MC202 - Estruturas de Dados

Alexandre Xavier Falcão

Instituto de Computação - UNICAMP

afalcao@ic.unicamp.br

Desvio e Repetição

- Toda linguagem possui comandos de desvio e de repetição de instruções.

Desvio e Repetição

- Toda linguagem possui comandos de desvio e de repetição de instruções.

- Em C, comandos de desvio são **if**, **switch**, **goto**, **break**, e **continue**, mas vamos evitar os três últimos.

Desvio e Repetição

- Toda linguagem possui comandos de desvio e de repetição de instruções.
- Em C, comandos de desvio são **if**, **switch**, **goto**, **break**, e **continue**, mas vamos evitar os três últimos.
- Comandos de repetição são **while**, **for**, e o **do**...**while**.

Desvio e Repetição

- Toda linguagem possui comandos de desvio e de repetição de instruções.
- Em C, comandos de desvio são **if**, **switch**, **goto**, **break**, e **continue**, mas vamos evitar os três últimos.
- Comandos de repetição são **while**, **for**, e o **do**...**while**.
- Vamos aproveitar para introduzir os operadores lógicos && (and), || (or), e ! (not).

Agenda

- Comando **if**.
- Comando **switch**.
- Comando **do. . .while**.
- Comando **while**.
- Comando **for**.

Comando **if**

O comando **if** pode ser simples ou composto.

- Simples:

```
if(expressão lógica){
        bloco de comandos
}
```

Comando **if**

O comando **if** pode ser simples ou composto.

- Simples:

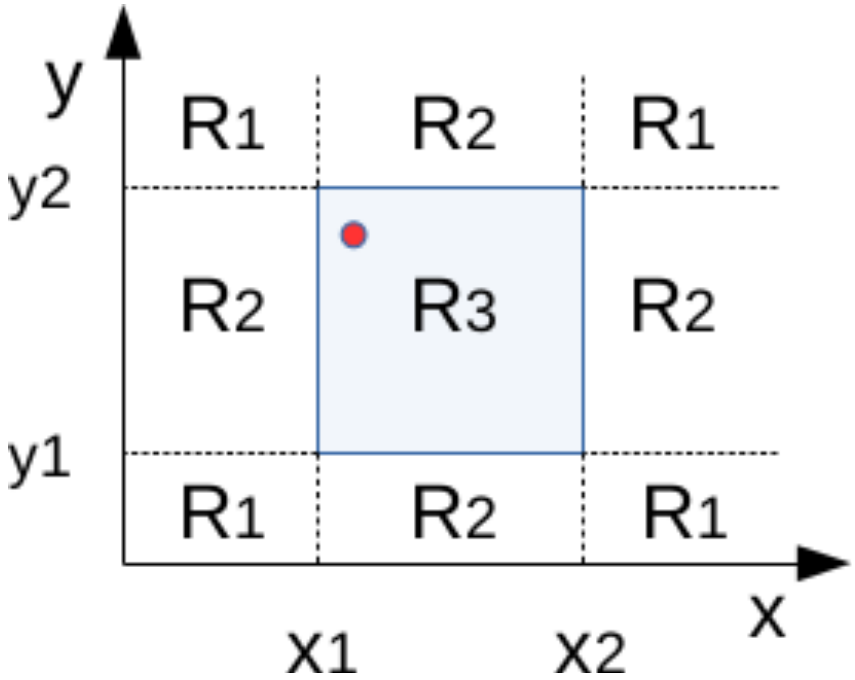
```
if(expressão lógica){
        bloco de comandos
}
```

- Composto:

```
if(expressão lógica){
        bloco de comandos
} else {
        bloco de comandos
}
```

Exemplo

Sejam R_1, R_2, e R_3 três regiões do $\Re^2$. Dado um ponto $(x, y) \in \Re^2$ qualquer, escreva o escopo de uma função que recebe x_1, x_2, y_1, y_2, x, y, e retorna o subescrito $r \in \{1, 2, 3\}$ da região que contém (x, y).



Solução

```
int Regiao(float x1, float x2, float y1, float y2, float x, float y)
{
  if ((x1 <= x) && (x <= x2) && (y1 <= y) && (y <= y2)){
    return(3);
  } else {
    if (((x < x1)  && (y1 <= y) && (y <= y2)) ||
        ((x > x2)  && (y1 <= y) && (y <= y2)) ||
        ((y < y1)  && (x1 <= x) && (x <= x2)) ||
        ((y > y2)  && (x1 <= x) && (x <= x2)) ) {
      return(2);
    } else { /* R1 */
      return(1);
    }
  }
}
```

Comando **switch**

O comando **switch** evita o uso de vários comandos **if** para verificar o conteúdo alfanumérico de uma variável.

```
switch(variável) {
case conteúdo1:
        bloco de comandos
        break;
case conteúdo2:
        bloco de comandos
        break;
⋮
case conteúdoN:
        bloco de comandos
        break;
default:
        bloco de comandos
}
```

Exemplo

Considere uma calculadora de operações simples, $*, +, -, /$, entre dois números x e y. Faça uma função que recebe y, y e a operação o desejada, retornando o resultado.

Solução

```
float Operacao(float x, float y, char o)
{
  switch(o) {
  case '*':
    return(x*y);
  case '/':
    if ((-0.00001 < y) && (y < 0.00001)){
      printf("Divisão indefinida\n"); exit(1);
    } else {
      return(x/y);
    }
  case '+':
    return(x+y);
  case '-':
    return(x-y);
  default:
    printf("Operação não implementada"); exit(2);
  }
}
```

Comando **do**. . .**while**

O comando **do**. . .**while** é uma estrutura de repetição onde a condição de parada é testada após o bloco de comandos.

do {
    bloco de comandos
} **while**(expressão lógica);

Comando **do**. . .**while**

O comando **do**. . .**while** é uma estrutura de repetição onde a condição de parada é testada após o bloco de comandos.

do {
    bloco de comandos
} **while**(expressão lógica);

Um exemplo é o algoritmo de Euclides para calcular o máximo divisor comum (MDC) entre dois números inteiros positivos.

MDC entre dois inteiros positivos

Entrada: Inteiros positivos m e n.
Saída: máximo divisor comum x.

1. $x \leftarrow m$, $y \leftarrow n$.
2. **do** $\{$
3. $\quad r \leftarrow x\%y$.
4. $\quad x \leftarrow y$, $y \leftarrow r$.
5. $\}$ **while**$(r \neq 0)$;
6. return(x);

Solução

```
int MDC(int m, int n)
{
  if ((m <= 0)||(n <= 0)){
    printf("valores inválidos\n");
    exit(1);
  }

  int x = m, y = n, r;
  do {
    r = x % y;
    x = y; y = r;
  } while (r != 0);

  return(x);
}
```

Comando **while**

O comando **while** é uma estrutura de repetição onde (1) uma variável de controle é inicializada, (2) uma condição de parada envolvendo esta variável é verificada, e (3) a variável de controle é atualizada, retomando à verificação.

```
inicialização
while(expressão lógica) {
    bloco de comandos
    atualização
}
```

Exemplo

- Sabemos que um número natural primo possui apenas dois divisores positivos e distintos (i.e., 1 não é primo).

Exemplo

- Sabemos que um número natural primo possui apenas dois divisores positivos e distintos (i.e., 1 não é primo).
- Sabemos que um número x só pode ser divisível por números menores do que $\frac{x}{2}$.

Exemplo

- Sabemos que um número natural primo possui apenas dois divisores positivos e distintos (i.e., 1 não é primo).
- Sabemos que um número x só pode ser divisível por números menores do que $\frac{x}{2}$.
- Então se x tiver um único dividor entre $[2, \lfloor \frac{x}{2} \rfloor]$, ele não é primo. Na verdade, basta verificar o intervalo $[2, \lfloor \sqrt{x} \rfloor]$.

Exemplo

- Sabemos que um número natural primo possui apenas dois divisores positivos e distintos (i.e., 1 não é primo).
- Sabemos que um número x só pode ser divisível por números menores do que $\frac{x}{2}$.
- Então se x tiver um único dividor entre $[2, \lfloor \frac{x}{2} \rfloor]$, ele não é primo. Na verdade, basta verificar o intervalo $[2, \lfloor \sqrt{x} \rfloor]$.
- Usando o comando **while** apenas, escreva uma função que imprime todos os números naturais positivos que são primos menores ou iguais a um dado número N.

Solução

```c
char Primo(int x)
{
  int  y = (int)sqrt(x);
  while (y >= 2){
    if ((x % y)==0)
      return(0); /* false */
    y--;
  }
  return(1); /* true */
}

void PrimosMenoresOuIguais(int N)
{
  if (N < 2){
    printf("Número inválido\n"); exit(1);
  }

  int x = N;
  while (x > 2) {
    if (Primo(x)) {
      printf("%d, ",x);
    }
    x = x - 1;
  }
  printf("2.\n");
}
```

Comando **for**

O comando **for** é uma simplificação do **while**, onde inicialização, controle de parada, e atualização são especificados no próprio comando.

for (inicialização; expressão lógica; atualização) {
    bloco de comandos
}

Comando **for**

O comando **for** é uma simplificação do **while**, onde inicialização, controle de parada, e atualização são especificados no próprio comando.

for (inicialização; expressão lógica; atualização) {
    bloco de comandos
}

Substitua **while** por **for** na implementação anterior.

Solução

```
char Primo(int x)
{
  for (int  y = (int)sqrt(x); (y >= 2); y--){
    if ((x % y)==0)
      return(0); /* false */
  }
  return(1); /* true */
}

void PrimosMenoresOuIguais(int N)
{
  if (N < 2){
    printf("Número inválido\n"); exit(1);
  }

  for (int x = N; (x > 2); x=x-1){
    if (Primo(x)) {
      printf("%d, ",x);
    }
  }
  printf("2.\n");
}
```

Exercícios

Escreva funções em C para calcular

- a integral $\int_{x_1}^{x_2} \exp\left(\frac{-(x-\mu)^2}{2\sigma^2}\right) dx$ para valores dados de $x_1 \leq x_2$, μ e σ,
- a integral dupla $\int_{y_1}^{y_2} \int_{x_1}^{x_2} xy^2 dxdy$ para valores dados de $x_1 \leq x_2$ e $y_1 \leq y_2$, e
- o valor máximo de um polinômio de grau 2, $Ax^2 + By^2 + 2Cxy + Dx + Ey + F$, para valores $x_1 \leq x_2$, $y_1 \leq y_2$, A, B, C, D, E, e F dados, retornando também o ponto (x_0, y_0) onde ocorre este máximo.