Função

Nesta lista de exercícios, iremos aprender a usar funções. Para isso, será necessário utilizar o CD-MOJ de uma forma diferente do que você está acostumada ou acostumado. Por isso, neste exemplo iremos passar um exercício para multiplicar o valor de Pi e iremos resolver juntos no CD-MOJ.

O problema

1. Neste exercício você deve escrever a função "multiplicaPi"

```
float multiplicaPi(int x)
```

que retorna o valor de pi com precisão simples, duas casas decimais de aproximação e multiplicado pelo inteiro "x"

Entrada

Não há dados de entrada para serem lidos.

Saída

Não há dados de saída para serem impressos.

Exemplo 1

```
multiplicaPi(1)
```

deve retornar

```
3.14
```

Exemplo 2

```
multiplicaPi(2)
```

deve retornar

```
6.28
```

Resolvendo o exercício

Se esse fosse um exercício normal do CD-MOJ, você teria apenas o texto acima para entender e resolver o problema. Mas iremos agora entrar em detalhes como funcionam os exercícios de funções e explicar os problemas mais frequentes durante esses exercícios.

A função "multiplicaPi" recebe como argumento um inteiro e retorna um float. O valor de Pi com duas casas decimais é "3.14". Então o código abaixo resolve o problema na forma que ele foi descrito:

```
float multiplicaPi(int x)
{
    float pi = 3.14;
    return pi * x;
}
```

Mas e o meu scanf? Onde ele está? Você não precisa fazer scanf! Veja que na seção de "Entrada" do problema ele falou "Não há dados de entrada para serem lidos." Por isso, você não precisou colocar scanf no seu código. Além disso, na seção "Saída" ele também disse "Não há dados de saída para serem impressos.". Então, você não colocou nenhum printf no seu código e ele vai ser aceito como resposta correta!

Testando o seu problema

A resposta deste problema não vai ter nenhum scanf e nenhum print. Mas antes de enviar a função para o CD-MOJ, você deve testá-la localmente e verificar a resposta. Existem diversas formas de fazer isso, você pode ler do teclado os valores, etc.

A forma mais rápida é codificar os exemplos diretamente no seu código e imprimir a resposta! Chamamos isso de *hard-code*, codificado rigidamente, seu programa irá imprimir apenas para os exemplos que estão no código-fonte e não para dados dinâmicos, passados pelo usuário.

Então faremos o seguinte, vamos incluir a biblioteca "stdio.h", vamos criar a função main com a variável "resp", que recebe o retorno da função que o exercício pediu e imprime o resultado na tela!

```c
#include <stdio.h>

float multiplicaPi(int x)
{
    float pi = 3.14;
    return pi * x;
}

int main()
{
    float resp;
    resp = multiplicaPi(1);
    printf("%f\n", resp);
    resp = multiplicaPi(2);
    printf("%f\n", resp);
    return 0;
}
```

Compile e execute o programa acima e você verá que ele imprime a seguinte resposta na saída padrão:

```
3.140000
6.280000
```

Exatamente o que queremos!

Enviando para o CD-MOJ

Se você enviar esse último código para o CD-MOJ, ele não será aceito como resposta. Irá causar um erro de compilação. Isso acontece pois o CD-MOJ irá colocar uma função main no seu código para testar a função. Como você tem uma função main no seu código e o CD-MOJ colocou outra função main, o código não irá compilar. Mas você não precisa saber como a função main do CD-MOJ funciona, dificilmente isso estará na descrição do problema. Outro problema que está acontecendo é que seu código está imprimindo dados na saída padrão (print), quando o problema pediu que você não imprimisse!

Acontece que você precisa da função "main" para poder compilar, rodar o código na sua máquina e testar os casos de exemplo. O CD-MOJ não precisa! Então, você pode remover o código da main para ter a resposta correta. Mas e se a resposta não for correta e você precisa testar novamente? Escrever a função main novamente irá dar trabalho. Para não perder a função main, você pode simplesmente COMENTAR a função main. Comentários no código são ignorados pelo compilador (e pelo CD-MOJ) no seu código e não irão mudar o comportamento do seu código:

```c
#include <stdio.h>

float multiplicaPi(int x)
{
    float pi = 3.14;
    return pi * x;
}

/*
int main()
{
    float resp;
    resp = multiplicaPi(1);
    printf("%f\n", resp);
    resp = multiplicaPi(2);
    printf("%f\n", resp);
```

```
    return 0;
}
*/
```

Tudo que estiver entre "/*" e "*/" será ignorado pelo compilador e CD-MOJ para rodar o seu programa, então o código acima irá produzir a resposta certa. Se você precisar testar localmente, basta retirar os comentários! Envie o código acima e obtenha a resposta correta, se quiser testar!

Últimos detalhes

Lembre-se que C é uma linguagem case-sensitive. Neste caso, são diferenciadas letras maíusculas e minúsculas. Então, quando um exercício de função, especifica o nome da função e os parâmetros, você precisa passar de forma IDÊNTICA ao problema. Caso contrário você terá problemas de compilação. Então, o tipo de retorno e a ordem dos parâmetros, diferenciam as funções. Se você criar as funções abaixo o seu código não irá funcionar (erro de compilação).

Exemplo 1

```c
int multiplicaPi(int x)
{
    float pi = 3.14;
    return pi * x;
}
```

Esta função retornar "int", mas "float" é esperado pela especificação do exercício.

Exemplo 2

```c
float multiplicaPi(int x, int y)
{
    float pi = 3.14;
    return pi * x;
}
```

Esta função recebe dois parâmetros "int x, int y", mas pela especificiação do exercício apenas "int x" é esperado.

Exemplo 3

```c
float multiplica_pi(int x)
{
    float pi = 3.14;
    return pi * x;
}
```

Esta função chama-se "multiplica_pi" e o exercício espera uma função chamada "multiplicaPi". Para o computador e para o CD-MOJ, são funções completamente diferentes!!!

Conclusão

Note que NESTE EXEMPLO não foi necessário fazer scanf ou printf dentro da função. Isso nem sempre será verdade e dependerá da descrição do exercício. Você deve ler o texto da questão e entender quando será necessário.

Tenha em mente que para os problemas de funções você não precisa enviar a função main, mas você precisará ser criativo e criar uma função main que teste o seu código localmente. Caso contrário, você nunca irá conseguir encontrar os problemas no seu código.

Além disso, comentar a função main é a melhor prática para resolver esses problemas, pois permite que rapidamente você adapte o seu código e descubra problemas nele.

Bons exercícios!

Author: Daniel Sundfeld <daniel.sundfeld@unb.br>