Lista 05/06/2017

Prof. Dr. Gustavo Teodoro Laureano
Profa. Dra. Luciana Berretta
Prof. Dr. Thierson Rosa Couto

Sumário

1 *Swap* via ponteiros (+) **2**

2 Tamanho de ponteiros (+) **3**

3 Escovando *bits* (+++) **4**

4 Escovando *bytes* (+++) **6**

5 *string to int* (+++) **7**

6 Turismo (+++) **8**

7 Números Racionais (++++) **9**

8 Ordena palavras (++++) **11**

9 Raízes de equações quadradas (++++) **12**

10 *string to double* (++++) **14**

1 *Swap* via ponteiros (+)

 (+)

Faça um programa que leia dois inteiros "`int a, b;`", troque o valor dessas variáveis via um único ponteiros e apresente o resultado na tela. O programa deve acessar as variáveis a e b somente via ponteiro. Qualquer comando de atribuição às variáveis a ou b, ou seja, aqueles que comecem com "`a= `" e "`b= `", são proibidos.

Entrada

Dois números inteiros.

Saída

Uma linha contendo dois inteiros separados por um espaço.

Exemplo

Entrada	Saída
1 2	2 1

2 Tamanho de ponteiros (+)

 (+)

Faça um programa que apresente a quantidade de *bytes* de variáveis e de seus respectivos ponteiros. O programa deve ler um número inteiro que especifica o tipo de dado desejado e apresentar a informação de tamanho em seguida. Para esse programa, a seguinte associação deve ser seguida:

1. `char`
2. `short`
3. `int`
4. `long int`
5. `float`
6. `double`
7. `long double`

Entrada

Um número inteiro.

Saída

A saída são duas linhas no formato: ”<tipo>: *k*\n<tipo>*: *k*”, onde <tipo> é o nome do tipo de dado e *k* é a quantidade de *bytes*.

Exemplo

Entrada	Saída
2	short: 2 short*: 8

3 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 dubla 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

4 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 dubla 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,

5 *string to int* (+++)

 (+++)

Faça um programa que leia um número inteiro fornecido como uma *string* e o converta para um `long int`. Você deve implementar a função:

```
/**
 * Converte a string str para o valor inteiro correspondente.
 * @param str string contendo um número inteiro
 * @return o número inteiro correspondente
 */
long int string2int( const char * str );
```

Entrada

O programa deve ler uma sequência de *strings* contento um número inteiro, de no máximo 128 caracteres, usando o comando: `scanf("%s", str);`, até atingir o final do arquivo, ou seja, enquanto `( scanf("%s", str) != EOF )`.

Saída

A saída é composta por linhas linha contendo o número inteiro e o seu dobro impressos usando o comando `printf("%ld %ld\n", n, n*n);`, onde *n* é o número convertido.

Exemplo

Entrada	Saída
1 -2 3 -4	1 2 -2 -4 3 6 -4 -8

Entrada	Saída
15	15 30

Entrada	Saída
-1234	-1234 -2468

6 Turismo (+++)

 (+++)

Os acessos e distâncias entre 6 cidades são listadas pela Tabela 1. Cada célula da tabela mostra a distância, em quilômetros, entre a cidade de cada linha com as cidades de cada coluna. O caracter '-' indica que não há acesso entre as cidades, partindo da cidade da linha correspondente.

Tabela 1: Tabela de distâncias e acessos entre cidades.

	Cárceres	Bugres	Cuiabá	Várzea	Tangará	Lacerda
Cárceres	0	63	210	190	-	190
Bugres	63	0	160	150	95	10
Cuiabá	210	160	0	10	1	10
Várzea	190	150	10	0	10	20
Tangará	10	95	7	21	0	80
Lacerda	190	2	-	41	80	0

Tendo conhecimento dessa tabela, uma agencia de turismo gostaria de ter um programa que, dada uma rota, verifique se a rota é válida e que calcule e apresente a distância da rota fornecida.

As cidades Cárceres, Burgres, Várzea, Tangará e Lacerda são representadas pelos números $0, 1, 2, 3, 4, 5$ respectivamente. Desse modo, uma rota pode ser representada por um vetor de inteiros que indica o translado entre as cidades listadas.

Por exemplo, o vetor {1, 2, 3} indica que a rota válida que inicia pela cidade de Bugres, passa pela cidade de Cuiabá e termina em Várzea, totalizando 170 km. Uma rota é inválida se a sequência do vetor atinge um elemento da matriz com o caracter '-'.

Entrada

O programa deve ler um número inteiro N, correspondente ao tamanho da rota, sendo $0 < N \leq 100$, e um vetor de inteiros com N elementos.

Saída

O programa deve apresentar a distância total da rota percorrida ou a mensagem "rota invalida!"caso a rota seja inválida.

Exemplo

Entrada	Saída
3 1 2 3	170

Entrada	Saída
3 0 4 1	rota invalida!

7 Números Racionais (++++)

 (++++)

Número racional é todo o número que pode ser representado por uma razão (ou fração) entre dois números inteiros. O conjunto dos números racionais (representado por $\mathbb{Q}$) é definido por:

$$\mathbb{Q} = \{\frac{a}{b} \mid a \in \mathbb{Z}; b \in \mathbb{Z}^*\} \tag{1}$$

Em outras palavras, o conjunto dos números racionais é formado por todos os quocientes de números inteiros a e b, em que b é não nulo. O uso da letra "Q"é derivado da palavra inglesa *quotient*, cujo significado é quociente, já que a forma de escrever um número racional é o quociente de dois números inteiros. São exemplos de números racionais:

$$\frac{5}{8}; 1; 2; \frac{1}{3}; \text{-}8; -\frac{2}{5};$$

Faça um programa que defina um novo tipo de dado através de uma estrutura chamada `tRacional`, com os componentes inteiros a e b, conforme a definição anterior. Escreva as seguintes funções para operar sobre o novo tipo:

```
1
2  /**
3   * Calcula o MDC de x e y
4   * @param x
5   * @param y
6   * @return
7   */
8  int MDC(int x, int y);
9
10
11 /**
12  * Recebe dois inteiros a e b e retorna o racional
13  * @param a numerador
14  * @param b denominador
15  * @return
16  */
17 struct tRacional racional(int a, int b);
18
19 /**
20  * Recebe um racional e retorna o seu negativo (-r).
21  * @param r numero racional
22  * @return
23  */
24 struct tRacional negativo(struct tRacional r);
25
26 /**
27  * Recebe dois racionais e retorna a adição de ambos (r1 + r2).
28  * @param r1 fator esquerdo da soma
29  * @param r2 fator direito da soma
30  * @return
31  */
32 struct tRacional soma(struct tRacional r1, struct tRacional r2);
33
34 /**
35  * Recebe dois racionais e retorna o produto de ambos (r1 * r2).
36  * @param r1 primeiro fator do produto
37  * @param r2 segundo fator do produto
38  * @return
```

```
 */
struct tRacional mult(struct tRacional r1, struct tRacional r2);

/**
 * Recebe dois racionais e retorna o quociente de ambos (r1/r2).
 * @param r1 numerador
 * @param r2 denominador
 * @return
 */
struct tRacional div(struct tRacional r1, struct tRacional r2);


/**
 * Recebe um racional e reduz a fração ao máximo.
 * @param r o número racional a ser reduzido
 */
void reduzFracao( struct tRacional * r);
```

Entrada

A entrada consiste de várias linhas no seguinte formato: a b operação a b, onde: $-10000 \leq a \leq 10000$, $0 < b \leq 10000$. E operação será um dos seguintes caracteres: +, -, * ou /. A entrada termina com `EOF`.

Saída

A saída consiste de várias linhas com o resultado da operação sobre os racionais. É necessário reduzir a fração ao máximo. Após a impressão do último resultado quebre uma linha.

Exemplo

Entrada	Saída
1 5 + 2 10 2 3 + 5 7 17 24 - 5 6 8 3 * 4 3 -5 2 * 4 3 8 3 / 4 3 2 3 / 2 5	2 5 29 21 -1 8 32 9 -10 3 2 1 5 3

8 Ordena palavras (++++)

 (++++)

Faça um programa que leia no máximo 100 palavras de no máximo 32 caracteres e as apresente na tela de forma ordenada crescente. Você deverá implementar as funções:

```c
/**
 * Compara duas strings.
 * @param s1
 * @param s2
 * @return s1-s2
 */
int string_cmp(const unsigned char * s1, const unsigned char * s2);

/**
 * Troca o conteúdo das strings s1 e s2
 * @param s1
 * @param s2
 */
void string_swap(char * s1, char * s2);
```

Entrada

O programa deve ler um número inteiro N, correspondente à quantidade de palavras a serem lidas e N palavras.

Saída

O programa deve apresentar N linhas com as palavras ordenadas de modo crescente.

Exemplo

Entrada	Saída
3 tempo lua cama	cama lua tempo

9 Raízes de equações quadradas (++++)

 (++++)

Uma equação de grau 2 tem o formato $ax^2 + bx + c = 0$, onde a, b, c são os coeficientes da equação e x a variável independente. Faça uma função que calcule, imprima e retorne as as raízes de uma equação de segundo grau. Para isso, crie as estruturas `Complex` e `RaizEqu2`, sendo Complex formada por duas variáveis do tipo **`float`**, representando a parte real e imaginária; e `RaizEqu2`, contendo duas variáveis `Complex`, representando as duas raízes da equação. Você deve implementar as funções:

```
/**
 * Função que calula as raízes de uma equação de segundo grau.
 *
 * @param a coeficiente quadrado
 * @param b coeficiente linear
 * @param c constante
 * @return retorna uma estrutura RaizEqu2 com dois números complexos
 */
struct RaizEqu2 calcula_raiz_equ_2( float a, float b, float c);
```

```
/**
 * Imprime numeros complexos na saída padrão do sistema. A impressão segue o formato
 * a + bi, onde a é a parte real e b a imaginária. Os valores são apresentados
 * somente se forem diferente de zero. No caso de a e b forem zero, o valor 0 é
 * apresentado. Se o valor de b for 1 ou -1, somente o caracter i ou -i é
   apresentado.
 *
 * @param c Numero complexo a ser impresso.
 */
void complex_print(struct Complex c);
```

As soluções das raízes desse tipo de equação são dadas pela fórmula de Bhaskara 2.

$$x_1 = \frac{-b + \sqrt{b^2 - 4ac}}{2a} \quad x_2 = \frac{-b - \sqrt{b^2 - 4ac}}{2a} \tag{2}$$

Entrada

O programa deve ler 3 valores reais, correspondentes aos coeficientes a, b, c.

Saída

O programa deve apresentar as raízes x_1 ex_2 em linhas separadas. Para os valores positivos deve-se omitir o sinal $+$. A parte imaginária deve ser apresentada seguida do caracter i. Os valores iguais a zero devem ser omitidos. Apresenta-se o valor 0 quando a raíz possui a parte real e imaginária zeradas. Todos os valores devem ser apresentados com 2 casas decimais.

Exemplo

Entrada	Saída
1 0 0	x1 = 0.00 x2 = 0.00

Entrada	Saída
1 2 1	x1 = -1.00 x2 = -1.00

Entrada	Saída
1 0 1	x1 = i x2 = -i

Entrada	Saída
1 5 4	x1 = -1.00 x2 = -4.00

Entrada	Saída
1 -2 2	x1 = 1.00+i x2 = 1.00-i

10 *string to double* (++++)

 (++++)

Faça um programa que leia um número real fornecido como uma *string* e o converta para um **double**. Um número real pode ter ou não o caracter '.' para separar a parte inteira da fracionária. Você deve implementar a função:

```
/**
 * Converte a string str para o valor real correspondente.
 * @param str string contendo um número real
 * @return o número inteiro correspondente
 */
double string2double( const char * str );
```

Entrada

O programa deve ler uma sequência de *strings* contento um número real, de no máximo 128 caracteres, usando o comando: `scanf("%s", str);`, até atingir o final do arquivo, ou seja, enquanto `( scanf("%s", str) != EOF )`.

Saída

A saída é composta por linhas linha contendo o número real e o seu dobro impressos usando o comando `printf("%.3lf %.3lf\n", n, n*n);`, onde *n* é o número convertido.

Exemplo

Entrada	Saída
1.01 -2.2 3.5 -4.0	1.010 2.020 -2.200 -4.400 3.500 7.000 -4.000 -8.000

Entrada	Saída
15	15.000 30.000

Entrada	Saída
0.5	0.500 0.250

Entrada	Saída
10.02	10.020 20.040

Entrada	Saída
-1234	-1234.000 -2468.000