CURSO CIÊNCIA DA COMPUTAÇÃO
DISCIPLINA: INTRODUÇÃO À PROGRAMAÇÃO
TEMA: ESTRUTURA DE REPETIÇÃO DO..WHILE

TEXTO PARA APOIO AO ESTUDO

INTRODUÇÃO

Estruturas de controle são essenciais no desenvolvimento de programas, possibilitando analisar e tratar situações condicionais em que pode ocorrer algum desvio no curso de ação do programa, assim como utilizar recursos para repetir determinados conjuntos de comandos e operações.
Há casos em que precisamos que um bloco de comandos seja executado repetidas vezes. Para isso, utilizamos **estruturas de repetição**. Isso nos permite repetir um bloco de comandos quantas vezes forem necessárias. A linguagem C utiliza três estruturas de repetição: FOR, WHILE e DO..WHILE. Esse material abrange a parte de estrutura de repetição, utilizando o comando DO..WHILE.

ESTRUTURA DE REPETIÇÃO DO..WHILE

É uma estrutura de repetição que, diferente das outras estruturas de repetição, verifica o teste da condição ao final do corpo da estrutura. Desse modo, os comandos do laço *do..while* são executados ao menos uma vez. Essa estrutura costuma ser muito utilizada com o processamento de menus de opções.

Sintaxe:

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

Onde:

1) A instrução (bloco de execução) é executada;
2) A condição é avaliada;
3) Se o resultado da condição retornar o valor verdadeiro, então volta-se ao ponto 1;
4) Se o resultado da condição avaliada em 2) for falso, então o laço DO..WHILE finaliza e o programa continua na instrução imediatamente posterior ao laço.

Exemplo 1:

```
#include <stdio.h>
int main() {
        int i =1;
        do {
                printf ("%d\n", i);
                i++;
        } while(i <=10);
}
```

Ao executar esse exemplo, o programa retornará como resultado a impressão dos números de 1 a 10 na tela.

Exemplo2: Uso da repetição para informar um número positivo.

```
#include <stdio.h>
int main() {
	int num;
	do {
		printf ("\nInforme um número positivo:");
		scanf(“%d”, &num);
	} while(num <0);
	printf(“Número positivo: %d”,num);
}
```

No exemplo acima, o programa repetirá a leitura de um número enquanto o usuário não digitar um número positivo. Ao digitar o valor positivo, o programa encerrará a repetição e exibirá esse valor.

A estrutura *do..while* é comumente utilizada quando deseja-se apresentar menus com opções. E exemplo a seguir ilustra essa situação.

Exemplo3: Fazer um programa que apresente ao usuário o seguinte menu de opções:

```
**** Cálculo de áreas ****
1- Quadrado
2- Triângulo
3- Retângulo
4- Sair
```

O programa deve permitir que o usuário faça qualquer uma das operações quantas vezes ele quiser.

```
#include <stdio.h>
int main(){
    int op, l1, l2;
    float area;
    do{
        printf("\n**** Cálculo de áreas ****\n");
        printf("1- Quadrado\n");
        printf("2- Triângulo\n");
        printf("3- Retângulo\n");
        printf("4- Sair\n");
        printf("Escolha sua opção: ");
        scanf("%d",&op);
        switch(op){
            case 1:{
                printf("Informe o lado do quadrado: ");
                scanf("%d",&l1);
                area=l1*l1;
                printf("Área: %.1f\n",area);
                break;
            }
            case 2:{
                printf("Informe a base do triângulo: ");
                scanf("%d",&l1);
                printf("Informe a altura do triângulo: ");
                scanf("%d",&l2);
                area=(l1*l2)/2;
```

```
                printf("Área: %.1f\n",area);
                break;
            }
            case 3:{
                printf("Informe a base do retângulo: ");
                scanf("%d",&l1);
                printf("Informe a altura do retângulo: ");
                scanf("%d",&l2);
                area=l1*l2;
                printf("Área: %.1f\n",area);
                break;
            }
            case 4:{
                printf("\nSaindo...\n");
                break;
            }
            default:{
                printf("\nOpção inválida!\n");
                break;
            }
        }
    }while(op!=4);
}
```

No exemplo 4, o menu de operações será exibido repetidas vezes enquanto o usuário não escolher a opção de 4 para encerrar o programa. Como o menu deve ser apresentado ao menos uma vez, essa estrutura se adequa perfeitamente a esse tipo de situação.

COMANDOS DE DESVIO

Alguns comandos de desvio são utilizados com o objetivo de interromper de algum modo a execução do programa ou de alguma parte do código.

Comando break

A instrução *break* já é conhecida e foi utilizada para finalizar o conjunto das instruções executadas dentro de um comando *switch*. A instrução *break*, quando utilizada dentro de uma repetição, finaliza o laço correspondente dando sequência ao programa na instrução posterior a esse laço.

Exemplo 5: Uso do comando break.

```
for (i = 1; i <= 10; i++){
        scanf("%d", &num );
        if (num < 0)
            break;
        printf ("%d\n", num );
}
```

No exemplo 5, o programa possibilita que o usuário informe até 10 números, no entanto, ao digitar um valor negativo, o programa encerra a repetição utilizando o comando *break*.

Comando continue

A instrução *continue*, quando utilizada dentro de uma repetição, finaliza a execução corrente do laço, passando para a próxima iteração do laço.

Exemplo 6: Uso do comando *continue*.

```
for (i = 1; i <= 10; i++){
        scanf("%d", &num );
        if (num < 0)
                continue;
        printf ("%d\n", num );
}
```

No exemplo 6, o programa permite a leitura de 10 números, no entanto, ao digitar um valor negativo, o programa encerra a iteração corrente e não exibe o número negativo, mas retorna à repetição na próxima iteração.

Comando return

É usado para interromper a execução de uma função e retornar um valor ao programa que chamou esta função. Caso haja algum valor associado ao comando *return*, este é devolvido para a função, caso contrário um valor qualquer é retornado. Este comando será visto mais adiante, no estudo de funções.

Comando exit()

Provoca a terminação de um programa, retornando o controle ao sistema operacional. O protótipo da função é void exit (int codigo). O código é usado para indicar qual condição causou a interrupção do programa. Usualmente utiliza-se o valor 0.

Estruturas de repetição são estruturas fundamentais para possibilitar a repetição de instruções sem que elas tenham que ser escritas diversas vezes. A estrutura DO..WHILE possibilita configurar repetições a partir de condições predefinidas e também em situações em que o número de repetição não é conhecido previamente. No entanto, diferente das outras estruturas, os comandos associados à repetição são executados ao menos uma vez, já que o teste da condição é realizado apenas no final da estrutura.

Para praticar os conceitos envolvidos nesse material, segue uma sugestão de atividades.

ATIVIDADE

1. Faça um programa que calcule o fatorial de um número positivo informado pelo usuário. Exemplo:
   5! = 5 * 4 * 3 * 2 * 1
   4! = 4 * 3 * 2 * 1

2. Faça um programa que leia vários números e encontra e imprima o maior número. O algoritmo encerra a leitura quando o usuário digitar -9999.

AS INFORMAÇÕES CONTIDAS NESTE MATERIAL DE APOIO AO ESTUDO FORAM BASEADAS NAS SEGUINTES PUBLICAÇÕES:

DAMAS, L. LINGUAGEM C. Rio de Janeiro: LTC, 2007.

ALBANO, R. S.; ALBANO, S. G. PROGRAMAÇÃO EM LINGUAGEM C. Rio de Janeiro: Editora Ciência Moderna, 2010.

ASCENCIO, A. F. G.; Edilene, A. V. FUNDAMENTOS DA PROGRAMAÇÃO DE COMPUTADORES - ALGORITMOS, PASCAL, C/C++ E JAVA. 2. ed. São Paulo: Pearson, 2007.