

MC-102 — Aula 06

Comandos Condicionais II

Instituto de Computação – Unicamp

Primeiro Semestre de 2006

Roteiro

- 1 Simulação de código
- 2 Decisão simples e decisão múltipla
- 3 O comando `switch`

Introdução

- Às vezes, acontece de programarmos um código, porém ele não faz o que esperávamos que fizesse.
- Isso acontece por vários motivos, entre os quais destacam-se:
 - Erros de programação: instruções escritas erradas.
 - Erros da nossa lógica: o conjunto de passos pensados que parecia resolver o problema na realidade não cobre todas as situações.
- Eventualmente, simplesmente olhar o código pode não trazer a tona o erro.
- Por isso, utiliza-se uma técnica de simulação do código
 - Pode ser automatizada (utilizando um *debugger*)
 - Pode ser feita manualmente, utilizando papel e caneta.

Simulação Manual

- Bem simples: Existem apenas 2 passos.
 - “Alocação” dos espaços de variáveis
 - “Execução” de uma instrução de cada vez.
- Alocação de memória:
 - Ex. Suponha o código:
 1. `int divisor,dividendo;`
 2. `float resultado;`
- Após “executar” a linha 1

Tipo	<code>int</code>	<code>int</code>
Nome	<code>divisor</code>	<code>dividendo</code>
Valor	?	?

Simulação Manual

- Bem simples: Existem apenas 2 passos.
 - “Alocação” dos espaços de variáveis
 - “Execução” de uma instrução de cada vez.
- Alocação de memória:
 - Ex. Suponha o código:
 1. `int divisor,dividendo;`
 2. `float resultado;`
- Após “executar” a linha 2

Tipo	<code>int</code>	<code>int</code>	<code>float</code>
Nome	<code>divisor</code>	<code>dividendo</code>	<code>resultado</code>
Valor	?	?	?

Simulação Manual

- Execução em memória:
 - Ex. Suponha o código:
 1. `int divisor,dividendo;`
 2. `float resultado;` ← Último executado
 3. `divisor=10;` ← Próximo Comando
 4. `dividendo=13;`
 5. `resultado = dividendo / divisor;`
- Após “executar” a linha 2

Tipo	<code>int</code>	<code>int</code>	<code>float</code>
Nome	<code>divisor</code>	<code>dividendo</code>	<code>resultado</code>
Valor	?	?	?

Simulação Manual

- Execução em memória:
 - Ex. Suponha o código:
 1. `int divisor,dividendo;`
 2. `float resultado;`
 3. `divisor=10;` ← Último executado
 4. `dividendo=13;` ← Próximo Comando
 5. `resultado = dividendo / divisor;`
 - Após “executar” a linha 3

Tipo	int	int	float
Nome	divisor	dividendo	resultado
Valor	? 10	?	?

Simulação Manual

- Execução em memória:
 - Ex. Suponha o código:
 1. `int divisor,dividendo;`
 2. `float resultado;`
 3. `divisor=10;`
 4. `dividendo=13;` ← Último executado
 5. `resultado = dividendo / divisor;` ← Próximo Comando
- Após “executar” a linha 4

Tipo	int	int	float
Nome	divisor	dividendo	resultado
Valor	?	?	?
	10	13	

Simulação Manual

- Execução em memória:
 - Ex. Suponha o código:
 1. `int divisor,dividendo;`
 2. `float resultado;`
 3. `divisor=10;`
 4. `dividendo=13;`
 5. `resultado = dividendo / divisor;` ← Último executado
- Após “executar” a linha 5

Tipo	int	int	float
Nome	divisor	dividendo	resultado
Valor	?	?	?
	10	13	1.0

Simulação Manual

- Execução em memória:
 - Ex. Suponha o código:
 1. `int divisor,dividendo;`
 2. `float resultado;`
 3. `divisor=10;`
 4. `dividendo=13;`
 5. `resultado = dividendo / divisor;` ← Último executado
- Término da execução (não há mais comandos)

Tipo	int	int	float
Nome	divisor	dividendo	resultado
Valor	?	?	?
	10	13	1.0

Simulação Manual

- Execução em memória:
 - Ex. Suponha o código (corrigido):
 1. `int divisor,dividendo;`
 2. `float resultado;`
 3. `divisor=10;`
 4. `dividendo=13;`
 5. `resultado = (float)dividendo / (float)divisor;`
- Execução completa

Tipo	int	int	float
Nome	divisor	dividendo	resultado
Valor	?	?	?
	10	13	1.3

Decisão simples e decisão múltipla

- Dependendo do problema proposto, o programa pode ser formado por um conjunto muito grande de comandos `if` e expressões lógicas.

Exemplo

Faça um programa que, dado um RA, emite uma mensagem se o aluno estiver matriculado em uma turma de MC102.

Decisão simples

Para apenas um aluno, a solução seria:

```
main () {  
    int a;  
    scanf("%d", &a);  
    if (a == 10129) {  
        printf("O aluno %d está matriculado\n", a);  
    }  
}
```

Decisão múltipla

Para dois alunos, a solução seria:

```
main () {  
    int a;  
    scanf("%d", &a);  
    if (a == 10129 || a == 16267) {  
        printf("O aluno %d está matriculado\n", a);  
    }  
}
```

Decisão múltipla

- Problema: cada turma de MC102 possui cerca de 60 alunos e temos 14 turmas neste semestre.

```
if (a == 2582 || a == 10129 ||  
    a == 16267 || ...  
    a = 962185) {  
    printf("O aluno %d está matriculado\n", a);  
}
```

- Teríamos muitas condições a serem testadas.

Decisão simples e decisão múltipla

Exemplo 2

- Faça um programa que, dado um RA, mostre o nome desse aluno.

Decisão simples

Para apenas um aluno, a solução seria:

```
main () {  
    int a;  
    scanf("%d", &a);  
    if (a == 10129) {  
        printf("Maria Cândida Moreira Telles\n");  
    }  
}
```

Decisão múltipla

```
main () {  
    int a;  
    scanf("%d", &a);  
    if (a == 10129)  
        printf("Maria Cândida Moreira Telles\n");  
    if (a == 33860)  
        printf("Larissa Garcia Alfonsi\n");  
    if (a == 33967)  
        printf("Leonardo Kozlowiski Kenupp\n");  
}
```

Decisão múltipla

- Novamente, temos um conjunto muito grande de alunos.
- Além disso, não podemos utilizar os operadores lógicos que utilizamos anteriormente.
- Podemos tentar diminuir o número de testes realizados?
- Uma construção bem comum é o uso da seqüência `if else if`:

```
if (<condição1>
    <comando>
else if (<condição2>)
    <comando>
...
else if (<condiçãoN>)
    <comando>
```

O comando `switch`

- O objetivo do comando `switch` é simplificar uma expressão onde uma variável **inteira** ou **caracter** deve fazer diferentes operações dependendo exclusivamente de seu valor.

Sintaxe

```
switch (variável inteira) {  
    case valor: comandos  
    break;  
    case valor: comandos  
    break;  
}
```

O comando `switch`

```
switch(a) {  
  case 10129:  
    printf("Maria Cândida Moreira Telles\n");  
    break;  
  case 33860:  
    printf("Larissa Garcia Alfonsi\n");  
    break;  
  case 33967:  
    printf("Leonardo Kozlowiski Kenupp\n");  
    break;  
}
```

O comando `switch`

- Os comandos começam a ser executados a partir do ponto onde o valor da variável corresponde ao valor antes dos dois pontos (:).
- Executa todos os comandos até que encontre um comando `break` ou que chegue ao final do bloco de comandos do `switch`

Valor padrão

- Você pode utilizar, ao invés de um valor, o valor `default`. A execução dos comandos inicia no comando `default` se nenhum outro valor for correspondente ao valor da variável.

Sintaxe

```
switch (variável inteira) {  
    valor:  comandos break;  
    default: comandos  
}
```

Valor padrão

```
switch(a) {  
case 10129:  
    printf("Maria Cândida Moreira Telles\n");  
    break;  
case 33860:  
    printf("Larissa Garcia Alfonsi\n");  
    break;  
default:  
    printf("O aluno não está matriculado\n");  
}
```

Exercícios

- Dada uma letra, escreva na tela se essa letra é ou não uma vogal (pode considerar apenas letras minúsculas).
- Dado um caracter, escreva na tela se esse caracter é uma letra minúscula.

Exercícios

- Escreva um programa que recebe um operando, um operador aritmético e outro operando e calcule a operação indicada. As operações possíveis são soma(+), subtração(-), multiplicação(*) e divisão(/).

Exemplo

10 + 2

imprime 12 na tela.

Exercícios

- Escreva um programa que mostre na tela um *menu* de pratos (pelo menos 5), cada um associado a um número.

Prato 1 - Miojo

Prato 2 - Ensopado

- Quando um número é selecionado, o programa deve exibir uma breve descrição do prato. Por exemplo, ao digitar 1, o programa mostra: “Macarrão instantâneo”