Funções e Ponteiros

Prof. Msc. Elias Batista Ferreira
Prof. Dr. Gustavo Teodoro Laureano
Profa. Dra. Luciana Berretta
Prof. Dr. Thierson Rosa Couto

Sumário

1 Fatorial (++)

 (++)

Dado um número inteiro n, calcule seu fatorial $n!$. O fatorial de um número é dado pela equação: $n! = n(n-1)(n-2)\ldots 1$. Por definição, $0! = 1$.

Você deve implementar a função:

```
/**
 * Funcao que calcula o fatorial de um numero n
 * @param n um numero inteiro positivo
 * @return o fatorial de n
 */
unsigned long int fat( unsigned int n);
```

Entrada

O programa deve ler um número inteiro n.

Saída

O programa deve apresentar uma linha com a mensagem: "n! = f", onde n é o número lido e f o seu fatorial.

Observações

O fatorial de um número é resultado de uma operação de produtório que pode levar a valores incrivelmente grandes. Lembre-se de usar tipos de dados apropriados ao problema proposto.

Exemplo

Entrada
2
Saída
2! = 2

Entrada
4
Saída
4! = 24

2 Fibonacci (++)

 (++)

Faça um programa que, dados os termos iniciais da sequência de Fibonacci, calcule o n-ésimo número da sequência. Uma sequência é denominada sequência de Fibonacci segue todos os seus elementos são calculados pela soma de seus dois elementos antecessores. Exemplo:

$$F_n = F_{n-1} + F_{n-2} \tag{1}$$

Para $t_1 = 1$ e $t_2 = 1$, temos:

$$F(t_1, t_2) = F(1,1) = 1,1,2,3,5,8,13,21,... \tag{2}$$

Por exemplo: o quarto e o sétimo termos da sequência $F(1,1)$ são 3 e 13 respectivamente. Você deve implementar a função:

```c
/**
 * Retorna o n-ésimo termo da sequência de Fibonacci
 * @param t1 primeiro termo da sequência
 * @param t2 segundo termo da sequência
 * @param n a posição do termo desejado da sequência
 * @return o valor do n-ésimo termo da sequência
 */
int fibonacci( int t1, int t2, int n);
```

Entrada

O programa deve ler os dois termos iniciais t_1 e t_2 e a posição n do termo a ser retornado pela função.

Saída

O programa deve apresentar uma linha com o valor do n-ésimo termo da sequência.

Exemplo

Entrada	Saída
3 8 5	30

Entrada	Saída
10 10 5	50

Entrada	Saída
2 2 19	8362

3 Número Invertido (++)

 (++)

Escreva um programa para ler um número de três dígitos e imprimir o número invertido. Seu programa deve ter uma função: separaDigitos que possui quatro parâmetros. O primeiro, é um parâmetro de entrada e corresponde a um número inteiro (com três dígitos), os demais três parâmetros são de saída, e correspondem, respectivamente ao primeiro, ao segundo e ao terceiro dígitos do número correspondente ao primeiro parâmetro. A função recebe um valor inteiro no primeiro parâmetro e devolve os dígitos que o formam nos três parâmetros seguintes.

Entrada

A entrada contém apenas um número com três dígitos. Esse número é diferente de zero e não é múltiplo de 10 ou 100.

Saída

A saída deve conter apenas uma linha com o número correspondente ao valor da entrada, com seus dígitos invertidos. Logo após o número, deve ser impresso o caractere de quebra de linha: '\n'.

Exemplos

Entrada
123
Saída
321

Entrada
987
Saída
789

4 Número perfeito (++)

 (++)

Dado um número n inteiro e positivo, dizemos que *n* é perfeito se *n* for igual à soma de seus divisores positivos diferentes de *n*. Construa um programa que leia um número inteiro *n*, apresenta a soma dos divisores de *n* e verifica se o número informado é perfeito ou não.

Escreva uma função somaDivisores que receba como parâmetro um número inteiro e retorne a soma dos divisores desse número excluindo o próprio número como divisor de si mesmo. Seu programa deve chamar a função somaDivisores para resolver o problema.

Entrada

O programa deve ler um número inteiro *n*.

Saída

O programa deve apresentar uma linha contendo o texto: "n = d1 + d2 + d3 + ... + dk = x (MENSAGEM)", onde *n* é o número lido, d_i são os divisores de *n* em ordem crescente, *x* é a soma dos divisores e MENSAGEM é a mensagem "NUMERO PERFEITO"ou "NUMERO NAO E PERFEITO".

Observações

Suponha que o usuário sempre fornecerá um número maior que 1.

Exemplo

Entrada
`6`
Saída
`6 = 1 + 2 + 3 = 6 (NUMERO PERFEITO)`

Entrada
`12`
Saída
`12 = 1 + 2 + 3 + 4 + 6 = 16 (NUMERO NAO E PERFEITO)`

5 Raízes de equações de grau 2 (++)

 (++)

Desenvolver um programa que leia os coeficientes (a, b e c) de uma equação de segundo grau e calcule as raízes da equação. O programa deve mostrar a classificação das raízes, e, quando possível, o valor das raízes calculadas.

Seu programa deve criar uma função raizesEq2Grau que tenta computar as raízes de uma equação do segundo grau. A função deve retornar 2 se existir duas raízes reais distintas entre si, ou 1 se existir uma única raiz real, ou ainda, zero se as raízes são imaginárias. A função deve ter como parâmetros de entrada os coeficientes a, b, c de uma equação de segundo grau e deve ter dois parâmetros de saída, correspondendo às raízes da equação. No caso em que a função retorna 0 (raízes imaginarias) os parâmetros de saída não são utilizados pelo seu programa.

Entrada

O programa deve ler três valores reais na entrada. O primeiro valor corresponde ao valor do coeficiente a, o segundo, do coeficiente b e o terceiro, do coeficiente c, de uma equação de segundo grau. Os três valores ocorrem em uma única linha na entrada, separados entre si por um espaço.

Saída

O programa deve imprimir uma linha contendo uma das seguintes frases, conforme for o resultado do cálculo das raízes da equação: RAIZES DISTINTAS, ou RAIZ UNICA, ou RAIZES IMAGINARIAS. No primeiro caso o programa deve imprimir uma outra linha contendo a frase X1 = x_1, onde x_1 é o valor da menor raiz encontrada para a equação. Ainda no primeiro caso, o programa deve imprimir uma terceira linha com a frase X2 = x_2, onde x_2 corresponde ao valor da segunda raiz. No segundo caso, o programa deve imprimir uma frase X1 = x_1, onde x_1 é o valor da única raiz da equação. O terceiro caso não há o que imprimir pois as raízes são imaginárias.

Observações

Dada uma equação do segundo grau do tipo $ax^2 + bx + c$, Δ (delta) $= b^2 - 4ac$. Se $\Delta = 0$, a raiz da equação é ÚNICA. Se $\Delta < 0$. As raízes da equação são IMAGINÁRIAS. Se $\Delta > 0$, então há duas RAÍZES DISTINTAS para a equação. A fórmula geral para computar as raízes de uma equação do segundo grau é a fórmula de Báskara, dada por:

$$x = \frac{-b \pm \sqrt{\Delta}}{2a}$$

Exemplo

A seguir são mostrados três exemplos distintos de entrada, e suas correspondentes saídas, entretanto, existe apenas uma linha de entrada para esse problema.

Entrada
`2 12 10`
Saída

`RAIZES DISTINTAS`
`X1 = -1.00`
`X2 = -5.00`

Entrada
2 12 18
Saída
RAIZ UNICA X1 = -3.00

Entrada
15 17 89
Saída
RAIZES IMAGINARIAS

6 Cálculo da raiz quadrada (+++)

 (+++)

Os Babilônios utilizavam um algoritmo para aproximar uma raiz quadrada de um número qualquer, da seguinte maneira:

Dado um número n, para calcular $r = \sqrt{n}$ assume-se uma aproximação inicial $r_0 = 1$ e calcula-se r_k para $k = 1, \ldots, \infty$ até que $r_k^2 \approx n$. O algoritmo deve realizar a aproximação enquanto $|n - r_k^2| > e$. O método babilônico é dado pela seguinte equação:

$$r_k = \frac{r_{k-1} + \frac{n}{r_{k-1}}}{2} \tag{3}$$

Funções a serem implementadas:

```
1
2  /**
3   * Função que calcula a raiz quadrada de n.
4   * @param n um numero real qualquer
5   * @return a raiz quadrada de n
6   */
7  double raiz( double n );
8
9  /**
10  * Valor absoluto de um numero qualquer
11  * @param n um número real qualquer
12  * @return o valor absoluto de n
13  */
14 double absoluto( double n );
```

Entrada

O programa deve ler um número **double** n, cuja raiz quadrada deseja-se obter, e o erro e que deverá ser considerado pelo algoritmo.

Saída

A saída deve apresentar cada iteração do algoritmo, sendo cada linha composta pelo valor aproximado da raiz quadrada de n com 9 casas decimais, seguido do erro, também com 9 casas decimais.

Exemplo

Entrada
2 0.00001
Saída
r: 1.500000000, err: 0.250000000 r: 1.416666667, err: 0.006944444 r: 1.414215686, err: 0.000006007

7 Escovando *bits* (+++)

 (+++)

Faça um programa que leia um número real (double), o converta para variáveis dos seguintes tipos de dados: `unsigned char`, `unsigned short`, `unsigned int`, `float`, `double` e apresentes os *bits* de cada *byte* de cada variável na mesma sequência da lista. Você deve implementar a função:

```
/**
 * Imprime os bits dos n bytes endereçados por end_byte.
 * @param end_byte endereço do primeiro byte a ser impresso
 * @param quantidade de bytes a serem impressos
 */
void print_bytes( const void * end_byte, int n );
```

Entrada

Um número real com dupla precisão.

Saída

Cinco linhas contendo os *bits* dos *bytes* de cada variável, separados por espaços.

Exemplo

Entrada	Saída
127	01111111 01111111 00000000 01111111 00000000 00000000 00000000 00000000 00000000 11111110 01000010 00000000 00000000 00000000 00000000 00000000 11000000 01011111 01000000

Entrada	Saída
256	00000000 00000000 00000001 00000000 00000001 00000000 00000000 00000000 00000000 10000000 01000011 00000000 00000000 00000000 00000000 00000000 00000000 01110000 01000000

Entrada	Saída
0.3	00000000 00000000 00000000 00000000 00000000 00000000 00000000 10011010 10011001 10011001 00111110 00110011 00110011 00110011 00110011 00110011 00110011 11010011 00111111

8 Escovando *bytes* (+++)

 (+++)

Faça um programa que leia um número real (double), o converta para variáveis dos seguintes tipos de dados: `unsigned char`, `unsigned short`, `unsigned int`, `float`, `double` e apresentes o conteúdo de cada *byte* de cada variável na mesma sequência da lista.

Entrada

Um número real com dupla precisão.

Saída

Cinco linhas contendo o valores dos *bytes* de cada variável, impressos como "%u"e separados por ’,’.

Exemplo

Entrada	Saída
127	127, 127,0, 127,0,0,0, 0,0,254,66, 0,0,0,0,0,192,95,64,

Entrada	Saída
256	0, 0,1, 0,1,0,0, 0,0,128,67, 0,0,0,0,0,0,112,64,

Entrada	Saída
0.3	0, 0,0, 0,0,0,0, 154,153,153,62, 51,51,51,51,51,51,211,63,

9 José (+++)

 (+++)

João tem um irmão mais novo, José, que começou a ir à escola e já está tendo problemas com números. Para ajudá-lo a pegar o jeito com a escala numérica, sua professora escreve dois números de três dígitos e pede a José para comparar esses números. Mas em vez de interpretá-los com o dígito mais significativo à esquerda, ele deve interpretá-lo com o dígito mais significativo à direita. Ele tem que dizer à professora qual o maior dos dois números. Escreva um programa que irá verificar as respostas de José.

Escreva uma função inverte que receba como parâmetro um número inteiro (de três dígitos) e retorne o número invertido. Sugestão: aprenda a aproveitar código! Copie a função SeparaDigitos que você escreveu para o problema "Número Invertido" e cole no seu programa. Faça a função inverte chamar a função SeparaDigitos para te auxiliar a computar o número invertido. O seu programa deve chamar a função inverte quantas vezes for necessário para resolver o problema.

Entrada

A entrada conterá um inteiro T , o número de casos de testes, e, para cada caso de teste, uma única linha com dois números de três dígitos, A e B, os quais não serão iguais e não conterão zeros.

Saída

A saída deve conter, numa linha para cada caso de teste, com o maior dos números na entrada, comparados como descrito no enunciado da tarefa. O número deve ser escrito invertido, para mostrar a José como ele deve lê-lo.

Exemplo

Entrada
3 734 893 221 231 839 237
Saída
437 132 938

10 Próxima potência (+++)

 (+++)

Escreva um programa que leia dois números inteiros, n e p, e calcule o número mais próximo de n que seja uma potência inteira de p. Por exemplo, para o número $n = 5$ e $p = 2$ o número mais próximo de 5 que é uma potência inteira de 2 é $4 = 2^2$. Para $n = 10$ e $p = 3$, o número mais próximo é $2^3 = 8$.

Você deve implementar a função:

```
/**
 * Função que calcula a potencia de p mais próximo a n.
 * @param n valor inteiro
 * @param p valor da potencia
 * @return retorna o valor da potencia mais proxima.
 */
int next_power( int n, int p );
```

Entrada

O programa deve ler os números inteiros n e p.

Saída

O programa deve apresentar uma linha com o valor da potência mais próxima a n no formato: "n -> k ^$p = x$", onde k é a base da potência e x o valor mais próximo de n.

Exemplo

Entrada	Saída
5 2	5 -> 2^2 = 4

Entrada	Saída
10 3	10 -> 2^3 = 8

Entrada	Saída
40 4	40 -> 2^4 = 16

11 Valor em Notas e Moedas (+++)

 (+++)

Escreva um algoritmo par ler um valor em reais e calcular qual o menor número possível de notas de $R 100, $R 50, $R 10 e moedas de $R 1 em que o valor lido pode ser decomposto. O programa deve escrever a quantidade de cada nota e moeda a ser utilizada.

Você deve escrever uma função converteEmNotasMoedas que possui 5 parâmetros. O primeiro parâmetro corresponde ao valor inteiro a ser convertido em notas e moedas, o segundo parâmetro corresponde ao número de notas de 100, o terceiro, corresponde ao número de notas de 50, o quarto, corresponde ao número de notas de dez e o quinto parâmetro corresponde ao número de moedas de 1 Real. Seu programa deve chamar essa função para resolver o problema proposto.

Entrada

O programa deve ler uma única linha na entrada, contendo um valor em Reais. Considere que somente um número inteiro seja fornecido como entrada.

Saída

O programa deve imprimir quatro frases, uma em cada linha: NOTAS DE 100 = X, NOTAS DE 50 = Y, NOTAS DE 10 = Z, MOEDAS DE 1 = W, onde X, Y, Z e W correspondem às quantidades de cada nota ou moeda necessárias para corresponder ao valor em Reais dado como entrada. Após cada quantidade, o programa deve imprimir um caractere de quebra de linha: '\n'.

Exemplo

Entrada
`46395`
Saída

`NOTAS DE 100 = 463`
`NOTAS DE 50 = 1`
`NOTAS DE 10 = 4`
`MOEDAS DE 1 = 5`

12 Cálculo de PI (++++)

 (++++)

Na história da ciência, muitas foram as tentativas de se encontrar o número π com a maior precisão possível. Uma dessas tentativas foi a do matemático John Wallis, que desenvolveu a série infinita da Equação 4 em 1655.

$$\frac{2}{1}\cdot\frac{2}{3}\cdot\frac{4}{3}\cdot\frac{4}{5}\cdot\frac{6}{5}\cdot\frac{6}{7}\cdot\frac{8}{7}\cdot\frac{8}{9}\cdots = \frac{\pi}{2} \tag{4}$$

Faça um programa que calcula o valor de *pi* usando a série proposta, permitindo o usuário definir a quantidade de termos da série.

Você deve implementar a função:

```c
/**
 * Função que calcula o valor de pi usando a série proposta por John Wallis
 * @param n quantidade de termos da série
 * @return o valor aproximado da constante pi
 */
double compute_pi( int n );
```

Entrada

O programa deve ler a quantidade de termos *n*.

Saída

O programa deve apresentar uma linha com o valor de π com 12 casas decimais.

Observações

Use precisão dupla (`double`) para os números reais.

Exemplo

Entrada	Saída
30	3.091336888596

Entrada	Saída
500	3.138458897672

Entrada	Saída
10000	3.141435593590