Praticando

Estruturas Simples e Condicionais

No material anterior abordamos conceitos básicos e iniciais da nossa disciplina, **foi interessante?** Esperamos que sim. Agora este novo material é um convite para praticar conosco o uso desses comandos simples e avançar na prática do uso dos comandos condicionais. **Que tal, vamos lá?**

Vimos que, por enquanto, na estrutura de um programa em C possuíamos diretivas de pré-processamento, cabeçalhos da função principal e o corpo da função principal (dentro de um bloco de comandos **{ }**). Então vamos pensar em um programa apenas para escrita de uma frase na tela do usuário, teríamos:

```
#include <stdio.h>                    } Diretiva do pré-processador

main( ) {                             } Cabeçalho da função principal
     printf("Primeiro Programa ");
}
```

Observe que a diretiva de pré-processamento utilizada foi para uso da biblioteca de entrada e saída de informações, uma vez que precisamos fazer uso da função printf.

Caso fôssemos declarar uma váriável constante, onde iríamos inseri-la? Vimos no material anterior que toda variável constante não muda seu valor e para diferenciá-la das demais também fazíamos uso de um pré-processador, logo, ficaria assim:

```
#include <stdio.h>                    }
#define pi 3.14                       } Diretivas do pré-processador

main ( ) {                            } Cabeçalho da função principal
   printf ("Segundo Programa \n");    }
   printf ("valor de pi = %f", pi);   } Programa principal com 2 printf's diferentes
}
```

Observe que a variável constante é declarada junto às bibliotecas, onde as diretivas de pré-processamento são incluídas. E neste programa incluímos o operador de "pula linha" \n e também fizemos uso de outra forma do printf parametrizando a frase a ser impressa com uso do fomatador para o tipo de variável de "ponto flutuante" (float), representado por %f.

Seguimos...

Vamos pra outra questão: **um programa para solicitar os lados de um retângulo, determinar e mostrar sua área e seu perímetro.**

Neste momento devemos **observar algumas características** do nosso programa:

- Quantas variáveis iremos utilizar e quais são seus tipos?
- Vamos precisar solicitar alguma informação ao usuário do programa?
- Vamos apresentar alguma coisa para o usuário? Se sim, o que, e qual tipo?

Vamos responder pensando sobre o enunciado da questão:

- Quantas variáveis iremos utilizar e quais tipos? Duas variáveis do tipo float, uma para cada lado do retângulo, será float pois os lados de um objeto geométrico possui tamanhos reais (não inteiros). Poderíamos chamar essas variáveis de ladoA e ladoB, que tal?
    - Seriam declaradas com o comando: **float ladoA, lado B;**
- Vamos precisar solicitar alguma informação ao usuário do programa? Sim, não sabemos os valores dos lados para calcular o perímetro. Logo, ladoA e ladoB devem ser informados pelo usuário .
- Vamos apresentar alguma coisa para o usuário? Se sim, o que, e qual tipo? Sim, no final vamos calcular o perímetro do retângulo. Eita, mas o que é o perímetro? Vamos dar um "google" pra lembrar?

  > O perímetro é a medida do contorno de um objeto bidimensional, ou seja, a soma de todos os lados de uma figura geométrica

  Fácil né? Então perímetro é a soma de todos os lados, então seria 2*(ladoA + ladoB) nessa questão. Resolvido! Precisamos de uma variável a mais, do mesmo tipo de laboA e ladoB (float), para receber esse cálculo. Logo:

    - **Perimetro = 2 * (ladoA + lado B);**

  Com isso fica mais fácil implementar, né? Olha só:

```
#include <stdio.h>
main( ) {
    printf (“Terceiro Programa \n");
    float ladoA, ladoB;                  //declaração de variáveis
    float perimetro;

    //solicitando as informações do usuário:
    printf (" Informe os tamanhos dos lados do retangulo: ");
    scanf ( "%f %f", &ladoA, &ladoB);
    perimetro = 2 * (ladoA + ladoB);

    printf ("\n\tSeu perímetro vale. %.4f\n\n", perimetro);
}
```

Tudo bem até aqui? Opa, alguém pode dizer tudo mais ou menos, o que é %.4f ? Não seria %f ? Pois é, essa é uma forma ainda melhor de formatar um número real (de “ponto flutuante”), com ela informamos quantos números queremos após o ponto. Neste caso está configurado para 4, pois colocamos %.4f, se quisessemos 2 números após o ponto, colocaríamos %.2f. Legal, né? **Olha aí, programar é bom demais!**

Vamos pra uma próxima? **Que tal uma de condicional?**

Faça um programa para ler um inteiro e determinar se é par ou ímpar.

Vamos responder aquelas perguntinhas, igual a última questão?

- Quantas variáveis iremos utilizar e quais tipos?
- Vamos precisar solicitar alguma informação ao usuário do programa?
- Vamos apresentar alguma coisa para o usuário? Se sim, o que, e qual tipo?

Vamos responder pensando sobre o enunciado da questão:

- Quantas variáveis iremos utilizar e quais tipos? Bom, será apenas um inteiro, podemos chama-lo de x?
- Vamos precisar solicitar alguma informação ao usuário do programa? Observando o enunciado, sim. Pois não temos o valor de x e o usuário vai ter que informar.
- Vamos apresentar alguma coisa para o usuário? Se sim, o que, e qual tipo? Vamos, vamos ter que validar se o número é par ou não e imprimir uma frase na tela.

E aí que mora a “dificuldade” da questão. Como saber se um número é par? Alguns podem ter dúvidas... Mas se observamos, um número é par se ele for divisível por 2. Basta lembra do 6, do 8, do 32... e assim por diante. E se um número é divisível por outro quer dizer que nesse processo de divisão o resto é zero. **OPA! Resolvemos, resolvemos fácil, fácil a questão**. Pois na linguagem C temos um operador que realiza a divisão de dois números e retorna o valor do resto desta operação, e este operador é o **%**. Basta fazer **z % y** e saberemos se a divisão de z por y produziu um valor diferente ou igual a zero. Logo, para essa questão teríamos uma variável **resto = x % 2;**

Show de bola, e **como saber se um valor é igual a outro? Comparando!** E em C aprendemos que a comparação é uma instrução condicional. Se não estamos tratando de menus de opção, então estamos tratando do uso de if e else. Para esta questão teríamos de validar se resto é igual a zero:

if (resto == 0)

E se for diferente de zero:

if (resto != 0)

Ou, poderíamos mudar essa última instrução por: se não for igual, combinando um else com o primeiro if:

if (resto == 0)

else { .... }

Com isso fica mais fácil implementar, né? Olha só:

```
#include <stdio.h>
main( ) {
    printf (“Quarto Programa \n");
    int x;
    int resto;
    printf ("Digite um inteiro: ");
    scanf ("%d", &x);
    resto = x % 2;
    if ( resto == 0)
        printf ("\nO número %d é par.\n\n", x);
    else
        printf ("\nO número %d é ímpar.\n\n", x);
}
```

Vamos pra uma próxima? **Que tal uma de switch-case?**

Vamos juntar essas duas questões em um menu de opções? A ideia é oferecer pro usuário a oportunidade de escolher entre:

1 – Calcular o perímetro de um retângulo, ou

2 – Saber se um número é par ou impar

Nesse caso as opções do usuário são 1 ou 2. Observe que a comparação dos número é inteira. Se tivermos um variável **op** para receber a escolha do usuário, ela será interia. Se ela

é inteira **ou** usamos um **if** para saber se op é igual a 1 ou 2 (if (op==1) e if (op==2). **Ou** temos uma instrução que pode ser utilizada nesta situação: a **Switch - Case!**

Vamos aplicá-la? Ficaria algo assim:

```
#include <stdio.h>
main( ) {
   printf (“Quarto Programa \n");
   int op;
   printf (“Informe a opcao que deseja:\n”);
   printf (“1 – Calcular o perímetro de um retângulo, ou\n”);
   printf (“2 – Saber se um número é par ou impar\n”);
   scanf (“%d”, &op);

   switch (op) {
      case 1: {
                //código do programa 3
      }
      case 2: {
                //código do programa 4
      }

   };
}
```

Legal o uso né? Vamos preencher com o código do programa 3:

```
#include <stdio.h>
main( ) {
   printf (“Quarto Programa \n");
   int op;
   printf (“Informe a opcao que deseja:\n”);
   printf (“1 – Calcular o perímetro de um retângulo, ou\n”);
   printf (“2 – Saber se um número é par ou impar\n”);
   scanf (“%d”, &op);

   switch (op) {
      case 1: {
         float ladoA, ladoB;          //declaração de variáveis
         float perimetro;

        //solicitando as informações do usuário:
        printf (" Informe os tamanhos dos lados do retângulo: ");
        scanf ( "%f %f",&ladoA, &ladoB);
        perimetro = 2 * (ladoA + ladoB);
```

```
            printf("\n\tSeu perímetro vale. %.4f\n\n", perimetro);
        }
        case 2: {
                    //código do programa 4
        }
    };
}
```

Vamos preencher com o quarto programa?

```
#include <stdio.h>
main( ) {
    printf (“Quarto Programa \n");
    int op;
    printf (“Informe a opcao que deseja:\n”);
    printf (“1 – Calcular o perímetro de um retângulo, ou\n”);
    printf (“2 – Saber se um número é par ou impar\n”);
    scanf (“%d”, &op);

    switch (op) {
        case 1: {
            float ladoA, ladoB;             //declaração de variáveis
            float perimetro;

           //solicitando as informações do usuário:
           printf (" Informe os tamanhos dos lados do retângulo: ");
           scanf ( "%f %f",&ladoA, &ladoB);
           perimetro = 2 * (ladoA + ladoB);

            printf("\n\tSeu perímetro vale. %.4f\n\n", perimetro);
        }
        case 2: {
            int x;
            int resto;
            printf ("Digite um inteiro: ");
            scanf ("%d", &x);
            resto = x % 2;
            if ( resto == 0)
                printf("\nO número %d é par.\n\n", x);
            else
                printf("\nO número %d é ímpar.\n\n", x);
        }
    };
}
```

Esse mesmo programa poderia ser escrito com a declaração de todas as variáveis no início do programa, não é verdade? Seria algo assim:

```
#include <stdio.h>
        main( ) {
        printf("Quarto Programa \n");
        float ladoA, ladoB, perimetro;
        int op, x, resto;
        printf ("Informe a opcao que deseja:\n");
        printf ("1 - Calcular o perímetro de um retângulo, ou\n");
        printf ("2 - Saber se um número é par ou impar\n");

        scanf ("%d", &op);

        switch (op) {
        case 1: {

                        printf (" Informe os tamanhos dos lados do retângulo: ");
                        scanf ( "%f %f",&ladoA, &ladoB);
                        perimetro = 2 * (ladoA + ladoB);
                        printf ("\n\tSeu perímetro vale. %.4f\n\n", perimetro);
        }
        case 2: {

                        printf ("Digite um inteiro: ");
                        scanf ("%d", &x);
                        resto = x % 2;
                        if (resto == 0)
                        printf ("\nO número %d é par.\n\n", x);
                        else
                        printf ("\nO número %d é ímpar.\n\n", x);

                }
        };
}
```

Você sabe qual a diferença de usar essa última disposição e a penúltima? Se sim, corre lá no fórum e compartilha com os colegas!