FACAMP
FACULDADES DE CAMPINAS

Linguagens de Programação

Aula 2

Subprogramas, passagem de parâmetros e noções de escopo

2º semestre de 2019
Prof José Martins Junior

Modularização

- Problema
  - Os programas tendem a ficar grandes
  - Principalmente quando realizam diversas operações e funcionalidades
- Uma forma inteligente
  - Dividir o problema e resolvê-lo em partes
  - Cada parte, ou módulo, resolve um pedaço

Modularização

- Característica básica da programação estruturada
  - Modular a resolução do problema através da sua divisão em subproblemas menores e mais simples
  - Neste processo, cada subproblema pode ser analisado de forma individual e independente dos demais
- Objetivo
  - Facilitar o trabalho com problemas complexos
  - Permitir a reutilização de módulos
    - Com a modularização de um programa, as partes que o compõem podem ser desenvolvidas por diferentes equipes

Modularização

- Refinamentos Sucessivos (top-down):
  - Divisão do problema inicial em subproblemas, e estes em partes ainda menores, sucessivamente, até que cada parte seja descrita através de um algoritmo claro e bem-definido
    - Um algoritmo que resolve um determinado subproblema é denominado subalgoritmo

Primórdios: GOTO

- Instrução presente em várias linguagens
  - Permite o **desvio incondicional** da linha de execução
  - Referencia um rótulo (label) sintático que indica para onde o ponteiro de instruções será enviado
  - Em Assembly, instrução JMP, mas haviam condicionais, geralmente precedidas de uma operação CMP
    - JE - Jump if Equal
    - JNE - Jump if Not Equal
    - JAE/JNB - Jump if Greater or Equal ou Jump if Not Less
    - JB/JNAE - Jump if Less ou Jump if Not Greater or Equal
    - JBE/JNA - Jump if Less or Equal ou Jump if Not Greater
  - ATENÇÃO: o uso de GOTO não é uma boa prática!!

Exemplo de GOTO em Basic

```
100 PRINT "Adivinhe! Quanto é 2 + 2?"
200 INPUT A
300 IF A = 4 THEN GOTO 600 ENDIF
400 PRINT "Errooooou... De novo!"
500 GOTO 200
600 PRINT "Acertooou! Gênio!"
700 END
```

Compilador online: https://www.tutorialspoint.com/execute_basic_online.php

Subprogramação

- As linguagens de programação oferecem algum tipo de suporte à subprogramação
- Exemplos:
  - C: funções
  - Pascal: procedimentos e funções
  - Java: métodos
  - Basic: sub-rotinas
- Em linhas gerais, existem duas categorias de subprogramas
  - Procedimentos
  - Funções

Exemplo de GOSUB em Basic

```
1 PRINT "Inicio"
2 GOSUB 1000
3 PRINT "Fim"
4 END
1000 REM Subrotina
1001 PRINT "Meio"
1002 RETURN
```

Compilador online: https://www.tutorialspoint.com/execute_basic_online.php

Procedimentos

- Forma de criar um subprograma
  - Quando um determinado conjunto de instruções tiver que ser repetido dentro da solução de um problema, é conveniente colocá-lo dentro de um procedimento
- Para se criar um procedimento é necessário
  - Um identificador (o nome do procedimento)
  - Uma lista de parâmetros (que possibilitam a comunicação entre o programa principal e o procedimento)
  - As ações a serem executadas (que formam o corpo do procedimento)
- Um procedimento não retorna nada

Exemplo de procedimento em Java

- Exibir uma mensagem formatada ao usuário

```
public static void exibirMensagem(String usuario) {
        System.out.println("Olá " + usuario + ", bom dia!");
}
```

- Chamada do procedimento no método main

```
public static void main(String[] args) {
        exibirMensagem("Maria");
}
```

- Exibirá na tela

```
Olá Maria, bom dia!
```

Exemplo de procedimento em C

- Exibir uma mensagem formatada ao usuário

```
void exibirMensagem(char *usuario) {
        printf("Olá %s, bom dia!\n", usuario);
}
```

- Chamada do procedimento na função main

```
int main() {
        exibirMensagem("Maria");
        return 0;
}
```

- Exibirá na tela

```
Olá Maria, bom dia!
```

Funções

- Também é uma forma de criar um subprograma
  - A função deve obrigatoriamente retornar um valor processado através do seu nome identificador
  - Uma função deve ser ativada em um contexto de expressão

Exemplo de função em Java

- Calcular a soma de dois números reais

```
public static double somaReal(double x, double y) {
        double r;
        r = x + y;
        return r;
}
```

- Chamada do procedimento no método main

```
public static void main(String[] args) {
        double res = somaReal(4.0, 3.0);
        System.out.println("Resultado = " + res);
}
```

- Exibirá na tela

```
Resultado = 7.0
```

Exemplo de função em C

- Calcular a soma de dois números reais

```
double somaReal(double x, double y) {
        double r;
        r = x + y;
        return r;
}
```

- Chamada do procedimento na função main

```
int main() {
        double res = somaReal(4.0, 3.0);
        printf("Resultado = %f\n", res);
        return 0;
}
```

- Exibirá na tela

```
Resultado = 7.000000
```

Escopo das Variáveis

- Cada módulo é tratado independentemente
  - Sendo assim, pode manipular variáveis e constantes
    - Externas ou globais
      - Declaradas no programa principal
      - Devem ser referenciadas ao módulo
    - Locais
      - Declaradas no próprio módulo
      - Só têm validade dentro do bloco no qual são declaradas

Exemplo de global em C

- Variáveis a e b declaradas externamente
- Função troca mudará os valores delas

```
#include <stdio.h>
int a, b;
void troca() {
  int aux;
  aux = a;
  a = b;
  b = aux;
}
int main() {
  a = 1;
  b = 2;
  printf("a = %d\nb = %d\n", a, b);
  troca();
  printf("a = %d\nb = %d\n", a, b);
  return 0;
}
```

Compilador online: https://code.dcoder.tech/

Exemplo de contextos em Java

```
class Exemplo {
    public static int n = 10;
    public static void main(String args[]) {
        int n = 20;
        System.out.println("n = " + n);
        imprimeN();
    }
    public static void imprimeN() {
        System.out.println("n = " + n);
    }
}
```

- Resultado

```
n = 20
n = 10
```

Compilador online: https://code.dcoder.tech/

Passagem de Parâmetro

- Meio de comunicação entre as módulos para
  - Apenas fornecer um valor para que a subrotina realize um processamento
  - Apenas retornar um valor processado pela subrotina
  - Fornecer um valor para processamento pela subrotina, e também ser responsável pelo retorno de um valor processado

Passagem de Parâmetro

- Duas formas de passagem
  - Por valor
    - O argumento será uma variável independente, ou seja, ocupará um espaço na memória durante a execução do módulo
    - Qualquer alteração no argumento, não afeta a variável externa
  - Por referência
    - O argumento não ocupará espaço de memória, pois utilizará o espaço já alocado pela variável declarada externamente
    - Qualquer alteração realizada no subalgoritmo afetará a variável externa

Passagem de parâmetros por valor

- Quando e porque passagem por valor
  - Apenas fornecer valores à subrotina para que ela realize um determinado processamento
  - Utilizados somente como “valores de entrada”
  - Protegem automaticamente o parâmetro real (variável externa ao módulo)
  - Deve ser explorado sempre que possível
  - Em Java, todos os parâmetros, exceto arranjos (vetores e matrizes) são passados por valor
  - Algumas linguagens permitem passar por referência
    - PASCAL (declaração var)
    - C (com ponteiros)

Exemplo de passagem por valor

- Imprimirá **José** na tela
  - A alteração interna da variável no método exemplo1 não será repassada ao método main

```
public static void main(String[] args) {
    String nome = "José";
    exemplo1(nome);
    System.out.println(nome);
}
public static void exemplo1(String n) {
    n = "João";
}
```

Exemplo de passagem por valor em C

```
#include <stdio.h>
void troca(int a, int b) {
  int aux;
  aux = a;
  a = b;
  b = aux;
}
int main() {
  int a = 1;
  int b = 2;
  printf("a = %d\nb = %d\n", a, b);
  troca(a, b);
  printf("a = %d\nb = %d\n", a, b);
  return 0;
}
```

Compilador online: https://code.dcoder.tech/

Passagem de parâmetros por referência

- Quando e porque passagem por referência
  - Quando a unidade ativada (subrotina) necessitar retornar um valor a ser utilizado pela unidade ativadora
  - As alterações realizadas pelo módulo afetarão a variável externa
  - Seu uso deve ser cuidadoso
  - Em Java, como em outras linguagens, os arranjos (vetores e matrizes) são passados por referência

Exemplo de passagem por referência

- Imprimirá **João** na tela
  - A alteração interna da variável no método exemplo2 será repassada ao método main

```
public static void main(String[] args) {
    String[] nomes = {"José"};
    exemplo2(nomes);
    System.out.println(nomes[0]);
}
public static void exemplo2(String[] n) {
    n[0] = "João";
}
```

Exemplo de passagem por referência em C

```
#include <stdio.h>
void troca(int *a, int *b) {
  int aux;
  aux = *a;
  *a = *b;
  *b = aux;
}
int main() {
  int a = 1;
  int b = 2;
  printf("a = %d\nb = %d\n", a, b);
  troca(&a, &b);
  printf("a = %d\nb = %d\n", a, b);
  return 0;
}
```

Compilador online: https://code.dcoder.tech/

Bibliografia

- SEBESTA, R. W. Conceitos de Linguagens de Programação. Porto Alegre: Bookman, 2011. 792p.