Referência do Arquivo funcoesTrabalho.h

```
#include <stdio.h>
#include <math.h>
```

Ir para o código-fonte desse arquivo.

Definições e Macros

#define	**_VELOCIDADE_LUZ_** 299792458

Funções

double	**expressaoFatorLorentz** (double velocidadeObjeto) Função que cria a expressão que será usada para calcular o fator de lorentz.
double	**expressaoVelocidade** (double comprimentoCeleiro, double comprimentoVara) Função que cria a expressão que será usada para a velocidade para que sofra uma dada contração.
double	**funcaoCalculaRaiz** (double estimativa) Função que cria a função a ser usada no método de Newton-Rapshon.
double	**derivadaFuncaoRaiz** (double estimativa, double h) Função que calcula a derivada da função usada no método de Newton-Rapshon.
double	**calculoRaizQuadrada** (double expressao, double tolerancia, double estimativa, int maxInteracoes) Função que calcula a raiz quadrada de qualquer função com base no método de Newton-Rapshon.
double	**velocidadeNecessaria** (double expressao, double tolerancia, double estimativa, int maxInteracoes) Função que calcula a velocidade nescessária para um dado comprimento.
double	**comprimentoObjeto** (double comprimentoOriginal, double fatorLorentz) Função que calcula o comprimento do meu objeto em um cenário relativístico.
void	**simultaneidade** (double comprimentoVara, double comprimentoCeleiro, double comprimentoVaraOriginal, double comprimentoCeleiroOriginal, double velocidade) Função mostra se existe ou não simultaneidade.

Definições e macros

◆ _VELOCIDADE_LUZ_

#define _VELOCIDADE_LUZ_ 299792458

Funções

- calculoRaizQuadrada()

```
double calculoRaizQuadrada ( double  expressao,
                             double  tolerancia,
                             double  estimativa,
                             int     maxInteracoes
                           )
```

Função que calcula a raiz quadrada de qualquer função com base no método de Newton-Rapshon.

Está função recebe quatro valores e os utiliza para calcular. A raiz quadrada de qualquer valor ou função. Pelo método de Newton-Rapshon.

Parâmetros

expressao	Variável que armazena a expressão que será calculada
tolerancia	Variável que armazena o quão pequeno a derivada deve ser
estimativa	Variável que armazena a estimativa da raiz quadrada
maxInteracoes	Variável que armazena o número máximo de interações o programa deve fazer

Retorna

raiz da expressão

Exemplo de uso:

```
double x = 1.434;
double exp = 2 + x;
int inte = 500;
double tol = 5e-5;
double raiz = calculoRaizQuadrada(exp, tol, x, inte);
```

```
119
    {
120     double x = estimativa;
121     int i;
122
123     for (i = 0; i < maxInteracoes; i++) {
124         double fx = funcaoCalculaRaiz(x) - expressao;
125         double dfx = derivadaFuncaoRaiz(x, 0.001);
126
127         x = x - fx/dfx;
128
129         if(fabs(fx) < tolerancia)
130             break;
131     }
132
133     return x;
134 }
```

◆ comprimentoObjeto()

```
double comprimentoObjeto ( double  comprimentoOriginal,
                           double  fatorLorentz
                         )
```

Função que calcula o comprimento do meu objeto em um cenário relativístico.

Está função recebe dois valores e com base neles. É calculada o comprimento de um objeto dado o fator de lorentz. Realizando o produto do comprimento pelo fator de lorentz.

Parâmetros

comprimentoOriginal Variável que armazena o comprimento original do meu objeto

fatorLorentz Variável que armazena o fator de Lorentz

Retorna

comprimento

```
double comp = 32.5;
double fatLor = 0.387;
double compRel = comprimentoObjeto(comp, fatLor);
```

```
184                                                                 {
185     double comprimento = comprimentoOriginal / fatorLorentz;
186     return comprimento;
187
188 }
```

◆ derivadaFuncaoRaiz()

```
double derivadaFuncaoRaiz ( double  estimativa,
                            double  h
                          )
```

Função que calcula a derivada da função usada no método de Newton-Rapshon.

Está função recebe dois valores e com base neles. É calculada a derivada da função utilizando o. Método da diferença central.

Parâmetros

estimativa Variável que armazena a estimativa da derivada

h Variável que armazena o incremento entre os pontos

Retorna

derivada da função

Exemplo de uso:

```
double est = 1.0;
double h = 0.05;
double derivada = derivadaFuncaoRaiz(est, h);
```

```
88                                                      {
89      double funcaoSoma = funcaoCalculaRaiz(estimativa + h);
90      double funcaoSub = funcaoCalculaRaiz(estimativa - h);
91      double derivada = (funcaoSoma - funcaoSub) / (2 * h);
92
93      return derivada;
94  }
```

◆ expressaoFatorLorentz()

double expressaoFatorLorentz (double velocidadeObjeto)

Função que cria a expressão que será usada para calcular o fator de lorentz.

Está função recebe a velocidade que o meu objeto está e a utilizando. cria a expressão a ser calculado do fator de lorentz e a retorna.

Parâmetros

velocidadeObjeto Variável que armazena a velocidade a qual o objeto se encontra

Retorna

A expressão

Exemplo de uso:

```
double velocidade = 56732.0;
double expressao = expressaoFatorLorentz(velocidade);
```

```
21                                                      {
22      double razaoLorentz = 1/(1 - (pow(velocidadeObjeto,2) / pow(_VELOCIDADE_LUZ_,
        2)));
23      return razaoLorentz;
24  }
```

◆ expressaoVelocidade()

```
double expressaoVelocidade ( double  comprimentoCeleiro,
                             double  comprimentoVara
                           )
```

Função que cria a expressão que será usada para a velocidade para que sofra uma dada contração.

Está função recebe dois valores de comprimento. E retorna a expressão para calcular a velocidade.

Parâmetros

comprimentoCeleiro Variável que armazena o comprimento do celeiro

comprimentoVara Variável que armazena o comprimento da vara

Retorna

A expressão

Exemplo de uso:

```
double compCel = 20.0;
double compVar = 34.2;
double expressao = expressaoVelocidade(compCel, compVar);
```

```
44                                                                  {
45      double velocidade = 1 - pow(comprimentoCeleiro/comprimentoVara, 2);
46      return velocidade;
47  }
```

◆ funcaoCalculaRaiz()

double funcaoCalculaRaiz (double estimativa)

Função que cria a função a ser usada no método de Newton-Rapshon.

Está função recebe um valor de estimativa e calcula a função x² a retornando-a

Parâmetros

estimativa Variável que armazena a estimativa da raiz

Retorna

x²

Exemplo de uso:

```
double x = 1.0;
double func = funcaoCalculaRaiz(x);
```

```
64                                                  {
65      double funcaoRaiz = pow(estimativa, 2);
66      return funcaoRaiz;
67  }
```

◆ simultaneidade()

```
void simultaneidade ( double  comprimentoVara,
                      double  comprimentoCeleiro,
                      double  comprimentoVaraOriginal,
                      double  comprimentoCeleiroOriginal,
                      double  velocidade
                    )
```

Função mostra se existe ou não simultaneidade.

Está função recebe cinco valores e com base eles. É calculada a distancia da ponta esquerda e direita da vara. Também a distancia da porta esquerda e da porta direita. E calcula o tempo nescessário para a ponta esquerda atingir a porta esquerda. E calcula o tempo nescessário para a ponta direita atingir a porta direita. Depois de calculado ela mostra se há ou não simultaneidade.

Parâmetros

comprimentoVara	Variável que armazena o comprimento da vara dada uma certa velocidade
comprimentoCeleiro	Variável que armazena o comprimento do celeiro dada uma certa velocidade
comprimentoVaraOriginal	Variável que armazena o comprimento original da vara
comprimentoCeleiroOriginal	Variável que armazena o comprimento original do celeiro
velocidade	variável que armazena a velocidade que o objeto se encontra

Retorna

none

```
double compVara = 10;
double compVaraOri = 16;
double compCelOri = 10;
double compCel = 6.25;
double vel = 234025312.518117;
simultaneidade(compVara, compCel, compVaraOri, compCelOri, vel);
```

```
217
    {
218
219     // Simultaneidade Vara
220     double distanciaPontaDireita = 10.0 + comprimentoCeleiroOriginal;
221     double distanciaPontaEsquerda = 10.0 + comprimentoVara;
222
223     double tempoDireitaVara = distanciaPontaDireita / velocidade;
224     double tempoEsquerdaVara = distanciaPontaEsquerda / velocidade;
225
226     // Simultaneidade Celeiro
227     double distanciaPortaDireita = 10 + comprimentoCeleiro;
228     double distanciaPortaEsquerda = 10 + comprimentoVaraOriginal;
229
230     double tempoDireitaCeleiro = distanciaPortaDireita / velocidade;
231     double tempoEsquerdaCeleiro = distanciaPortaEsquerda / velocidade;
```

```
232
233     printf("\n-==-==-==-==-==-==-==-==-==-==-==-==-\n");
234     printf("      SIMULTANEIDADE VARA");
235     printf("\n-==-==-==-==-==-==-==-==-==-==-==-==-\n");
236
237     printf("O tempo que a ponta direita da vara alcanca a porta direita do celeiro:
    %.12lf s\n", tempoDireitaVara);
238     printf("O tempo que a ponta esquerda da vara alcanca a porta esquerda do celeiro:
    %.12lf s\n", tempoEsquerdaVara);
239     printf("Existe simultaneidade\n");
240
241     printf("\n-==-==-==-==-==-==-==-==-==-==-==-==-\n");
242     printf("      SIMULTANEIDADE CELEIRO");
243     printf("\n-==-==-==-==-==-==-==-==-==-==-==-==-\n");
244
245     printf("O tempo que a porta direita do celeiro alcanca a ponta direita da vara:
    %.12lf s\n", tempoDireitaCeleiro);
246     printf("O tempo que a porta esquerda do celeiro alcanca a ponta esquerda da vara:
    %.12lf s\n", tempoEsquerdaCeleiro);
247     printf("Nao ha simultaneidade");
248
249 }
```

◆ velocidadeNecessaria()

```
double velocidadeNecessaria ( double  expressao,
                              double  tolerancia,
                              double  estimativa,
                              int     maxInteracoes
                            )
```

Função que calcula a velocidade nescessária para um dado comprimento.

Está função recebe quatro valores e com base neles. É estimada a velocidade para que seja feita uma dada contração. Realizando o produto da raiz obtida pela função que calcula raiz. pela velocidade da luz.

Parâmetros

expressao	Variável que armazena a expressao da velocidade
tolerancia	Variável que armazena a tolerancia para ser usada na função que calcula a raiz
estimativa	Variável que armazena a estimativa da velocidade
maxInteracoes	Variável que armazena o número de interações que devem ser feitas na função que calcula a raiz

Retorna

velocidade

Exemplo de uso:

```
double est = 1.2;
double exp = 2 + x² + 3;
double tol = 2e-8;
int max = 1000;
double vel = velocidadeNecessaria(exp,tol, est, max);
```

```
160
    {
161     double raiz = calculoRaizQuadrada(expressao, tolerancia, estimativa,
    maxInteracoes);
162     double velocidade = _VELOCIDADE_LUZ_ * raiz;
163     return velocidade;
164 }
```