



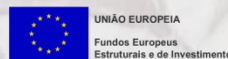
ESCOLA PROFISSIONAL CRISTÓVÃO COLOMBO

www.epcc.pt



Sistemas Operativos – Processamento Computacional

Estruturas de Controlo



Autor da apresentação

09/09/2014

Estruturas de Controlo

As estruturas de controlo são a essência de qualquer linguagem de programação uma vez que determinam a sequência pela qual as instruções de um programa são executadas.

Instrução Simples e Compostas

Na linguagem C o ponto e vírgula ; é o terminador de instruções.

As chavetas { } são usadas para agrupar instruções em instruções compostas ou blocos de modo a serem sintacticamente equivalentes a uma instrução única.

Assim uma instrução em sentido genérico pode ser:

- Uma única instrução.
- Um conjunto de instruções.
- Nenhuma instrução (instrução vazia).

Estrutura compostas ou Bloco

Uma instrução composta ou bloco tem a estrutura:

```
{  
declarações (opcional)  
Instruções  
}
```


Estruturas de Decisão - *if*

Testa uma condição, que é uma expressão que é avaliada. Se for verdadeira (tiver um valor não nulo) é executada uma instrução, se for falsa e existir o *else* então é executada a instrução para o senão.

Cada uma destas instruções pode ser na realidade um bloco delimitado por chavetas.

Nota: o *else* é opcional

Estruturas de Decisão – *if* variações

```
if(condição)  
    instrução;
```

```
if(condição)  
    instrução;  
else  
    instrução;
```

```
if(condição){  
    instrução 1;  
    instrução 2;  
}
```

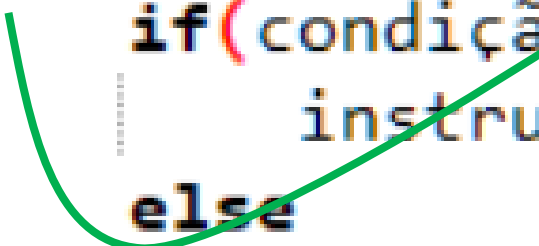
```
if(condição){  
    instrução 1;  
}else{  
    instrução 2;  
}
```

Estruturas de Decisão – *if encadeados*

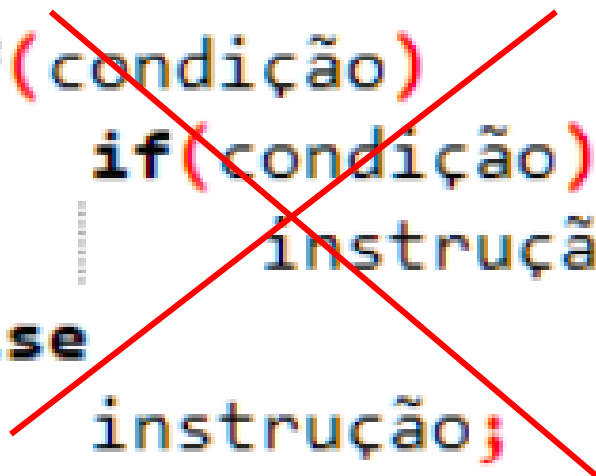
Imaginemos o seguinte código como exemplo:

```
if(condição)
if(condição)
    instrução;
else
    instrução;
```

```
if(condição)
    if(condição)
        instrução;
    else
        instrução;
```



```
if(condição)
    if(condição)
        instrução;
    else
        instrução;
```



Estruturas de Decisão - *if encadeados*

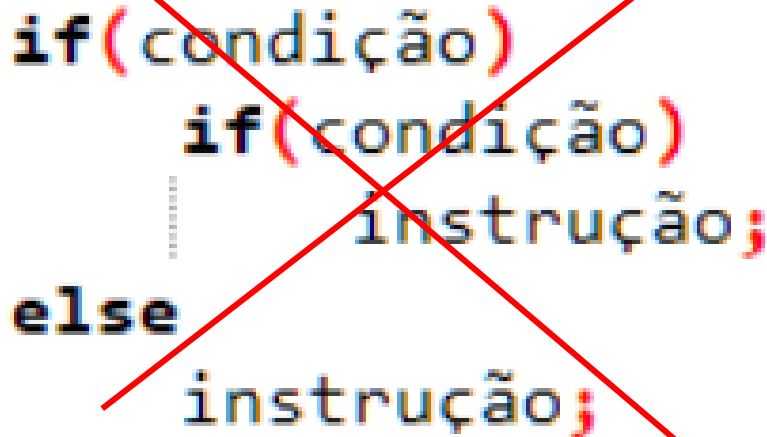
Explicação:

A duas interpretações foram mostradas usando indentações diferentes, no entanto o compilador de C é insensível a indentações. A linguagem C tem uma regra simples que permite resolver esta ambiguidade . Um *else* pertence ao *if* mais interior que ainda não tem associado um *else* .

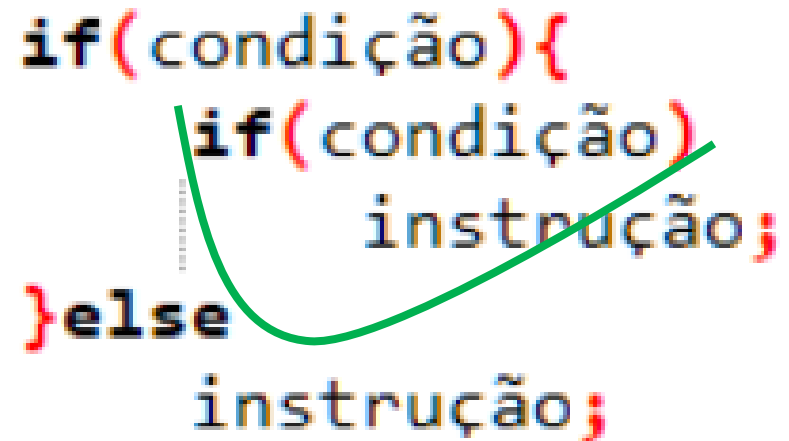
Estruturas de Decisão - *if encadeados*

Explicação:

Se se pretender a segunda interpretação terão de ser utilizadas chavetas:



```
if(condição)
    if(condição)
        ...
        instrução;
else
    instrução;
```



```
if(condição){
    if(condição)
        instrução;
}else
    instrução;
```

Estruturas de Decisão - *if múltiplos*

Sem bloco

```
if(condição)
    instrução;
else if(condição)
    instrução;
else
    instrução;
```

Com bloco

```
if(condição){
    instrução;
}else if(condição){
    instrução;
}else{
    instrução;
}
```

Estruturas de Decisão – *if Exemplo*

```
int nm,i;
nm = 1 ;

printf("Introduza valor: ");
scanf("%d",&i);

if(i == nm){
    printf("Igual!!!\n");
    printf("%d é o nm:\n",nm);
}else if(i > nm){
    printf("Valor alto de mais.\n");
    printf("%d é o nm:\n",nm);
}else{
    printf("Valor baixo de mais.\n");
    printf("%d é o nm:\n",nm);
}
system("pause");
```

Estruturas de Decisão - *Switch*

Estrutura de escolha múltipla, o C fornece uma instrução específica para isso, a instrução *switch*. Nesta instrução é testada sucessivamente uma variável para verificar se coincide com uma lista de valores inteiros (ou caracteres).

Estruturas de Decisão – *Switch Syntaxe*

```
switch(variavel){  
    case exp1 : instrução1; break;  
    case exp2 : instrução2; break;  
    ...  
    default: instruçãoon; break;  
}
```

```
switch(variavel){  
    case exp1 :  
    case exp2 : instrução2; break;  
    ...  
    default: instruçãoon; break;  
}
```


Estruturas de Decisão – *Switch Exemplo*

```
int i;

printf("Introduza valor: ");
scanf("%d",&i);

switch(i){
    case 0 : printf("valor: 0"); break;
    case 1 : printf("valor: 1"); break;
    default: printf("Outro valor"); break;
}
```

Estruturas de Decisão – *Switch Exemplo 2*

```
char ch;
```

```
printf("Introduza caracter: ");  
scanf("%c",&ch);
```

```
switch(ch){  
    /* retorna o código ASCII do carater */  
    case 'A' : printf("valor: %d",ch); break; // 65  
    case 'b' : printf("valor: %d",ch); break;  
    default: printf("Outro valor"); break;  
}
```

Estruturas de Repetição - *while*

Estrutura de repetição ,os ciclos permitem ao computador executar uma sequência de instruções repetidamente até que deixe de ser satisfeita uma determinada condição.

Sintaxe

```
while(expressão)  
Instrução;
```

Estruturas de Decisão – *while* Exemplo

```
/* Ciclo que espera pela introdução do carater 'A'*/  
char c;  
/* uma string termina no primeiro caracter  
   nulo = todos bits 0,  
   ou terminador, ou "\0". */  
c = '\0';  
while(c != 'A')  
    scanf("%c",&c);
```

Estruturas de Repetição - *for*

Sintaxe

```
for(exp1; exp2;exp3)
```

```
Instrução;
```

Vs WHILE

```
Exp1;  
while(exp2){  
    Instrução  
Exp3
```

Estruturas de Repetição– *for Exemplo*

```
int x;  
for(x=1;x<=20;x++){  
    printf("%d\t",x);  
}
```

```
int x;  
for(x=10;x>0;x--){  
    printf("%d\t",x);  
}
```


Estruturas de Repetição– *for Exemplo*

```
int x;  
for(x=5;x<100;x = x+5){  
    printf("%d\t",x);  
}
```

Estruturas de Repetição – *do...while*

O ciclo *while* e o ciclo *for* testam a condição de ciclo no início do ciclo. O ciclo *do..while* testa a condição no fim do ciclo pelo que as instruções que fazem parte do corpo do ciclo são executadas pelo menos uma vez.

Sintaxe

```
do{
```

```
    Instrução;
```

```
}while(expressão);
```

Estruturas de Decisão – *do..while* Exemplo

```
/* Enquanto o valor digitado
   for superior a 100*/
int n;
do{
   scanf("%d",&n);
}while(n>100);
```

Estruturas de Decisão – *do..while* Exemplo

```
/* Enquanto o valor digitado  
for superior a 100*/  
int n;  
do{  
    scanf("%d",&n);  
}while(n>100);
```

Instruções de break

Por vezes é útil sair de um ciclo sem ser pelo teste no início ou no fim do ciclo. A instrução `break`, que já foi usada para sair da instrução `switch`, força a saída do ciclo `for`, `while` sem testar a condição de fim de ciclo.

Instruções de break - *Exemplo*

```
int t;  
for(t=0;t<15;t++){  
    printf("%d",t);  
    if(t == 10)  
        break;  
}
```


Instruções de break - *Exemplo abrangente*

```
/* A quebra é executada apenas no ciclo interno
   em cada quebra volta ao ciclo mais abrangente
   e executa até acontecer nova quebra
*/
int t, count;
for(t=0;t<2;t++){
    count = 1;
    //for em ciclo infinito
    for(;;){
        printf("%d",count);
        count++;
        if(count == 10)
            break;
    }
}
```

Instruções de continue

A instrução continue obriga a que o ciclo for, while ou do..while inicie um novo ciclo saltando qualquer código intermédio. Se se tratar de um ciclo while ou do..while a condição de teste é imediatamente executada, se for um ciclo for é executada imediatamente a expressão correspondente ao incremento. A instrução continue apenas se aplica a ciclos, não à instrução switch.

Instruções de continue- *Exemplo*

```
/* De cada vez que x é um número ímpar, a instrução condicional if  
é executada porque x%2 é 1, correspondendo ao valor lógico verdadeiro.  
Um número ímpar leva assim à execução do continue que provoca uma nova  
iteração do ciclo, passando por cima do printf.  
*/  
int x;  
for(x=0;x<100;x++){  
    if(x % 2) continue;  
    printf("%d \n",x);  
}
```