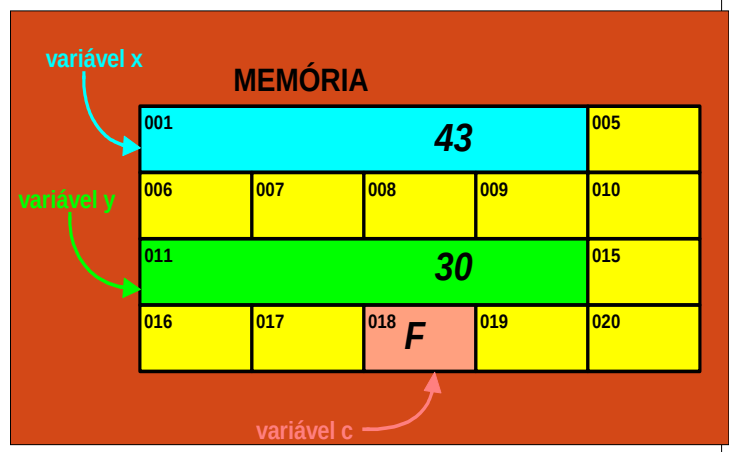


As duas primeiras atribuições, definem os conteúdos iniciais de x e y, com os valores 10 e 30.

13

Variável endereço

```
Exemplo:
void main(){
 int x,y;
 char c;
 x=10;
 y=30;
 x=x+3+y;
 c='F';
  ...
```



A terceira atribuição altera o conteúdo de x, com o valor 43; a quarta atribuição define o conteúdo de c, com o caractere F .

14

Variável *tipo*

O tipo de uma variável define:

- **a faixa ou conjunto de valores possíveis de se armazenar na variável;**
- **o conjunto de operações possíveis de se efetuar com valores do tipo considerado;**
- **o "tamanho" e a organização do conjunto de células associado à variável.**

Exemplo:

Em C, a declaração `int a,b;` define para cada variável (`a` e `b`) um conjunto de células de memória com 2 ou 4 bytes (depende da implementação), com capacidade para armazenar valores inteiros. A faixa de valores pode ser de *–32768* a *32767* (2 bytes) ou de *–2147483648* a *2147483647* (4 bytes).

Exemplos de operações possíveis:

`a / b`	>>>	**quociente inteiro da divisão de a por b**
`a % b`	>>>	**resto da divisão de a por b**

15

Variável *valor*

O valor de uma variável é o conteúdo do conjunto de células de memória associado à variável.

O valor-R é o valor armazenado em uma variável (conteúdo do conjunto de células vinculado à variável).

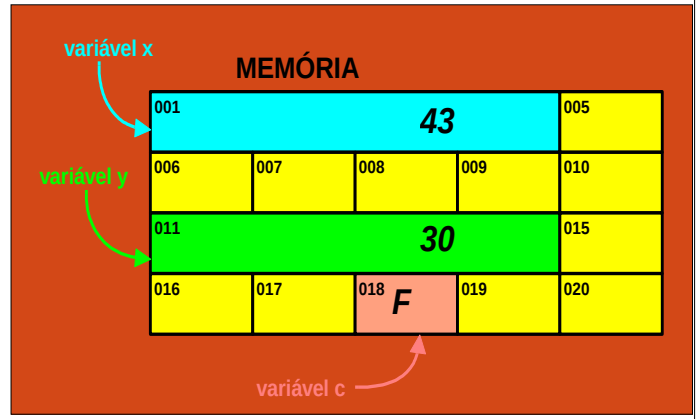
O valor-L é o seu endereço (endereço do conjunto de células vinculado à variável).

16

Variável

valor

```
Exemplo:
void main(){
 int x,y;
 char c;
 x=10;
 y=30;
 x=x+3+y;
 c='F';
  ...
```



Ao final da seqüência: variável x: valor-R 43, valor-L 001
variável y: valor-R 30, valor-L 011
variável c: valor-R 'F', valor-L 018

17

Vinculação

Conceito

Vinculação ←→ Associação entre atributo e entidade ou entre operação e símbolo.

O momento em que uma vinculação ocorre é o <u>tempo de vinculação</u>

Vinculações podem ocorrer em diversos momentos:

No tempo	**de projeto de uma linguagem**
	de implementação da linguagem
	de compilação de um programa
	de ligação (link)
	de carregamento
	de execução

18

Vinculação *Conceito*

Exemplos

projeto de linguagem
símbolo * ← → operador multiplicação

implementação de linguagem
float ← → coleção de valores

compilação
variável ← → tipo de dados específico

carregamento
variável ← → célula de memória

19

Vinculação *Tempo de Vinculação*

Exemplo: uma atribuição em C

```
int conta;
 . . .
conta = conta + 5;
```

Tempo de projeto da linguagem → variável `conta` e tipos possíveis
Tempo de compilação → variável `conta` e tipo `int`
Tempo de projeto do compilador → variável `conta` e tipos possíveis
Tempo de execução da instrução → variável `conta` e valor calculado
Tempo de definição da linguagem → Operador + e significado possível
Tempo de compilação → Significado de + na instrução
Tempo de projeto de compilador → Representação inteira do literal 5

20

Vinculação

Vinculação estática

•Ocorre antes do tempo de execução, permanece inalterada ao longo da execução.

Vinculação dinâmica

•Ocorre ou modifica-se (ou pode modificar-se) durante a execução.

21

Vinculação — Vinculação de tipo

vinculação *variável* ← → *tipo de dados*
ocorre antes de qualquer referência à variável.

Tipos podem ser especificados estaticamente por declaração explícita ou implícita.

Declaração explícita: instrução em um programa que contêm lista de nomes de variáveis e uma especificação de tipo.
Exemplos:
```
Var Base, Altura:Real;
int conta;
```

Declaração implícita: a vinculação *variável-tipo* ocorre a partir de alguma convenção própria da linguagem.
Exemplo: Fortran IV - letra inicial do nome.

22