Laboratório de Programação I

Variáveis II

Verificação de tipos, Escopo, Tempo de Vida

Baseado no capítulo 4 do livro - **CONCEITOS DE LINGUAGENS DE PROGRAMAÇÃO**
Sebesta, Robert W. Bookman

1

Objetivos

Estudar os conceitos relativos às variáveis de um programa

- **vinculação e tempo de vida,**
- **verificação de tipos,**
- **escopos.**

2

Vinculação

Vinculações de armazenamento e tempo de vida

Alocação: é a ação do sistema que corresponde a "tomar" de um conjunto de células disponíveis, aquelas que serão vinculadas a uma variável.

Desalocação: é a ação do sistema que corresponde a "devolver" ao conjunto de células disponíveis, aquelas que estavam vinculadas a uma variável.

Tempo de vida: tempo durante o qual uma variável fica vinculada a um conjunto de células de memória específico.

3

Vinculação

Vinculações de armazenamento e tempo de vida

Tempo de vida – categorias

- Variáveis estáticas
- Variáveis stack-dinâmicas
- Variáveis heap-dinâmicas

4

Vinculação

Vinculações de armazenamento e tempo de vida

Variáveis estáticas:

- **Vinculação de armazenamento definida antes do início da execução e permanece válida durante toda a execução.**
- **Característica: eficiência – endereçamento direto.**

Em C, a especificação de uma variável estática é feita dispondo-se sua declaração antes (fora) do bloco da função principal (void main()...)

```
#include <stdio.h>
float area;
void main(){
  char letra;
```

variável estática

5

Vinculação

Vinculações de armazenamento e tempo de vida

Variáveis stack-dinâmicas:

- **Vinculação de armazenamento ocorre em tempo de execução, com a elaboração de sua instrução de declaração.**
- **Característica: permite compartilhamento de memória (o conjunto de células vinculado a uma variável pode ser reutilizado, em outro momento, por outra variável).**

Elaboração da declaração
é o processo de alocação e vinculação de tipo.

```
#include <stdio.h>
float area;
void main(){
  char letra;
```

variável stack-dinâmica

6

Vinculação

Vinculações de armazenamento e tempo de vida

Variáveis heap-dinâmicas

•Células de memória alocadas/desalocadas por instruções explícitas que ocorrem em tempo de execução, especificadas pelo programador.

Em C, uma implementação desse tipo de variável é denominada ponteiro. De maneira simplificada: ponteiro é um tipo especial de variável capaz de armazenar o endereço de um conjunto de células de memória do *heap*.

7

Vinculação

Vinculações de armazenamento e tempo de vida

Variáveis heap-dinâmicas - Exemplo em C

```
#include <stdio.h>
#include <stdlib.h>
void main(){
  float *pont, *pont2;
  pont=malloc(4);
  *pont=11.11;
  *pont=*pont+10;
  free(pont);
  pont2=malloc(4);
  *pont2=22.22;
  free(pont2);
}
```

variáveis tipo ponteiro para float

alocação de um conjunto de 4 bytes

desalocação do conjunto de 4 bytes

8

Vinculação

Vinculações de armazenamento e tempo de vida

Variáveis heap-dinâmicas - exemplo em C - comentários

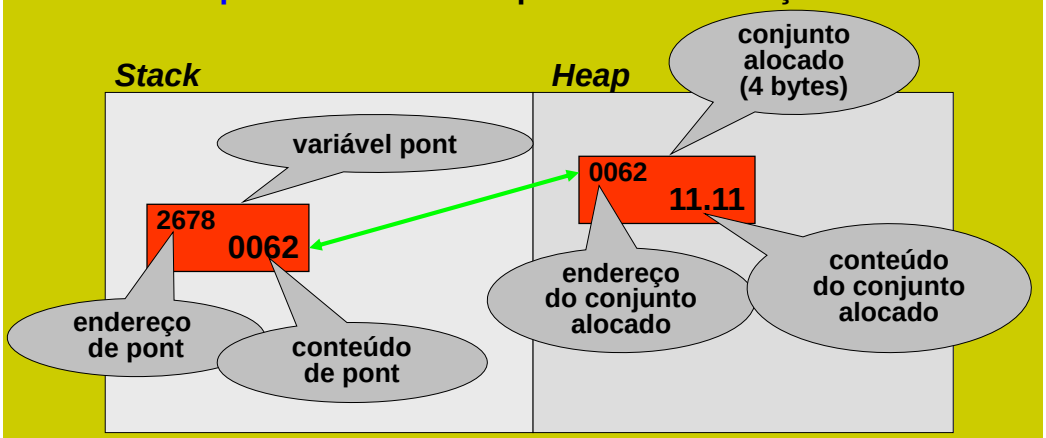
A declaração float *pont, *pont2; define dois ponteiros para float; a atribuição pont=malloc(4) faz a alocação de um conjunto de 4 bytes (float), da memória "heap" que passa a ser "apontado" por pont. Nesse momento, o conteúdo de pont será o endereço do conjunto de células alocado, o conteúdo desse conjunto de células é definido e depois redefinido, pelas atribuições *pont=11.11 e *pont=*pont+10.
A função free(pont) desaloca o conjunto de células que pont apontava. Dessa forma, a próxima atribuição pont2=malloc(4) aloca o mesmo conjunto de células (4 bytes), cujo endereço define o conteúdo do ponteiro pont2.

9

Vinculação

Vinculações de armazenamento e tempo de vida

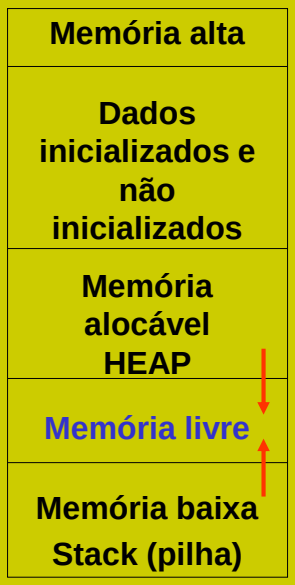
Variáveis heap-dinâmicas - exemplo em C - ilustração



10

Vinculação

Vinculações de armazenamento



código

Organização da memória

variáveis estáticas ou globais

variáveis heap-dinâmicas

variáveis stack-dinâmicas
(variáveis locais)

11

Constante Nomeada

É a vinculação entre variável e um valor único, no momento em que ocorre a vinculação variável-armazenamento.
O valor não pode sofrer modificação por qualquer instrução (atribuição, entrada ...).

Recurso empregado para melhorar legibilidade e facilitar operações de verificação/manutenção do programa.

Em C, esse recurso é disponível a partir da diretiva de pré-compilação `#define`.

Exemplo:

```
#include <stdio.h>
#define limite 100
```

Durante a pré-compilação todas as ocorrências do identificador limite, no texto do programa, serão substituídas pelo valor 100.

12

Inicialização de variável

É a vinculação entre a variável e um valor, no momento em que ocorre a vinculação variável-armazenamento.
O valor pode sofrer modificação por qualquer instrução (atribuição, entrada ...).

Exemplo em C:

```
#include <stdio.h>
void main(){
  int passo=4;
  float fator=1.22;
```

no momento da elaboração da declaração ocorrem as vinculações variável-tipo, variável-armazenamento e variável-valor.

13

Verificação de tipos

Finalidade: ***assegurar que operandos de um operador sejam de tipos compatíveis (tipos adequados ao operador).***
Objetivo: ***detectar erros de tipo.***

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

Considere o segmento de programa:

```
#include <stdio.h>
float area;
void main(){
  char letra;
```

Nesse exemplo, area (variável estática) será vinculada ao tipo float em tempo de carregamento (antes do início da execução); letra (variável stack-dinâmica) será vinculada ao tipo char em tempo de execução (no início da execução do programa).

14

Verificação de tipos

Ao definir-se a vinculação variável-tipo, ficam estabelecidos:

- **o conjunto ou faixa de valores que constituem os possíveis conteúdos da variável;**
- **a forma de representação interna desses conteúdos;**
- **o "tamanho" do conjunto de células de memória associado à variável;**
- **o conjunto de operações para as quais a variável poderá ser referenciada como operando ou alvo.**

15

Verificação de tipos

Como operador considera-se: operador aritmético, operador lógico, operador relacional, instrução de atribuição.
Como operando considera-se: valores constantes, variáveis, expressões, alvo de atribuição.

Tipo compatível é aquele que é válido ou pode ser convertido automaticamente para um tipo válido para um operador.
Essa conversão automática é denominada coerção.

Considere o segmento de programa:

```
int a,b;
float x;
  ...
x=3*a/(2*b);
```

o valor inteiro obtido como resultado da expressão `3*a/(2*b)` é convertido para real (float) ao ser armazenado em x

16

Verificação de tipos

A verificação de tipos pode ser feita estaticamente - se todas as vinculações de tipos forem estáticas, nesse caso a verificação de tipos ocorre em tempo de compilação.

A vinculação dinâmica de tipos exige a verificação de tipos em tempo de execução. Essa forma de verificação é mais difícil de ser implementada porque as células de memória podem armazenar valores de tipos diferentes em diferentes momentos da execução.

Alguns exemplos:
- **uniões em C e C++,**
- **registros variantes em Pascal,**
- ***equivalence* em Fortran.**

17

Tipificação forte

Uma linguagem é fortemente tipificada se todos os erros de tipo são detectados em tempo de compilação ou de execução.

FORTRAN, C, C++
não são fortemente tipificadas.

ML
é fortemente tipificada

Ada, Modula 3, Java, Pascal
quase fortemente tipificadas.

18

Regras de compatibilidade de tipos

Há dois métodos básicos para verificação de compatibilidade de tipos:

- **compatibilidade por nome de tipo;**
- **compatibilidade por estrutura de tipo.**

A compatibilidade por *nome de tipo* é mais fácil de ser implementada.
No caso mais simples, a compatibilidade é verificada confrontando-se apenas os nomes dos tipos.
Assim, duas variáveis são de tipos compatíveis se estiverem na mesma declaração ou em declarações que usem o mesmo nome de tipo.

A compatibilidade por estrutura tem a implementação mais difícil. A verificação é fundamentada na definição da estrutura do tipo e não apenas em seu nome.
Por ser mais complexa, admite variedade de critérios.

19

Escopo *bloco*

Bloco é uma seção ou área ou faixa de código, delimitada de alguma forma, dentro do texto de um programa.

Exemplo em C:

```
...
if(x>y){
   temp=x;
   x=y;
   y=temp;
}
...
```

bloco delimitado por { … }

20

Escopo

Escopo de uma variável (ou outra entidade) é a área (faixa) de instruções do programa onde a variável (ou entidade) é reconhecida (pode ser referenciada).

Regras de escopo de variáveis definem:

- **como a ocorrência de um nome está associada à variável;**
- **como são consideradas as vinculações nome←→variável←→atributos;**
- **como as referências a variáveis declaradas fora do bloco que estiver em execução são associadas às suas declarações e, portanto a seus atributos.**

21

Escopo

Uma variável é <u>local</u> em um bloco ou em uma unidade se tiver sido declarada nesse bloco ou unidade.

Uma variável é <u>não local</u> em um bloco se sua declaração não estiver contida nesse bloco, mas puder ser referenciada nesse bloco.

Exemplo em C:

```
float x,y;
...
if(x>y){
   float temp;
   temp=x; x=y; y=temp;
}
...
```

nesse bloco, `temp` é local e `x,y` são não locais

22

Escopo

Há dois métodos básicos para a definição de escopos:
- **escopo estático**
- **escopo dinâmico**

O escopo estático pode ser determinado estaticamente antes da execução (durante a compilação), e é fundamentado basicamente em relações espaciais (áreas ou faixas de código).

O escopo dinâmico é definido em tempo de execução, não pode ser determinado durante a compilação; depende da seqüência de ações estabelecidas durante a execução do programa.

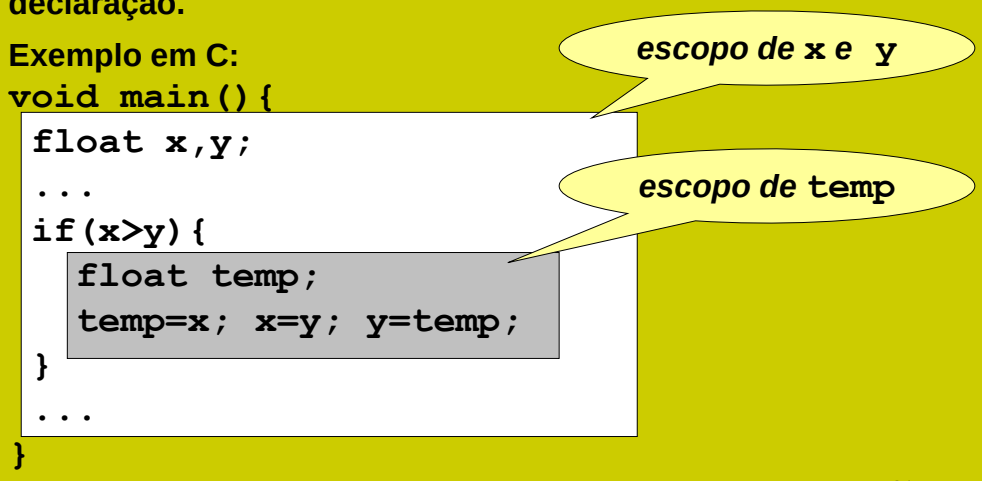
Nesse momento do curso será tratado apenas o conceito de escopo estático.

23

Escopo

Em C, o escopo de uma variável é a área de código abaixo de sua declaração até o final do bloco onde está contida sua declaração.

Exemplo em C:



24

Escopo e tempo de vida

Escopo e Tempo de Vida parecem estar relacionados mas são conceitos diferentes:

> ***tempo de vida*** **é um conceito temporal - o tempo de vida de uma variável é o tempo transcorrido entre as ações de alocação e desalocação da variável.**
>
> ***escopo*** **é um conceito espacial ou textual - o escopo de uma variável é a área de texto de um programa onde a variável pode ser referenciada.**

Ambiente de referenciamento

Ambiente de referenciamento de uma instrução é o conjunto de todas as variáveis (ou entidades) que se pode referenciar nessa instrução.

25