Estruturas de Repetição

Professores(as):
Virgínia Fernandes Mota
João Eduardo Montandon de Araujo Filho
Leandro Maia Silva

INTRODUÇÃO A PROGRAMAÇÃO - SETOR DE INFORMÁTICA

Estruturas de Repetição

- Uma estrutura de repetição permite que uma sequência de comandos seja executada repetidamente, enquanto determinadas condições são satisfeitas;
- Essas condições são representadas por expressões lógicas (como por exemplo, $A > B$, $C == 3$, Letra==’a’).
- As estruturas de repetição podem ser dos seguintes tipos:
  - Repetição com teste no início (pré-testadas);
  - Repetição com teste no final (pós-testadas);
  - Repetição contada (controle).

Estruturas de Repetição

- O real poder dos computadores está na sua habilidade para repetir uma operação ou uma serie de operações muitas vezes;
- Esta repetição chamada laços(loop) é um dos conceitos básicos da programação estruturada.

Repetição com teste no início

```
1 ...
2 while (condição){
3     bloco de operações
4 }
5 ...
```

Repete o bloco de operações enquanto a condição for verdadeira

Comando while

Passos:

1. avalia a condição:
   - se verdadeiro, executa o bloco de operações;
   - caso contrário, termina o laço.

Comando while - Exemplo

Faça um programa que mostre na tela os números de 1 a 100:

```
1 #define SUCESSO 0
2 int main(int argc, char ** argv){
3    printf("1 2 3 4 ... ");
4    return SUCESSO;
5 }
```

A solução acima é inviável para valores grandes. Precisamos de algo mais eficiente e inteligente, não é mesmo?

Comando while - Exemplo

Faça um programa que mostre na tela os números de 1 a 100:

```c
#define SUCESSO 0
int main(int argc, char ** argv){

   int numero = 1;

   while(numero <= 100){
      printf("%d ", numero);
      numero++;
   }

   return SUCESSO;
}
```

Observe que a variável numero é usada como um contador, ou seja, vai contar quantas vezes o loop será executado

Repetição com teste no final

```
1 ...
2 do{
3       bloco de operações
4 }while(condição)
5 ...
```

O comando do-while: é utilizado sempre que o bloco de comandos deve ser executado ao menos uma vez, mesmo que a condição seja falsa.

Comando do-while

Passos:

1. executa o bloco de operações;
2. avalia a condição:
   - se verdadeiro, re-executa o bloco de operações;
   - caso contrário, termina o laço.

Comando do-while - Exemplo

```c
#include <stdio.h>
#include <stdlib.h>

#define SUCESSO 0

int main(int argc, char ** argv){
     int i;

     do{
          printf("Escolha uma opacao: \n");
          printf("(1) opcao 1\n");
          printf("(2) opcao 2\n");
          printf("(3) opcao 3\n");
          scanf("%d",&i);
     }while((i < 1) || (i > 3));

     return SUCESSO;
}
```

Comando do-while - Exemplo

Qual a diferença entre os dois trechos de código?

```
1 ...
2       int A = 0;
3
4       while(A < 10){
5             printf("%d", A);
6             A++;
7       }
8
9 ...
```

```
1 ...
2       int A = 0;
3
4       do{
5             printf("%d", A);
6             A++;
7       } while(A < 10);
8
9 ...
```

Repetição contada

```
1 ...
2     for (inicialização; condição; incremento){
3         bloco de operações
4     }
5 ...
```

1. Inicialização: inicia uma variável;
2. avalia a condição. Se for verdadeira, executa o bloco de operações, senão encerra o laço;
3. Incremento: ao término do bloco de operações, incrementa o valor da variável;
4. repete o processo até que a condição seja falsa.

Comando for

Passos:

1. Executa a inicialização;
2. avalia a condição:
   - se verdadeiro, executa o bloco de operações;
   - caso contrário, termina o laço.
3. Executa o incremento e volta para (2).

Comando for- Exemplo

- Em geral, utilizamos o comando for quando precisamos ir de um valor incial até um valor final;
- Para tanto, utilizamos uma variável para realizar a contagem. Exemplo:

```c
#include <stdio.h>
#include <stdlib.h>

#define SUCESSO 0

int main(int argc, char ** argv){
   int i;

   for(i = 1; i <= 10; i++){
      printf("%d \n", i);
   }

   return SUCESSO;
}
```

Comando for- Exemplo

Faça um programa em C que leia um valor inicial e um valor final, e depois imprima todos os números inteiros entre o valor inicial e final

```c
#include <stdio.h>
#include <stdlib.h>

#define SUCESSO 0

int main(int argc, char ** argv){
   int vInicial, vFinal;

   printf("Digite o valor inicial: \n");
   scanf("%d",&vInicial);
   printf("Digite o valor final: \n");
   scanf("%d",&vFinal);

   for(i = vInicial; i <= vFinal; i++){
      printf("%d \n", i);
   }

   return SUCESSO;
}
```

Como sair prematuramente de um loop?

1. break: Força o loop terminar;

   Vai para a linha imediatament depois do fim do loop.

2. continue: Pula para a próxima iteração;

   Volta para o início do loop sem executa as instruções que estão abaixo do *continue*. O incremento é executado normalmente.

3. exit: Finaliza o programa.

   Não deve ser utilizado a menos que você tenha certeza que a única solução é encerrar seu programa neste ponto. Extremamente deselegante e vai deixar o professor Leandro bem triste se usar.

Exercícios

1. Elabore um programa que calcule N! (fatorial de N), sendo que o valor inteiro de N é fornecido pelo usuário.
2. Escrever um algoritmo que lê um valor N inteiro e positivo e que calcula e escreve o valor de E: E = 1 + 1/2! + 1/3! + ... + 1/N!
3. Faça um programa que, dado um conjunto de valores inteiros e positivos (fornecidos um a um pelo usuário), determine qual o menor e o maior valor do conjunto. O final do conjunto de valores é conhecido através do valor zero, que não deve ser considerado.
4. Fazer um programa para calcular e mostrar os N primeiros termos da série de Fibonacci. O número N é fornecido pelo usuário. A série de Fibonacci é : 1 1 2 3 5 8 13 ... Isto é f1 = f2 = 1, f3 = f1 + f2, f4 = f2 + f3...
5. A conversão de graus Fahrenheit para Centígrados é obtida pela fórmula C = 9*(F-32)/5. Escreva um programa que calcule e escreva uma tabela de graus centígrados em função de graus Fahrenheit que variem de 50 a 150 de 1 em 1.