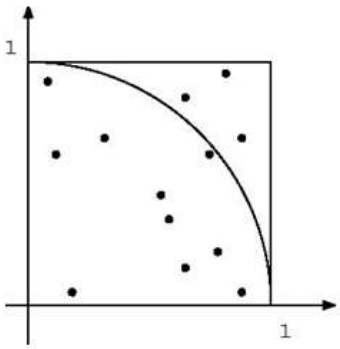
Instruções:

- Leia toda a avaliação antes de começar a responde-la.
- A avaliação ocorre em modelo de atividade assíncrona.
- O prazo para a entrega é de 6 horas, impreterivelmente, a contar após o momento do envio do exame por parte do professor.
- As respostas devem ser entregues pelo aluno em um único arquivo programado e compilável, intitulado *"ESCREVA_0_SEU_NOME.c"*. A não entrega em arquivo ".c" ou a não compilação do arquivo acarretará na não correção por parte do professor e, consequentemente, na atribuição de nota 0 (zero).
- Não serão aceitos comandos com funções nem operadores ainda não estudados no Curso.
- Após a entrega das respostas por parte do aluno, o professor tem a prerrogativa de arguir o aluno a respeito da solução encontrada para as suas respostas, em modelo de atividade síncrona.

Descrição do problema proposto:

O método de Monte Carlo é um método de soluções para uma variedade de problemas usando experimentos em computadores. Por exemplo, é possível calcular o valor de π usando o método de Monte Carlo. Para isso, é possível realizar um experimento simples tomando um quarto de círculo inscrito em um quadrado de lado 1, como mostra a figura a seguir.



A área do círculo pode ser obtida através da expressão $A = \pi \times r^2$. Nessa equação, A é a área do círculo, π é a constante próxima a 3,141592 e r é o raio do círculo. Dessa forma, a área de um quarto de círculo é expressa por:

$$\frac{\pi \times r^2}{4}$$

Ao considerar o raio de tamanho 1, a área do quarto de círculo passa a valer $\frac{\pi}{4}$.

Considere uma pessoa muito paciente atirando milhares de dardos neste quadrado. Ao final do experimento, essa pessoa conta o número de dardos localizados dentro do quarto de círculo. A razão entre o número de dardos dentro do quarto de círculo para o número total de dardos tende a ser próximo de $\frac{\pi}{4}$.

Para simular esse atirador, um programa de computador pode gerar uma quantidade muito grande de pares de números aleatórios entre 0 e 1, os quais servirão como as coordenadas *x* e *y* do ponto onde o dardo atinge o quadrado.

Para que um ponto (*x*, *y*) esteja dentro de um círculo de raio *r*, é preciso que a distância desse ponto ao centro do círculo seja menor ou igual a *r.*

Falta resolver como gerar números aleatórios. Em C, podemos usar a função *rand()* que retorna um número entre 0 e *RAND_MAX*. *RAND_MAX* é uma constante que representa o maior inteiro que se pode armazenar em 32 *bits*. Tanto a função *rand()* quanto a constante *RAND_MAX* estão definidas no arquivo de cabeçalho *stdlib.h*.

Este gerador precisa de uma semente para gerar a sequência de números aleatórios. O manual da linguagem C recomenda que se use, apenas antes da primeira chamada à função *rand()*, o seguinte comando:

srand(time(NULL));

A função *srand* é a responsável por determinar, através do seu parâmetro, qual semente será usada. Quando o seu parâmetro é definido por *time(NULL)*, a semente usada é um número inteiro correspondente à hora atual do sistema operacional. A função *time()* pertence ao arquivo de cabeçalho *time.h* e a constante *NULL* pertence ao *stdlib.h*.

Após essa explicação, segue um programa de computador compilável em linguagem C. Esse programa já dispõe de uma função chamada *pi_MonteCarlo* que gera pontos a partir de uma semente geradora de números aleatórios para uma quantidade de dardos e os armazena em um arquivo binário do tipo de registro *TPonto*. O tipo *TPonto* também vem definido nesse programa.

```
#include<stdio.h>
#include<math.h>
#include<stdlib.h>
struct _TPonto_
{
  float x;
  float y;
};
typedef struct _TPonto_ TPonto;

void  pi_MonteCarlo (int semente, int dardos, FILE * arquivo);
void  transfereDeArquivoParaVetor (FILE * arquivo, TPonto v[], int tamanho);
float distancia (TPonto p);
int   dardosNoQuartoDeCirculo (TPonto v[], int tamanho, float raio);
void  criaLog (FILE * arquivo, TPonto v[], int pontosNoAlvo, int totalDePontos);

main()
{
  FILE * arquivo;
  TPonto * pontos;
  int totalDeDardos, dardosNoAlvo, semente;
```

```
 printf("QUANTIDADE DE DARDOS: ");
 scanf("%d", &totalDeDardos);

 pontos = (TPonto *)malloc(sizeof(TPonto[totalDeDardos]));

 semente = time(NULL);

 pi_MonteCarlo (semente, totalDeDardos, arquivo);
 transfereDeArquivoParaVetor (arquivo, pontos, totalDeDardos);
 dardosNoAlvo = dardosNoQuartoDeCirculo(pontos, totalDeDardos, 1.0);
 criaLog(arquivo, pontos, dardosNoAlvo, totalDeDardos);

 free(pontos);
}

//Gera pontos em uma área correspondente a um quadrado de lado 1
//com vértices nos pontos (0,0) , (0,1) , (1,1) e (1,0).
//Grava esses pontos em um arquivo binário.
void pi_MonteCarlo (int semente, int dardos, FILE * arquivo)
{
 TPonto p;
 int i;

 //Abre um arquivo binário para escrita.
 arquivo = fopen("pontos.dat", "wb+");

 //Estabelece uma raiz para a geração de números aleatórios.
 srand(semente);
 for (i=0; i<dardos; i++)
 {
  //Sorteia valores de 0 a 1 para x e para y de um ponto.
  p.x = 1.0 * rand() / RAND_MAX;
  p.y = 1.0 * rand() / RAND_MAX;
  //Escreve um registro do tipo TPonto no aqruivo.
  fwrite(&p, sizeof(TPonto), 1, arquivo);
 }

 //Fecha o arquivo.
 fclose(arquivo);
}
```

Esse programa, entretanto, não tem todas as suas funções implementadas. Sem alterar a função *main*, é sua tarefa programar:

1) **(2,5 pontos)** A função **void** transfereDeArquivoParaVetor (FILE * arquivo, TPonto v[ ], **int** tamanho). Essa função deve transferir cada ponto registrado no *arquivo* que foi criado pela função *pi_MonteCarlo* para o vetor *v*, correspondente ao segundo parâmetro dessa função. O *tamanho* desse vetor é determinado pelo último parâmetro da função.

2) **(2,5 pontos)** A função **float** distancia (TPonto p). Essa função deve retornar a distância do ponto *p* à origem. A expressão que calcula tal distância é $\sqrt{x^2 + y^2}$, sendo *x* e *y*, respectivamente, os valores das coordenadas do ponto *p*.

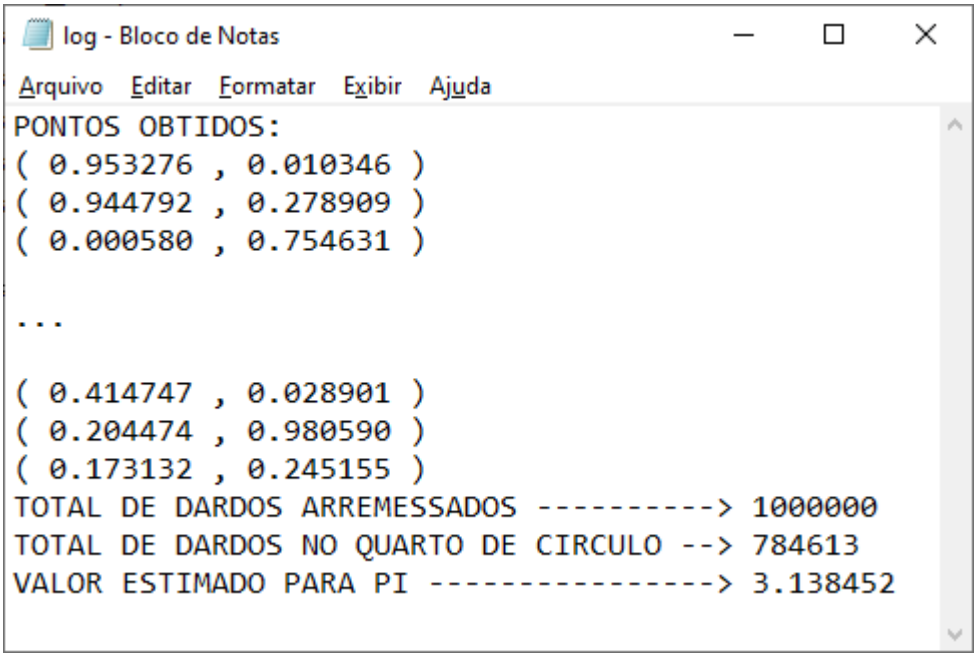
3) **(2,5 pontos)** A função **int** dardosNoQuartoDeCirculo (TPonto v[ ], **int** tamanho, **float** raio). Usando a função *distancia*, do item anterior, essa nova função deve computar a quantidade de pontos presentes no vetor *v*, cujo *tamanho* é definido no segundo parâmetro da função, que se encontram no quarto de círculo com *raio* igual ao valor do terceiro parâmetro da função.

4) **(2,5 pontos)** A função **void** criaLog (FILE * arquivo, TPonto v[ ], **int** pontosNoAlvo, **int** totalDePontos). Essa função deve escrever em um arquivo texto (o primeiro parâmetro da função), todos os pontos que estão armazenados no vetor v, segundo parâmetro da função. Essa função deve também escrever nesse mesmo arquivo:

- a quantidade de dardos que atingiram o quarto de círculo, valor armazenado no parâmetro *pontosNoAlvo*;
- o total de dardos que foram arremessados, valor armazenado no parâmetro *totalDePontos*;
- e uma estimativa para π.

Excepcionalmente, você pode programar a saída de dados nessa função auxiliar. O resultado dessa função será um arquivo de texto com conteúdo semelhante ao da imagem abaixo:

Saída (arquivo editado):



Boa avaliação.