ALGORITMOS E PROGRAMAÇÃO DE COMPUTADORES

Disciplina: 113476

Profa. Carla Denise Castanho

Universidade de Brasília – UnB
Instituto de Ciências Exatas – IE
Departamento de Ciência da Computação – CIC

5. SUBALGORITMOS

ESCOPO DE VARIÁVEIS

Escopo de Variáveis

- Lembra quando dissemos que os **nomes das variáveis** no programa que chama uma função não têm nenhuma relação com os **nomes dos parâmetros** dessa função?
- E lembra quando dissemos que **parâmetros também são variáveis** dentro da função?
- Mas e se o programa principal e uma função têm variáveis **com o mesmo nome**? Como funciona? Opa... e pode isso?

Escopo de Variáveis

- Quando sugiu a idéia de dividir os algoritmos em vários **pedaços** menores, isto é, em **módulos**, surgiu também a necessidade de **diferenciar nomes iguais em módulos diferentes** de um programa. Para isso, criou-se a noção de **escopo**.
- O **escopo** de um **nome** é uma **região** de código **bem definida**, onde o **nome é visível e acessível**.
- Um **nome** pode referir-se a qualquer identificador, isto é, variáveis, constantes, funções, etc... No entanto, aqui estamos preocupados especificamente com o **escopo de variáveis**. Você vai entender o escopo de outros tipos de nomes por analogia...

Escopo de Variáveis

- Vamos pensar sobre o assunto...
- Quando declaramos uma variável no **programa principal**, quais partes do nosso algoritmo sabem que ela **existe** e podem **acessar** essa variável (ler e escrever nela)?
- Certamente todo o código que está no programa principal! Mas e as funções (subalgoritmos)?
- A regra é bem simples: cada macaco no seu galho! **Uma função não enxerga as variáveis da outra!**

Escopo de Variáveis

- Lembre-se da Linguagem C: a *main* é apenas mais uma função. Logo, uma função auxiliar não enxerga as variáveis do programa principal ou de outras funções, e vice-versa!

Escopo de Variáveis

- Lembre-se da Linguagem C: a *main* é apenas mais uma função. Logo, uma função auxiliar não enxerga as variáveis do programa principal ou de outras funções, e vice-versa!

Exemplo - Escopo em Funções

```
Algoritmo FatorialModular

Função Fatorial(n : inteiro) : inteiro
Variáveis
    fat : inteiro
Início
    fat ← 1
    Enquanto n > 1 faça
        fat ← fat * n
        n ← n - 1
    Fim-Enquanto
    retorne fat
Fim

Variáveis
    fat, n : inteiro
Início
    Escreva ("Informe um número inteiro:")
    Leia (n)
    fat ← Fatorial(n)
    Escreva ("O fatorial de ", n, " é ", fat, ".")
Fim
```

Escopo de Variáveis

- Lembre-se da Linguagem C: a *main* é apenas mais uma função. Logo, uma função auxiliar não enxerga as variáveis do programa principal ou de outras funções, e vice-versa!

Exemplo - Escopo em Funções

```
Algoritmo FatorialModular

Função Fatorial(n : inteiro) : inteiro
Variáveis
    fat : inteiro
Início
    fat ← 1
    Enquanto n > 1 faça
        fat ← fat * n
        n ← n - 1
    Fim-Enquanto
    retorne fat
Fim

Variáveis
    fat, n : inteiro
Início
    Escreva ("Informe um número inteiro:")
    Leia (n)
    fat ← Fatorial(n)
    Escreva ("O fatorial de ", n, " é ", fat, ".")
Fim
```

Essas variáveis *n* e *fat* são da função *Fatorial*, só existem nessa função!

Essas aqui são do programa principal, não estão relacionadas com as da função! O mesmo nome, mas variáveis diferentes!!!

Variáveis Locais *vs* Globais

- No exemplo anterior, dizemos que *n* e *fat* são variáveis com escopo local ou, ainda, que são **variáveis locais**.
- Isso porque elas só estão acessíveis em uma região bem **localizada** do código, ou seja, no corpo de uma função, e em nenhum outro lugar.
- Além disso, cada chamada da função é **independente** de outras chamadas passadas ou futuras! As variáveis locais são **criadas do zero**, com novos valores (lixo). Os parâmetros recebem os valores com que a função foi chamada (lembra, eles são variáveis locais também!).
- Mas e se quiséssemos criar **variáveis globais**, isto é, variáveis acessíveis a todos os módulos?
- Vamos criar uma nova seção, no início do programa, que conterá estas variáveis globais...

Variáveis Locais *vs* Globais

Exemplo - Variáveis Globais

```
Algoritmo Globalize
Variáveis
    nome : string
    nota : real

Função LeAluno()
Variáveis
    nota_temp : real
Início
    Leia (nome)
    Faça
        Escreva ("Informe a nota entre 0 e 10:")
        Leia (nota_temp)
    Enquanto (nota_temp < 0) OU (nota_temp > 10)
    nota ← nota_temp
Fim

Variáveis
    n, i : inteiro
    media : real
Início
    Escreva ("Informe o numero de alunos:")
    Leia (n)
    Para i ← 1 até n faça
        LeAluno()
        media ← media + nota
    Fim-Para
    Escreva ("A média foi: ", media / n, ".")
Fim
```

Variáveis Locais *vs* Globais

Exemplo - Variáveis Globais

```
Algoritmo Globalize
Variáveis
    nome : string
    nota : real

Função LeAluno()
Variáveis
    nota_temp : real
Início
    Leia (nome)
    Faça
        Escreva ("Informe a nota entre 0 e 10:")
        Leia (nota_temp)
    Enquanto (nota_temp < 0) OU (nota_temp > 10)
    nota ← nota_temp
Fim

Variáveis
    n, i : inteiro
    media : real
Início
    Escreva ("Informe o numero de alunos:")
    Leia (n)
    Para i ← 1 até n faça
        LeAluno()
        media ← media + nota
    Fim-Para
    Escreva ("A média foi: ", media / n, ".")
Fim
```

SEÇÃO DE VARIÁVEIS GLOBAIS

Variáveis Locais *vs* Globais

Exemplo - Variáveis Globais

```
Algoritmo Globalize
Variáveis
    nome : string
    nota : real

Função LeAluno()
Variáveis
    nota_temp : real
Início
    Leia (nome)
    Faça
        Escreva ("Informe a nota entre 0 e 10:")
        Leia (nota_temp)
    Enquanto (nota_temp < 0) OU (nota_temp > 10)
    nota ← nota_temp
Fim

Variáveis
    n, i : inteiro
    media : real
Início
    Escreva ("Informe o numero de alunos:")
    Leia (n)
    Para i ← 1 até n faça
        LeAluno()
        media ← media + nota
    Fim-Para
    Escreva ("A média foi: ", media / n, ".")
Fim
```

SEÇÃO DE VARIÁVEIS GLOBAIS

Essas variáveis são globais, acessíveis a todo o código!

Essas variável é local, somente LeAluno sabe de sua existência!

Essas variáveis são locais, somente o programa principal sabe de sua existência!

Variáveis Locais *vs* Globais

- **ATENÇÃO!** Evite ao máximo utilizar variáveis globais, elas só devem ser utilizadas em situações muito específicas: apenas quando existir efetivamente algum valor que precise ser compartilhado ou manipulado por vários módulos diferentes do programa. O algoritmo anterior foi apenas um exemplo, veremos uma solução melhor sem variáveis globais, na próxima seção!
- Lembre-se: quando decidimos **modularizar** nossos programas, nosso objetivo era diminuir a complexidade: cada módulo deve realizar apenas sua **tarefa específica**, com o **mínimo de acoplamento** (*i.e.*, dependência, conexões) em relação aos outros módulos.
- Quando criamos uma variável global, todos os módulos podem potencialmente manipulá-la, fica muito mais **difícil dar manutenção e corrigir erros** no código!
- Além disso, elas desperdiçam memória: **variáveis locais só existem quando seu escopo está ativo**, ou seja, só enquanto são necessárias, **variáveis globais existem sempre**!
- Se a sua primeira solução envolve uma variável global, é altamente provável que você esteja modelando o problema de uma maneira errônea! Repense a sua solução e tente encontrar um jeito mais elegante de resolver o problema!

Escopo de Variáveis

- OK, eu entendi a diferença entre variáveis locais e globais, e que devo evitar usar essas últimas...
- Mas o que acontece se uma **variável local** e uma **variável global** têm o **mesmo nome**?
- Nesse caso, aplicamos nossa segunda regra simples: **o escopo mais interno tem preferência sobre o escopo mais externo!**
- Vamos entender melhor esse conceito...

Escopo de Variáveis

- Em pseudocódigo, existem apenas dois escopos: o local e o global.
- É óbvio que o escopo local está contido no escopo global. Logo, o **escopo local é o mais interno**.
- Portanto, em pseudocódigo, a **variável** declarada **local**mente sempre **tem preferência** sobre a **variável global**.
- Isto é, se você se referir ao nome de uma variável qualquer, estará referindo-se ao escopo local. Somente se essa variável não existir no escopo local é que será utilizada a variável global...

Variáveis Locais *vs* Globais

Exemplo - Variáveis Globais vs Locais

```
Algoritmo Globalize2
Variáveis
    nome : string
    nota : real

Função LeAluno() : real
Variáveis
    nota : real
Início
    Leia (nome)
    Faça
        Escreva ("Informe a nota entre 0 e 10:")
        Leia (nota)
    Enquanto (nota < 0) OU (nota > 10)
    Retorne nota
Fim

Variáveis
    n, i : inteiro
    media : real
Início
    Escreva ("Informe o numero de alunos:")
    Leia (n)
    Para i ← 1 até n faça
        nota ← LeAluno()
        media ← media + nota
    Fim-Para
    Escreva ("A média foi: ", media / n, ".")
Fim
```

A referência a nome é à variável global, já que não existe variável local com o mesmo identificador.

Todas a referências a nota são sempre à variável local, ela tem preferência sobre a nota global!

Aqui, a referência a nota é à variável global, já que não existe variável local com o mesmo identificador.

Escopo de Variáveis

- Se existem apenas dois escopos, porque a regra diz que o escopo **mais interno** tem preferência sobre o **mais externo**?
- Várias linguagens de programação, inclusive a linguagem C, definem a noção de **aninhamento** de escopos.
- Do mesmo jeito que você pode colocar um *if* dentro de outro, ou um *loop* dentro de outro, você pode criar um **novo escopo dentro de outro escopo**.
- Na verdade, em C, cada vez que você abre um **bloco** – `{ . . . }` – você está **criando um novo escopo**. E este novo escopo é *mais interno* em relação ao escopo anterior...
- Vamos entender melhor...

Escopo de Variáveis

- Vamos filosofar um pouco...
- Você está no planeta Terra, no Continente Americano, no país Brasil, na unidade da federação Distrito Federal, na cidade de Brasília, no bairro Asa Norte, na região Campus Universitário Darcy Ribeiro, na Universidade de Brasília, no Pavilhão João Calmon, na sala... você entendeu!
- Agora reflita:
  - No planeta Terra, todos conhecem os nomes América, Antártida, Oceano Atlântico, Grande Barreira de Corais...
  - Na América, todos conhecem os nomes Cordilheiras dos Andes, Amazônia, Golfo do México...
  - No Brasil, todos conhecem os nomes Pantanal, Pico da Bandeira, Pampas Gaúchos...
  - No DF, todos conhecem os nomes Ermida Dom Bosco, Plano Piloto, Água Mineral...
  - Em Brasília, todos conhecem os nomes Esplanada, Eixinho, Parque da Cidade...
  - Na Asa Norte, todos conhecem os nomes Deck Norte, Pôr do Sol, Conjunto Nacional...
  - No Campus Darcy Ribeiro, todos conhecem os nomes UnB, CEU, CDT, Multiuso...
  - Na Universidade de Brasília, todos conhecem os nomes Minhocão, Pavilhões, etc...

Escopo de Variáveis

- Agora, sabendo que existe um prédio chamado “Minhocão” em São Paulo, se estivermos na UnB e eu me referir ao Minhocão, a qual prédio você vai assumir que eu estou me referindo?
- Os escopos nas linguagens de programação funcionam exatamente do mesmo jeito: vale sempre o escopo mais local, mais restrito, ou seja, **o último escopo que foi aberto**.
- Tudo que existe no escopo pai também existe no escopo filho. No entanto, se você declarar uma nova variável no escopo filho com o mesmo nome de uma variável já existente, a nova variável terá preferência! Por isso, o escopo local tem preferência sobre o global.
- Lembre-se! **Não é possível declarar duas variáveis com o mesmo nome em um mesmo escopo**, mas é possível fazê-lo em um novo escopo *mais interno*.
- O escopo **global** é o escopo raiz, é o escopo **mais amplo possível**!
- Agora você tem mais um motivo para não utilizar variáveis globais: elas podem causar **ambiguidades** quando têm o mesmo nome que variáveis locais!

Escopo na Linguagem C

- Em CB, vamos utilizar no máximo dois escopos: local (função) e global. Como funcionam em C?

Escopo na Linguagem C

- Em CB, vamos utilizar no máximo dois escopos: local (função) e global. Como funcionam em C?

Exemplo - Escopo em C

```c
#include <stdio.h>
char g_nome[30];
float g_nota;

void le_aluno () {
    scanf("%s", g_nome);
    scanf("%f", &g_nota);
    while ((g_nota < 0) || (g_nota > 10))
        scanf("%f", &g_nota);
}

int main () {
    int n, i;
    float media;
    scanf("%d", &n);
    for (i = 0; i < n; ++i) {
        le_aluno();
        media += g_nota;
    }
    printf("A media eh: %f.\n", media / n);

    return 0;
}
```

Escopo na Linguagem C

- Em CB, vamos utilizar no máximo dois escopos: local (função) e global. Como funcionam em C?

Exemplo - Escopo em C

```c
#include <stdio.h>
char g_nome[30];
float g_nota;

void le_aluno () {
    scanf("%s", g_nome);
    scanf("%f", &g_nota);
    while ((g_nota < 0) || (g_nota > 10))
        scanf("%f", &g_nota);
}

int main () {
    int n, i;
    float media;
    scanf("%d", &n);
    for (i = 0; i < n; ++i) {
        le_aluno();
        media += g_nota;
    }
    printf("A media eh: %f.\n", media / n);

    return 0;
}
```

Em C, variáveis declaradas fora de qualquer função são globais. É uma boa prática usar o prefixo g_ nessas variáveis, para evitar confusões e chamar atenção para o fato de que são globais!
Lembre-se também: na Linguagem C, é necessário declarar tudo antes de usar, por isso, as variáveis globais vêm no começo do programa.

Conforme o esperado, as variáveis declaradas dentro do bloco de uma função são locais. Lembre-se: abriu chaves, abriu um novo escopo!

Escopo – Exercício

1. Escreva um algoritmo que possui duas variáveis globais x e y e uma função $LePonto$, sem parâmetros e sem retorno. A função $LePonto$ deve pedir ao usuário as coordenadas de um ponto no plano cartesiano, que serão armazenadas em x e y. O programa principal deve solicitar dois pontos e validar se possuem abscissas diferentes: $(x1 \neq x0)$. Caso os pontos sejam válidos, o programa deve calcular e mostrar o coeficiente angular da reta que passa entre os dois pontos, caso contrário, deve mostrar uma mensagem de erro. Você pode criar outras funções auxiliares, caso ache necessário.