9002 — Aula 06
Algoritmos e Programação de Computadores

Instituto de Engenharia – UFMT

Segundo Semestre de 2014

07 de outubro de 2014

Roteiro

Na aula passada...

- Vimos como criar algoritmos que realizam decisões...
- Vimos também como definir **expressões relacionais** e **comandos condicionais** simples e encaixados...

Expressões lógicas

Expressões lógicas são aquelas que realizam uma operação lógica (`ou`, `e`, `não`, etc...) e retornam verdadeiro ou falso (como as expressões relacionais).

Operadores Lógicos

Os operadores lógicos são

- && : operador E
- || : operador OU
- ! : operador NÃO

Expressões lógicas

- <expressao> && <expressao>: Retorna verdadeiro quando ambas as expressões são verdadeiras. Sua tabela verdade é:

Op_1	Op_2	Ret
V	V	V
V	F	F
F	V	F
F	F	F

Exemplo

```
a == 0 && b == 0
```

Expressões lógicas

- <expressao> || <expressao>: Retorna verdadeiro quando pelo menos uma das expressões é verdadeiras. Sua tabela verdade é:

Op_1	Op_2	Ret
V	V	V
V	F	V
F	V	V
F	F	F

Exemplo

```
a == 0 || b == 0
```

Expressões lógicas

- ! <expressao>: Retorna verdadeiro quando a expressão é falsa e vice-versa. Sua tabela verdade é:

Op_1	Ret
V	F
F	V

Exemplo

```
!(a == 0)
```

Simplificações úteis

Equivalências

- `!(a == b)` é equivalente a `a != b`
- `!(a != b)` é equivalente a `a == b`
- `!(a > b)` é equivalente a `a <= b`
- `!(a < b)` é equivalente a `a >= b`
- `!(a >= b)` é equivalente a `a < b`
- `!(a <= b)` é equivalente a `a > b`

Combinando comandos condicionais

Às vezes podemos simplificar

```
if (cond1) {
  if (cond2) {
    if (!cond3) {
      comando;
    }
  }
}
```

por

```
if (cond1 && cond2 && !cond3) {
  comando;
}
```

if-else-if Encaixados

- Várias vezes a construção if-else-if alterna entre valores inteiros:

```
#include <stdio.h>
int main () {
    long rga;
    scanf("%ld", &rga);
    if (rga == 201421901004)
        printf("Abel Breno Ventura de Abreu\n");
    else if (rga == 201421904008)
        printf("Ana Flavia Rosa Manzano\n");
    else printf("Nenhum aluno com RGA informado!");

    return 0;
}
```

O comando `switch`

- O objetivo do comando `switch` é simplificar uma expressão onde as condições ocorrem sobre uma variável inteira (pode ser caractere):

Sintaxe

```
switch (variável inteira) {
    case valor1:
        comandos
        break;
    case valor2:
        comandos
        break;
}
```

O comando switch

```
switch(rga) {
   case 201421901004:
      printf("Abel Breno Ventura de Abreu\n");
      break;
   case 201421904008:
       printf("Ana Flavia Rosa Manzano\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 uma condição `default`. A execução dentro da alternativa `default` ocorre se nenhuma outra condição foi verdadeira (assim como o último else do if-else-if encaixados).

Sintaxe do `switch`

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

Valor padrão

```
 switch(rga) {
  case 201421901004:
     printf("Abel Breno Ventura de Abreu\n");
     break;
  case 201421904008:
      printf("Ana Flavia Rosa Manzano\n");
      break;
    default:
        printf("O aluno não está matriculado\n");
}
```

Legibilidade de código

Legibilidade

É a qualidade do código que permite o fácil entendimento por seres humanos (programadores).

Importância

- Um código é escrito uma vez e lido várias vezes.
- Várias pessoas podem ler e entender um mesmo programa.
- É mais fácil fazer alterações futuras ou aproveitar em outros projetos.

Legibilidade de código

Para um código ser legível, ele deve ter as seguintes características.

Características

- Comentários que resumem o que faz um conjunto de comandos.
- Formatação bem definida (indentação, espaçamentos, nomes)
- Documentação adequada de cada programa ou algoritmo.

Um pouco de maniqueísmo - Comentários

Mal

```
#include <stdio.h>
#include <math.h>
int main() {
    // declara três variáveis
    float x, y, z;
    // lê um valor do teclado
    scanf("%f", &x);
    // lê outro valor
    scanf("%f", &y);
    // calcula o x ao quadrado
    // mais y ao quadrado
    z = x*x + y*y;
    // tira a raiz de z
    z = sqrt(z);
    // mostra o valor de z
    printf("%f\n", z);
    return 0;
}
```

Um pouco de maniqueísmo - Comentários

Mal

```
#include <stdio.h>
#include <math.h>
int main() {
    // declara três variáveis
    float x, y, z;
    // lê um valor do teclado
    scanf("%f", &x);
    // lê outro valor
    scanf("%f", &y);
    // calcula o x ao quadrado
    // mais y ao quadrado
    z = x*x + y*y;
    // tira a raiz de z
    z = sqrt(z);
    // mostra o valor de z
    printf("%f\n", z);
    return 0;
}
```

Bem

```
#include <stdio.h>
#include <math.h>
int main() {
    // catetos e hipotenusa
    float x, y, z;
    // lê valores dos catetos
    scanf("%f", &x);
    scanf("%f", &y);

    // aplica Pitágoras
    z = x*x + y*y;
    z = sqrt(z);

    // devolve a hipotenusa
    printf("%f\n", z);
    return 0;
}
```

Um pouco de maniqueísmo - Indentação

Mal

```
#include <stdio.h>
int main() {
int n;

printf("n: "); scanf("%d",&n);

if(n==1){printf("Unidade");}

else if (n>=0) {
if (n%3)
  printf("Deixa resto\n");
else   printf("Não deixa");

}else{printf("Negativo");}}
```

Um pouco de maniqueísmo - Indentação

Mal

```
#include <stdio.h>
int main() {
int n;

printf("n: "); scanf("%d",&n);

if(n==1){printf("Unidade");}

else if (n>=0) {
if (n%3)
  printf("Deixa resto\n");
else   printf("Não deixa");

}else{printf("Negativo");}}
```

Bem

```
#include <stdio.h>
int main() {
    int n;
    printf("n: ");
    scanf("%d",&n);

    if (n==1) {
        printf("Unidade");
    } else if (n>=0) {
        if (n%3)
            printf("Deixa resto");
        else
            printf("Não deixa");
    } else {
         printf("Negativo");
    }
    return 0;
}
```

Um pouco de maniqueísmo - Nome de variáveis

Mal

```
#include <stdio.h>
int main() {
  float n1, n2, n3, n4, aluno;
  int exercices;

  scanf("%d %f%f",&exercices,
      &n1, &n3);
  n2 = 4.0;
  n4 = 6.0;
  if (exercices > 7)
    n4 = 7.0;
  aluno = (n1*n2 + n3*n4)/10;
  printf("%f", aluno);
}
```

Um pouco de maniqueísmo - Nome de variáveis

Mal

```
#include <stdio.h>
int main() {
  float n1, n2, n3, n4, aluno;
  int exercices;

  scanf("%d %f%f",&exercices,
      &n1, &n3);
  n2 = 4.0;
  n4 = 6.0;
  if (exercices > 7)
    n4 = 7.0;
  aluno = (n1*n2 + n3*n4)/10;
  printf("%f", aluno);
}
```

Bem

```
#include <stdio.h>
int main() {
  float nota1, nota2;
  float peso1, peso2;
  float media;
  int exercicios;

  scanf("%d %f%f", &exercicios,
      &nota1, &nota2);
  peso1 = 4.0;
  peso2 = 6.0;
  if (exercicios > 7)
    peso2 = 7.0;
  media = (nota1*peso1 +
           nota2*peso2)/10;
  printf("%f", media);
  return 0;
}
```

Um pouco de maniqueísmo - Documentação

Mal

```
#include <stdio.h>
int main() {
  float r;
  float v;
  scanf("%f", &r);
  v = (4.0*3.1425*r*r*r)/3.0;
  printf("%f", v);
}
```

Um pouco de maniqueísmo - Documentação

Mal

```
#include <stdio.h>
int main() {
  float r;
  float v;
  scanf("%f", &r);
  v = (4.0*3.1425*r*r*r)/3.0;
  printf("%f", v);
}
```

O que esse programa faz???

Um pouco de maniqueísmo - Documentação

Mal

```
#include <stdio.h>
int main() {
  float r;
  float v;
  scanf("%f", &r);
  v = (4.0*3.1425*r*r*r)/3.0;
  printf("%f", v);
}
```

O que esse programa faz???
O valor de pi está errado, com quem eu falo???

Um pouco de maniqueísmo - Documentação

Mal

```
#include <stdio.h>
int main() {
  float r;
  float v;
  scanf("%f", &r);
  v = (4.0*3.1425*r*r*r)/3.0;
  printf("%f", v);
}
```

O que esse programa faz???
O valor de pi está errado, com quem eu falo???

Bem

```
/* DESCRIÇÃO: Lê o raio de
      uma esfera pelo teclado
      e imprime o seu volume.
   ENTRADA: o raio da esfera
   SAÍDA: o volume da esfera
   PRÉ-CONDIÇÃO: O raio deve
      ser maior que zero.
   AUTOR: Nome do Aluno */
#include <stdio.h>
int main() {
  float r;
  float v;
  scanf("%f", &r);
  v = (4.0*3.1415*r*r*r)/3.0;
  printf("%f", v);
  return 0;
}
```

Exercícios

1. Refazer os exercícios da aula anterior. Dessa vez, utilize operadores lógicos para simplificar o código.
2. Escreva um programa que, dadas duas datas, determine qual delas ocorreu cronologicamente antes. Cada data é composta por 3 números inteiros: o ano, o mês e o dia.
3. Troque seus códigos com algum colega de sala. Corrija o código do seu colega e verifique se ele é legível. Atente-se para comentários, indentação, etc.

Na Próxima aula...

- colocaremos a mão na massa mais um pouco :)
- FIM.