

1. Estruturas de controle

Normalmente, as instruções são executadas uma após a outra, na ordem em que foram escritas. Isto é chamado de **programação sequencial**. Várias instruções em C permitem que o programador especifique que a próxima instrução a ser executada seja diferente da próxima na sequência. Isto é chamado de **transferência de controle**. Foi demonstrado que todos os programas podiam ser escritos em termos de apenas três estruturas de controle, que eram: a **estrutura de sequência**, a **estrutura de seleção** e a **estrutura de repetição**.

A linguagem C fornece três tipos de estruturas de seleção: a estrutura de seleção **if**, a estrutura **if/else** e a estrutura **switch**. A linguagem C também fornece três tipos de estruturas de repetição, que são: **while**, **do/while** e o **for**.

2.A estrutura de seleção **if** e **if/else**

A estrutura de seleção **if** é usada para se fazer uma escolha entre várias linhas de ação alternativas. Sua instrução pode ser escrita em C como:

```
if (condição){  
    execução dentro do if;  
}
```

Caso a condição seja atendida, o fluxo do código entra dentro da estrutura **if** e realiza o que está contido dentro dela. Se não for atendida, segue executando o restante do código.

A estrutura de seleção **if** realiza uma ação indicada somente quando a condição for verdadeira, caso contrário, a ação é ignorada. Um conjunto de instruções dentro de um par de chaves é chamado de **instrução composta**. A estrutura de seleção **if/else** permite ao programador especificar que sejam realizadas ações diferentes conforme a condição seja verdadeira (primeiro programa) ou falsa (segundo programa).

```
if(true){  
    execução dentro do if;  
}else{  
    execução dentro do else;  
}  
  
if (!true){  
    execução dentro do if;  
}else{  
    execução dentro do else;  
}
```

Estruturas **if/else** aninhadas verificam vários casos inserindo uma estrutura **if/else** em outras, como no exemplo abaixo:

```
if(condição){
    execução dentro do if;
}else if(condição){
    execução dentro do else/if;
}else if(condição){
    execução dentro do else/if;
}else if(condição){
    execução dentro do else/if;
}else{
    execução dentro do else;
}
```

Os erros de sintaxe são acusados pelo compilador. Os erros lógicos produzem seus efeitos durante o tempo de execução. Um erro lógico fatal faz com que o programa falhe e termine prematuramente. Um erro lógico não fatal permite que o programa continue a ser executado, mas produz resultados incorretos.

3.A estrutura de repetição *while*

Uma estrutura de repetição permite ao programador especificar que uma ação deve ser repetida enquanto uma determinada condição for verdadeira. Essa ação será realizada repetidamente enquanto a condição permanecer verdadeira. A instrução contida na estrutura de repetição *while* constitui o corpo do *while* que pode ser uma instrução simples ou composta. Posteriormente, a condição se tornará falsa. Nesta ocasião, a repetição termina, e é executada a primeira instrução colocada após a estrutura de repetição.

```
while(condição){
    execução dentro do while;
}
```

A repetição controlada por controlador é usada para obter um grau de cada vez. Essa técnica usa uma variável chamada contador para especificar o número de vezes que um conjunto de instruções deve ser executado. A repetição controlada por controlador é denominada *repetição definida* porque o número de repetições é conhecido antes do loop começar a ser executado.

A repetição controlada por sentinela é chamada *repetição indefinida* porque o número de repetições não é conhecido antes do loop começar a ser executado. O valor sentinela deve ser escolhido de forma que não possa ser confundido com um valor aceitável de entrada.

4.A estrutura de repetição *for*

Em uma repetição controlada por um contador, uma variável de controle é usada para contar o número de repetições. A variável de controle é incrementada cada vez que o grupo de

instruções é realizado. Quando o valor da variável de controle indicar que o número correto de repetições foi realizado, o loop é encerrado e o computador continua a execução do programa a partir da instrução imediatamente após o loop. A repetição controlada por contador exige:

- O nome de uma **variável de controlador**
- O valor inicial da **variável de controle**
- O **incremento/decremento** pelo qual a variável de controle é modificada cada vez que o loop é realizado
- A **condição** que testa o valor final da variável de controle

```
for(expressão 1; expressão 2; expressão 3){  
    execução dentro do for;  
}
```

Quando a estrutura ***for*** começa a ser executada, a variável de controle ***contador*** é inicializada com um valor. A variável de controle contador é então incrementada ou decrementada e o loop começa novamente com seu teste de continuação. Como a variável de controle aumentou/diminuiu e o valor final não excedeu, o programa continua até que a variável de controle “contador” seja incrementada até o seu valor final. Isso faz com que o teste de continuação do loop não seja verdadeiro e a repetição termine.

A ***expressão 1*** inicia a variável de controle do loop. A ***expressão 2*** é a condição de continuação do loop e a ***expressão 3*** incrementa ou decrementa a variável de controle. O operador de vírgula é usado mais frequentemente em uma estrutura ***for***.

Sua utilidade principal é permitir que o programador use expressões múltiplas para inicialização e/ou incremento/decremento. As três expressões na estrutura ***for*** são opcionais. A expressão de incremento na estrutura “for” age como uma instrução independente do C no final do corpo do ***for***. Desta forma: contador = contador + 1; contador += 1; contador++ e ++contador são equivalentes na parte incremental da estrutura ***for***.

Dentro da estrutura, existem algumas observações a se fazer:

- A inicialização, a condição de continuação do loop e o incremento podem conter expressões aritméticas
- O incremento pode ser negativo
- Se a condição de continuação do loop ***for*** for inicialmente falsa, a parte do corpo do loop não é realizada. Desta forma, a execução prossegue com a instrução imediatamente após o fim da estrutura
- Frequentemente, a variável de controle é impressa ou usada em cálculos no corpo de um loop, mas não é exigido que isso aconteça

5.A estrutura de seleção múltipla ***switch/case***

A linguagem C fornece a estrutura de seleção múltipla para manipular tal tomada de decisão. Essa estrutura consiste em uma série de rótulos **case** e de um caso opcional **default**. A instrução **break** faz com que o controle do programa continue com a primeira instrução após a estrutura **switch**. A instrução é usada porque, caso contrário, os **cases** em uma instrução **switch** seriam todos executados. Cada **case** pode ter uma ou mais ações. Ao usar a estrutura **switch**, lembre-se que ela só pode ser usada para verificar uma *expressão constante inteira*. Qualquer combinação de constantes de caracteres e constantes inteiras que levam a um valor inteiro constante.

```
switch(value){  
    case 1:  
        execução dentro de 1;  
        break;  
    case N:  
        execução dentro de N;  
        break;  
    case default;  
        execução dentro de default;  
        break;  
}
```

Ao usar uma estrutura **switch**, ela só pode ser usada para verificar uma expressão constante inteira i.e., qualquer combinação de constantes de caracteres e constantes inteiras que levam a um valor inteiro constante.

6.A estrutura de repetição **do/while**

A estrutura de repetição **do/while** é similar à estrutura **while**. Na segunda, a condição de continuação é testada em seu início, antes do corpo da estrutura ser executada. Já na primeira, testa-se a condição de continuação depois do corpo do loop ser executado, portanto, ela será executada pelo menos uma vez. Quando um **do/while** termina, a execução continua com a instrução após a cláusula **while**.

```
do{  
    execução do do/while;  
}while(condição);
```

7.As instruções **break** e **continue**

Essas instruções são usadas para alterar o fluxo de controle. A instrução **break**, quando executada em uma estrutura **while**, **for**, **do/while** ou **switch**, faz com que aconteça a saída imediata daquela estrutura. A execução do programa continua com a primeira instrução depois da estrutura. Os usos mais comuns da instrução são para sair prematuramente de um loop, ou para saltar sobre o restante de uma estrutura de repetição **while**.

A instrução *continue* quando executada em uma estrutura *while*, *for*, ou *do/while*, ignora as instruções restantes no corpo daquela estrutura e realiza a próxima iteração do loop. Em estruturas *while* e *do/while*, o teste de continuação do loop é realizado imediatamente após a instrução *continue* ser executada. Na estrutura *for*, a expressão de incremento é executada e depois, o teste de continuação do loop é realizado.