ALGORITMOS E PROGRAMAÇÃO

Esp. Matheus de Melo Machado

INICIAR

introdução

Introdução

Na área da informática, no que se diz a lógica de programação, existem alguns tipos de estrutura, a estrutura que aceita realizar a execução mais de uma vez o mesma instrução ou conjunto de instruções, de combinação com uma condição ou com um contador. São usadas, por exemplo, para realizar a repetição de atos idênticos que são realizados para todos os elementos de uma lista de informação, ou de forma simples para realizar a repetição de um mesmo procedimento até que a condição seja contentada. Têm 3 estruturas de repetição fundamental para quase todas todas as linguagens de programação, seja C ou javascript .

Estrutura de Repetição: Conceitos de Contadores e Acumuladores

De acordo com Guedes (2014), para os comandos internos das estruturas de repetição existem dois tipos de uso de variáveis: os acumuladores e os contadores.

Quando a execução de um cálculo necessita de valores que sejam obtidos a cada iteração, ou quando o cálculo somente estará finalizado com a conclusão da repetição, utiliza-se os acumuladores. Para a operação em que será utilizado, um acumulador deve ser inicializado com um valor neutro (GUEDES, 2014).

saiba mais

Saiba mais

A linguagem C possui comandos para repetir uma sequência de instruções. Estas estruturas de repetição, também conhecidas como laços (do inglês *loops*). A principal construção que veremos é o while saiba mais em

Fonte: *Armando Luiz Nicolini Delgado* (2013).

ACESSAR

No caso de uma adição, o acumulador deve ser iniciado com o valor zero e, ser for uma multiplicação, o acumulador deve ser inicializado com o valor 1 (GUEDES, 2014). A variável acumulador é usada para acumular a soma uma série de valores e devem ser inicializadas com o valor zero antes de serem utilizadas, pois se isso não acontecer, a soma irá incluir o valor anterior armazenado na posição de memória da variável acumulador (DEITEL, 2011).

Exemplo:

Algoritmo Soma

```
#include <stdio.h>

int main()

{

int n, soma = 0, c, valor;

printf("Digite a quantidade de números para somar?\n");

scanf("%d", &n);
```

```
printf("Digite %d que serão somados\n", n);

for (c = 1; c <= n; c++){

scanf("%d", &valor);

soma = soma + valor;

}

printf("Soma dos numeros digitados = %d\n", soma);

return 0;

}
```

Já os contadores servem para executar a contagem de uma ação que ocorre dentro da estrutura de repetição (GUEDES, 2014). Essa técnica de utilizar um contador para controlar a estrutura de repetição serve então para especificar a quantidade de vezes que um conjunto de comandos será executado, e é conhecida como repetição definida, pois o número de repetições é conhecido antes que o loop comece a ser executado (DEITEL, 2011).

Essas variáveis contadoras devem ser inicializadas com os valores zero ou 1, dependendo da sua utilização (DEITEL, 2011).

Exemplo:

```
Algoritmo Compara_Numero

#include <stdio.h>

int main()

{

int n, soma = 0, c, valor;

printf("Digite o numero a ser comparado?\n");
```

```
scanf("%d", &n);

printf("Digite %d que serão comparados\n", n);

for (c = 1; c <= n; c++){scanf("%d", &valor);

if(valor > n){

soma = soma + 1;

}

}

printf("Quantidade de números maiores que  = %d\n", soma);

return 0;

}
```

Operadores de incremento

Durante o decorrer da unidade talvez você tenha reparado que, foi utilizado em algumas estruturas de repetição variáveis que controlam o número de repetições, no código que utilizamos para realizar a de impressão de números de 1 a 10, no final de cada iteração tínhamos: teste = teste + 1, essa sentença tem o significado que a variável teste vai receber um novo valor por conta do operador que realiza a atribuição, sendo assim novo valor é calculado do seguinte moda, ele realiza a soma de 1 ao valor da variável teste no lado direito da atribuição.

reflita

Reflita

Existe a possibilidade de se trabalhar com o pré-incremento e o pós-incremento, e também com o pré-decremento e o pós-decremento. Sendo Assim, é possível se utilizar os operadores a frente da variável.

Vamos a um exemplo prático da aplicação, utilizando para imprimir os números de 1 até 50 com while:

```
int teste= 1;

while (teste <= 50)

{ printf("%d\n" , teste);

++teste;

}
```

É normal que alguns desenvolvedores com mais experiência utilizem alguns operadores dentro de expressões mais complicadas, ou dentro das próprias condições que realizam o controle da execução de um comando while, nesta unidade vamos utilizar estes operadores apenas em

expressões simples, pois é necessário desenvolver um código mais fácil para que facilite o entendimento das expressões, sendo assim o teste ++ recebe um novo valor, e é atribuído à variável, cujo valor será acrescido com mais uma unidade, logo em seguida o novo valor, já com o crescimento da unidade, é retornado como o valor do cálculo da expressão dentro do comando. Uma forma de manipulação dos incrementos pode ser assim

```
#include <stdio.h>

int main()

{

int x;

while (x<2) {

printf("%d\n" , ++x );

}

}
```

No caso a saída obtida sera 1 e 2, Se o incremento for manipulado de outra forma

```
#include <stdio.h>

int main()

{

int x;

while (x<2) {

printf("%d\n" , x++ );

}

}
```

No caso a saída obtida sera 0 e 1, Ou seja, o primeiro modo incrementado a variável após utilizar o valor atual, e a segunda incrementa antes de utilizar o valor.

reflita

Reflita

Caso uma variável acumuladora ou contadora não for inicializada, os resultados do algoritmo podem estar incorretos. Por isso, é muito importante inicializar todas as variáveis acumuladores e contadores.

Fonte: DEITEL (2011, p. 53).

FOR ou PARA

Podemos encontrar a instrução for em todas linguagens procedurais de programação, normalmente a sua forma mais fácil, versa na inicialização é uma instrução de atribuição que o compilador utiliza para formar a variável que vai realizar o controle do laço, a condição é uma expressão de relação que vai realizar o teste da variável de controle do laço contra um certo valor para definir quando o laço vai terminar, a parte do incremento pode ser tanto positivo quanto negativo e vai determinar a forma como a variável de controle do laço vai ser modificada cada vez que o computador realizar a repetição do loop. Vamos entender um exemplo básico.

Estrutura	Condição	Quantidade de execução	condição da existência
Enquanto	Inicio	0 ou n	condição verdadeira
Repita	Final	1 ou n	condição falsa
Para	Inicio	0 ou n	condição verdadeira

Quadro 3.1 - Tipos de Estruturas
Fonte: Autor.

```
int main()

{

int x;

for (x = 1; x > 1; x--)

{

printf("%d ", x);

}

return;

}
```

Atividade

Um aluno foi solicitado a empregar o conceito de estrutura de repetição no desenvolvimento de um código para observar a saída da frase “Unidade 3”. Sabendo disso é necessário compreender o código e analisar como deve ser realizado a apresentação da frase por 3 vezes.

```
#include <stdio.h>

int main() {

int x;

_____

printf("Unidade 3");

return 0;

}
```

- ○ **a)** for(x = 1; x< 3; x++)
- ○ **b)** for(x = 1; x<= 3; x++)
- ○ **c)** for(y = 1; y<= 3; y++)
- ○ **d)** if(X <=1 ++3)
- ○ **e)** if(X =1 ++3)

Estrutura de Repetição

É dentre as 3 formas de repetição a mais fácil, a do-while realiza a repetição de um bloco de código enquanto uma condição continuar verdadeira, caso a condição seja falsa, as instruções dentro do-while não vai ser realizadas e a execução vai continuar com as instruções após o while. Já a repetição do while é controlada por uma condição que analisa uma determinada variavel.,

Entretanto para que o while trabalhe de forma correta é importante que essa variável suporte adulteração dentro do while, um exemplo um contador. Após estar dentro da repetição, o bloco de instrução sempre será realizado a execução, mesmo que dentro do bloco de instrução a variável que está realizando o controle da execução seja modificada

Do While

Muito semelhante com o while, entretanto tem uma diferença decisiva, a condição é determinada após a execução o bloco de comandos. Existe uma bloco de instrução e logo depois é realizado a verificação.

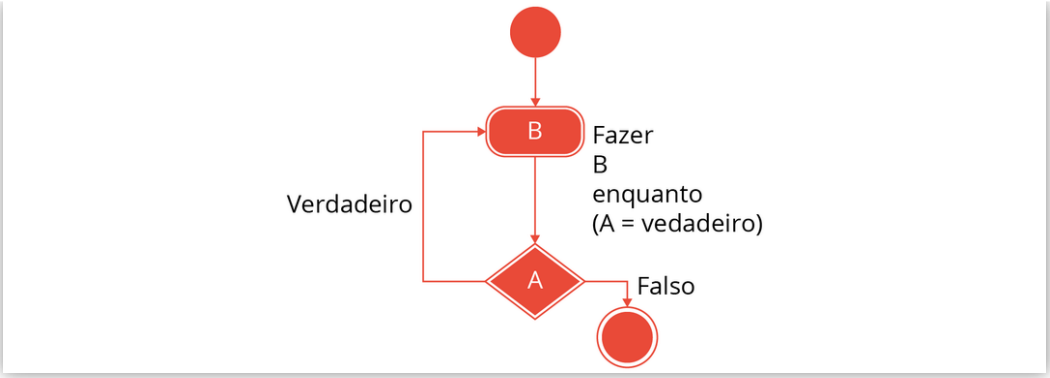


Figura: Do While
Fonte: Autor

De tal modo caso a variável de condição for demudada dentro do bloco de instrução, isso vai afetar a validação da espécie. A alternativa entre while e do while é menor, então vai depender do bom discernimento do programador, que vai optar pela estrutura que vai deixar o algoritmo mais fácil e legível.E uma observação a ser ressaltada que o do-while executa as instruções ao menos uma vez diferente do while.

Atividade

Durante uma aula o professor escreveu o seguinte pseudo código no quadro :

```
#include <stdio.h>

int main()

{

int y,z;

y = 1;

z = 2;

do{

z = z + 2;

//printf("%d \n",y);

y++;

} while (z < 10);

printf("z = %d \n",z);

printf("y = %d",y);

return 0;

}
```

O professor determinou que os alunos avaliassem e escrevessem o resultado que seria imprimido para os Valores Y e Z. Qual seria?

- ○ **a)** 9 e 11
- ○ **b)** 10 e 8
- ○ **c)** 10 e 5C.
- ○ **d)** 9 e 4
- ○ **e)** 11 e 11

Implementação de Algoritmos com Estrutura de Repetição

Laços ou loops na estruturas de repetição, são instruções existentes nas linguagens de programação que são determinadas a realizar execuções de uma ou mais comandos quantas vezes forem programadas para fazer, cada ciclo de um loop ou laço é chamado de iteração, podemos ter também um laço dentro de outro laço. E nesse conteúdo vamos realizar a implementação das principais estruturas, While, For e do-While.

FOR(PARA)

A instrução for é uma estrutura de repetição bem usada pelos desenvolvedores, é bem útil quando se sabe de antemão o valor de quantas vezes será executado a repetição, normalmente essa instrução usa uma variável para realizar o controle e a contagem do laço, pode se dizer que é um comando bem enxuto, já que ele próprio consegue realizar a inicialização, o incremento e também o encerramento do loop.

. Acompanhe a sintaxe da estrutura do for

```
for (inicialização; condição; incremento) { instrução_1;

instrução_2;

...

instrução_n;

}
```

A inicialização é a instrução que realiza a atribuição que começa em variável que realiza a forma de controle do laço, já a de condição é a expressão utilizada que vai ser determinada no final do laço, e o incremento: realiza a definição da variável que vai controlar e fazer a mudança a cada passada no laço.Vamos apresentar uma implementação

```
#include <stdio.h>

#include <conio.h>

int main()

{

int i; //variável de realiza o controle do laço

for(i = 1; i <= 15; i++)

{

printf("%d ", i);

}

return;

}
```

Resultado de saída : 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

A instrução For é utilizado para realizar a execução de um conjunto de comandos ou instruções por um número X de vezes. É incidida uma situação no começo, no caso uma condição e uma atuação a ser realizado a execução a cada vez que se realiza a repetição. Uma variável é normalmente tem um valor no início, que vai ser usada para realizar o controle e a quantidade de vezes em que o conjunto de instruções vai ser executado. E no fim do conjunto de instruções a variável vai sofrer uma modificação, realizando o aumento ou a diminuição de acordo com a lógica usada.

While(ENQUANTO)

A instrução while, no caso "enquanto", é um laço que realiza uma execução para alguns comandos enquanto uma condição for verdadeira, ou seja isso faz com que a instrução seja realizada a execução de uma vez a cada averiguação da condição, de forma geral as instruções sempre necessitam ser desenvolvida de um modo que se leve a condição de execução a ser falsa em um determinado evento, de modo a interromper o loop para que o resto do programa entre e realize a execução. Entenda a sintaxe do while.

```
int x=5;

while (condição)

{

// Dentro das chaves é inserido os comandos

printf("%d\n" , x);  // aqui apresenta o valor da variavel

}
```

Podemos observar no algoritmo acima que onde está escrito condição é o local deve ser inserido a regra para realizar a execução do comando while, e onde está escrito comando é o que vai ser realizado. Vamos observar um código que vai realizar a impressão de números de 1 até 10.

```
#include <stdio.h>

#include <conio.h>

int main()

{

int teste= 1;

while (teste <= 10)

{

printf("%d\n" , teste);

teste= teste+ 1;

if(teste == 5){

teste= teste+ 1;

}

}

}
```

Vamos entender o código, a variável definida como teste que realizará o controle do comando while. Ela vai guardar o valor que será impresso, o comando while vai realizar a verificação se o número está dentro do limite da condição inserida no caso se é menor ou igual a 10, no começo, a variável teste contém o valor 1 e, assim sendo, o comando while é atendida, o bloco de expressões será executado, imprimindo o valor da variável teste e aumentando seu valor em + 1, atente-se que isto vai afetar a condição que realiza o controle do bloco, pois nas próximas verificações a expressão também será verdadeira mas o valor verificado será 2, e assim por diante, entretanto no momento que o valor chegar em 5, existe uma condição que fará com que ele entre no bloco e realize a soma de mais 1, ou seja

apresentando o valor 6, depois dessa operação ele continua a execução até o valor chegar no valor 11, que a expressão que realiza o controle vai compararse 11 é menor ou igual a 10, sendo assim a condição vai se tornar falsa, sendo assim o bloco de repetição finaliza.Por fim o resultado apresentado será 1-2-3-4-6-7-8-9-10, pulando o 5.

Do-While

O comando do-while podemos dizer que nada mais é do que um *while* invertido, onde é necessário inserir o comando que será realizado as repetições antes da verificação da condição de vai ser executado ou seja os comandos do laço vão realizar a execução ao menos uma vez. Entenda a sintaxe do comando do-while.

```
do {

comando;

} while (condição);
```

Onde está definido como condição é a regra que será seguida no momento da execução do comando *do-while* , as instruções que pertencem ao laço que vai parar de realizar a repetição quando a condição for falsa, o algoritmo desenvolvido abaixo apresenta como seria o código do exemplo utilizado no tópico anterior do comando while, vamos converter para a utilização do comando *do-while* . Vamos entender o algoritmo que vai imprimir o número de 1 até 10:

```
#include <stdio.h>

#include <conio.h>

int main()

{

int teste= 1;
```

```
do{

printf("%d\n" , teste);

teste= teste+ 1;

if(teste == 3){

teste= teste+ 1;

}

} while (teste <= 10) ;

}
```

Como no tópico anterior realizamos a declaração de uma variável teste que vai realizar o controle do comando do-while, ela vai salvar o próximo valor a ser impresso, sendo assim a expressão que realiza a verificação se o valor está no limite desejado no caso se é menor ou igual a 10, sendo assim depois da verificação o bloco de código é executado, realizando a impressão do valor de teste e aumentando seu valor em uma unidade, logo depois de realizar a execução do bloco, a expressão vai analisar se a variável teste vai permanecer dentro do valor permitido no caso 10, se ele ainda se encontrar o bloco vai ser executado de novo. Sendo assim o bloco é executado quando o valor de teste guardar de 1 até 10, ao final da execução, o valor da variável teste será o valor 11, que foi será o valor que vai tornar a expressão do comando como falsa , ou seja a execução vai ser interrompida.Entretanto é possível observar que no meio da script existe uma condição que quando ele for 3 irá somar mais 1, ou seja dando 4, sendo assim o resultado de saída seria 1-2-4-5-6-7-8-9-10, ou seja pularia a apresentação do 3

Atividade

O professor durante o desenvolvimento de uma aula de lógica de programação realizou a apresentação do algoritmo aos alunos, para testar a capacidade de analisar uma algoritmo desenvolvido em uma determinada linguagem, você analisando o algoritmo abaixo, responda qual seria a saída de resultado das instruções.

```
int main()

{

int x,y;

for(x=0, y=0; x+y < 100; x=x+1, y=y+1)

printf("%d ",x+y);

system("PAUSE");

return 0;

}
```

- ○ **a)** imprime números de 0 a 98, indo de 1 em 1
- ○ **b)** imprime números de 0 a 98, indo de 2 em 2
- ○ **c)** imprime números de 1 a 100, indo de 1 em 1
- ○ **d)** imprime números de 1 a 100, indo de 2 em 2
- ○ **e)** imprime números de 0 a 100, indo de 2 em 2

Análise Entre For, While e Do-While

Vamos analisar as estruturas entender suas diferenças, a estrutura de controle de fluxo no caso o enquanto realiza a repetição do bloco de instruções até que a condição necessária acabar resultando em uma condição falsa.

Pode ocorrer do código nunca ser executado já que a verificação da condição acontece antes, nesse momento específico é como um estrutura de condição if, onde só vai ser executado se no código a condição for dada como verdadeira, a grande diferença é que ao ser executado o código ele vai até ao final do bloco e depois volta para o começo para realizar a verificação novamente, e conseguimos entender que o do-while é parecido com o while entretendo só muda sua forma de realizar a verificação, que está empregada no final ou seja, ele executa o comando e depois realiza a verificação.

compreendemos que o comando for e parecido com o while, mas é um outro modo mais estruturada de um modo geral é utilizado para ir de um ponto a para outro ponto onde se sabe qual o valor do início e o valor do fim, além de ter um comando de avanço em cada passo realizado.

Atividade

Durante uma apresentação de cargos em uma empresa o diretor apresentava como iria funcionar as metas de cargos e salários e as bonificações, sendo assim o diretor solicitou a um programador realizar uma implementação simples utilizando o do-while para realizar um aumento e apresentar na tela para seus funcionários, analise o código e responda, quantas vezes o código irá rodar ou seja ter iteração obedecendo às condições impostas no algoritmo e qual será o último número apresentado.

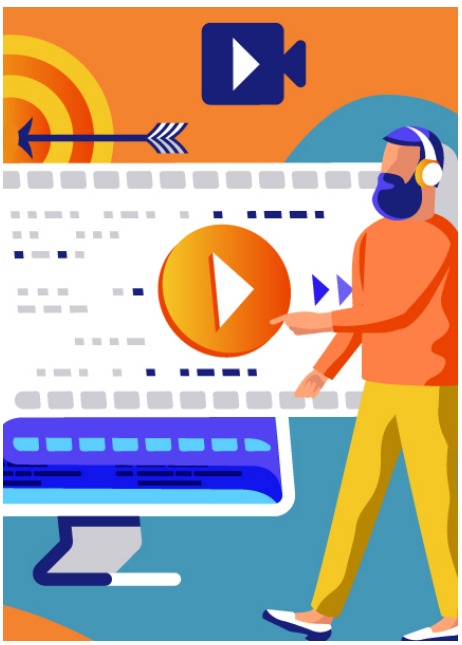
```
#include <stdio.h>

int main()

{

int i = 200;

do{

printf("O valor atual do salario é de " "%d ", i  );

i += 50;

} while (i < 300);

return 0;

}
```

- ○ **a)** 1 iterações o último valor impresso será 200
- ○ **b)** 3 iterações o último valor impresso será 300
- ○ **c)** 2 iterações o último valor impresso será 300
- ○ **d)** 3 iterações o último valor impresso será 250

○ **e)** 2 iterações o último valor impresso será 250

indicações

Material Complementar



FILME

Estruturas de Repetição 1 - Curso de Algoritmos #09 - Gustavo Guanabara

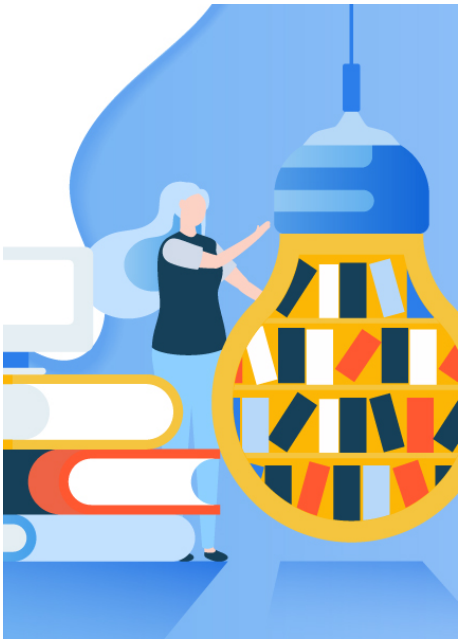
Ano: 2014

Comentário: A estrutura de repetição ENQUANTO vai permitir a execução de blocos de instruções diversas vezes simplificando o modo de realizar a representações lógicas que serão construídas para desenvolver aplicações.

Veja de uma maneira simples e objetiva como criar códigos que realizam repetições

Para conhecer mais sobre o filme, acesse o trailer disponível.

TRAILER



LIVRO

Introdução à Estruturas de Dados

Waldemar Celes, Renato Cerqueira e José Lucas Rangel

Editora: Elsevier Editora LTDA

ISBN: 978-85-352-8346-4(versão digital)

Comentário: Este livro foi utilizado por mim como base para estudos durante minha graduação ele tem o foco no o ensino prático de programação com embasamento na utilização certa e de forma eficiente o uso de diferentes estruturas de dados, ele foi de grande auxilio para me guiar a um conhecimento sólido de programação e de estruturação de dados, esta segunda edição traz um conteúdo que auxilia na introdução de estruturas de dados mais novas, com uma boa forma de didática, este livro consegue apresentar aspectos bem práticos de programação e estruturação de dados usando como linguagem a programação em C.

conclusão

Conclusão

A estrutura de repetição é uma excelente ferramenta que pode ser utilizado durante a programação, temos que ter cuidado ao utilizá-la, pois podemos cair em alguns problemas como um loop infinito, e saber qual e quando utilizá-las dependendo da forma como será inserida a informação que será lida pelo código, normalmente quando se sabe o valor de entrada pode-se utilizar o for e quando não se sabe por exemplo em uma estrutura igual o Enem, onde um sistema precisa realizar a leitura de finitas provas mas é impossível no momento saber quantas são, utiliza-se o while. Por fim é necessário entender que a estrutura ajuda na manutenção dos códigos e no amparo de um projeto ou sistema mais limpo onde não vai necessitar da utilização de muitos blocos de código.

referências

Referências Bibliográficas

Arnaldo V. Moura, Daniel F. Ferber. Título: **Estruturas de Repetição** , Unicamp, Campinas-SP, 2008

DEITEL, Paul; DEITEL, Harvey. **C Como Programar** . 6. ed. São Paulo: Pearson, 2011.

GUEDES, S. **Lógica de programação algorítmica** . São Paulo: Pearson Education do Brasil, 2014.

IMPRIMIR