Algoritmos - TADS

Algoritmos – Estruturas de controle em C

Professor: Victor Hugo L Lopes

Agenda

- Seleção
- Iteração
- Bloco
- Laços

Programação em Linguagem C

- Seleção simples e composta:

Definem fluxo condicional de execução do algoritmo.

Sintaxe:

if (expressão lógica) comando;

else comando;

```
        if (expressão lógica) {
A.              comandos
B.      }else{
C.              comandos
D.      }
```

Programação em Linguagem C

- Seleção simples e composta:

```
void main(void)
{
  int magic;   /* número mágico */
  int guess;   /* palpite do usuário */

  magic = rand();  /* gera o número mágico */

  printf("adivinhe o número mágico: ");
  scanf("%d", &guess);

  if(guess == magic) printf("** Certo **");
  else printf("Errado");
}
```

Programação em Linguagem C

- Ifs aninhados:

Um if aninhado é um if que é objeto de outro if ou else. Em C, um comando else sempre se refere ao comando if mais próximo. Exemplo:

```
if (i){
    if (j) comando1;
    if (k) comando2; //esse if
    else comando3;  //está associado a este else
} else comando4; //este else está associado a if (i)
```

Programação em Linguagem C

```
void main(void)
{
  int magic;  /* número mágico */
  int guess;  /* palpite do usuário */

  magic = rand();  /* gera o número mágico */

  printf("Adivinhe o número mágico: ");
  scanf("%d", &guess);

  if(guess == magic) {
    printf("** Certo **");
    printf(" %d é o número mágico\n", magic);
  }
  else {
    printf("Errado, ");
    if(guess > magic)  printf("muito alto\n");
    else printf("muito baixo\n");
  }
}
```

Programação em Linguagem C

- ? como alternativa à if/else:

```
int x = 10;
if (x>9) y = 100;
else y = 200;
```

```
int x = 10;
y = x>9  ?   100   :   200;
```

Programação em Linguagem C

- Operador ternário aninhado???:

```
int x = 10;
if (x>9) {
      if (x==10) y=100;
      else y=10;
}else y = 200;
```

```
int x = 11;
   int y;
   y = x>9  ?  (x==10 ? 100 : 10)  :  200;
```

Programação em Linguagem C

Múltiplas seleções com switch:

A linguagem C tem um comando de seleção múltipla, o switch, que testa sucessivamente o valor de uma expressão contra uma lista de constantes inteiras ou de caractere. Quando o valor coincide, os comandos associados à constante são executados. Sintaxe:

```
swtch(expressão){
        case constante1:
                comandos;
                break;
        case constante2:
                comandos;
                break
        ...
        default:
                comandos;
}
```

```
char ch;

printf("1. Checar Ortografia\n");
printf("2. Corrigir Erros de Ortografia\n");
printf("3. Mostrar Erros de Ortografia\n");
printf("Pressione Qualquer Outra Tecla para Abandonar\n");
printf("     Entre com sua escolha:   ");

ch=getchar();  /* Lê do teclado a seleção */

switch(ch) {
  case '1':
    check_spelling();
    break;
  case '2':
    correct_errors();
    break;
  case '3':
    display_errors();
    break;
  default :
    printf("Nenhuma opção selecionada");
  }
```

```
int flag;

flag = -1;

switch(i) {
  case 1:  /* Estes cases têm uma seqüência */
  case 2:  /* de comandos em comum */
  case 3:
    flag = 0;
    break;
  case 4:
    flag = 1;
  case 5:
    error(flag);
    break;
  default:
    process(i);
}
```

Programação em Linguagem C

Switch aninhado???

```
switch(x) {
   case 1:
     switch(y) {
       case 0: printf ("erro de divisão por zero");
               break;
       case 1: process(x,y);
     }
     break;
   case 2:
    .
    .
    .
```

Programação em Linguagem C

Laços de repetição:

Relembrando que um laço de repetição garante a repetição de comandos sob determinadas condições.

O laço for:

sintaxe:

for (inicialização; condição; incremento) comando;

O laço for permite muitas variações, mas é comum ter a inicialização com um comando de atribuição, que coloca um valor inicial para o passo. A condição é uma expressão condicional que garante o início e fim do laço, e incremento garante o passo.

Programação em Linguagem C

Laços de repetição:

```
#include <stdio.h>

void main(void)
{
   int x;

   for(x=1; x <= 100; x++) printf("%d ",x);
}
```

Programação em Linguagem C

Laços de repetição:

Se existirem múltiplos comandos, um bloco deve ser criado.

```
x = 10;
for(y=10; y!=x; ++y) printf("%d", y);
printf("%d", y); /* este é o único comando
                    printf() que executará */


for(x=100; x != 65; x-=5) {
   z = x*x;
   printf("O quadrado de %d, %f",x, z);
}
```

Programação em Linguagem C

Laços de repetição:

Múltiplas variáveis de controle:

```
for(x=0, y=0; x+y<10; ++x) {
  y = getchar();
  y = y-'0'; /* subtrai o código ASCII do caractere 0 de y */
     .
     .
     .
}
```

Programação em Linguagem C

Laços de repetição- o laço While:

É um laço com teste no início.

Outro laço disponível em C é o laço while, com a sintaxe:

while (condição) comando;

```
char ch;
ch = '';
while (ch != 'A') ch = getchar();

while ((ch=getchar()) != 'A'); //while sem corpo
```

Programação em Linguagem C

Laços de repetição- o laço While:

While com múltiplos comandos utiliza-se um bloco de comandos:

```
int working;

working = 1;   /* i.e., verdadeiro */

while(working) {
   working = process1();
   if(working)
     working = process2();
   if(working)
     working = process3();
}
```

Programação em Linguagem C

Laços de repetição- o laço do-While:

Com teste no fim.

Ao contrário dos laços for e while, que testam no início, o laço do-while verifica a condição ao final do laço. Isso significa que um laço do-while sempre será executado ao menos uma vez.

- Sintaxe:

```
do{
        comando;
}while(condição)
```

```
do {
   scanf("%d", &num);
} while(num > 100);
```

```
void menu(void)
{
  char ch;

  printf("1. Verificar Ortografia\n");
  printf("2. Corrigir Erros de Ortografia\n");
  printf("3. Mostrar Erros de Ortografia\n");
  printf("      Digite sua escolha:   ");

  do {
    ch = getchar();  /* lê do teclado a seleção */
    switch(ch) {
      case '1':
        check_spelling();
        break;

        case '2':
          correct_errors();
          break;
        case '3':
          display_errors();
          break;
      }
    } while(ch!='1' && ch!='2' && ch!='3');
  }
```