Esquema recomendado

- Interface
- Implementação
- Cliente

Esquema recomendado: exemplo

- Interface: `Num3.2.h`
- Implementação: `Num3.2.c`
- Cliente: `prog3.2_client.c`

Esquema recomendado: exemplo

Arquivo `Num3.2.h`:

```
/* Num3.2.h */
typedef int numType;
numType randNum();
```

Arquivo `Num3.2.c`:

```
/* Num3.2.c */
#include <stdlib.h>
#include "Num3.2.h"

numType randNum()
  { return rand(); }
```

Esquema recomendado: exemplo

```
/* prog3.2_client.c */
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include "Num3.2.h"

main(int argc, char *argv[])
  { int i, N = atoi(argv[1]);
    float m1 = 0.0, m2 = 0.0;
    numType x;
```

Esquema recomendado: exemplo

```
/* prog3.2_client.c */
    (cont.)
    for (i = 0; i < N; i++)
      {
        x = randNum();
        m1 += ((float) x)/N;
        m2 += ((float) x*x)/N;
      }
    printf("       Average: %f\n", m1);
    printf("Std. deviation: %f\n", sqrt(m2-m1*m1));
 }
```

Esquema recomendado: exemplo

```
yoshi@erdos:~/Main/www/2006ii/mac122a/exx
$ gcc -g -I. -Wall -pedantic -ansi -c prog3.2_client.c Num3.2.c
yoshi@erdos:~/Main/www/2006ii/mac122a/exx
$ gcc -lm prog3.2_client.o Num3.2.o -o prog3.2_client
yoshi@erdos:~/Main/www/2006ii/mac122a/exx
$ prog3.2_client 1000
      Average: 1091191424.000000
Std. deviation: 609526184.005430
yoshi@erdos:~/Main/www/2006ii/mac122a/exx
$
```

Esquema recomendado: exemplo modificado

Arquivo `Num3.2b.h`:

```
/* Num3.2b.h */
typedef double numType;
numType randNum();
```

Esquema recomendado: exemplo modificado

Arquivo `Num3.2b.c`:

```
/* Num3.2b.c */
#include <stdlib.h>
#include "Num3.2b.h"

numType randNum()
{ return 2*((double)rand())/RAND_MAX - 1; }
```

Vejam `http://www.ime.usp.br/~pf/algoritmos/apend/stdlib.h.html`

Esquema recomendado: exemplo modificado

```
/* prog3.2_clientb.c */
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include "Num3.2b.h"

main(int argc, char *argv[])
  { int i, N = atoi(argv[1]);
    float m1 = 0.0, m2 = 0.0;
    numType x;
    (...)
    printf("       Average: %f\n", m1);
    printf("Std. deviation: %f\n", sqrt(m2-m1*m1));
 }
```

Esquema recomendado: exemplo modificado

```
yoshi@erdos:~/Main/www/2006ii/mac122a/exx
$ gcc -g -I. -Wall -pedantic -ansi -c prog3.2_clientb.c Num3.2b.c
yoshi@erdos:~/Main/www/2006ii/mac122a/exx
$ gcc -lm prog3.2_clientb.o Num3.2b.o -o prog3.2_clientb
yoshi@erdos:~/Main/www/2006ii/mac122a/exx
$ prog3.2_clientb 10000
       Average: -0.005736
Std. deviation: 0.576531
yoshi@erdos:~/Main/www/2006ii/mac122a/exx
$
```

Tipos compostos

- Vetores: coleção indexada de objetos do mesmo tipo
- Registro (`structs`): mais geral

Tipos compostos

Exemplo de `struct`:

```
struct point { float x; float y; };
struct point a, b;

a.x = 1.0; a.y = 1.0; b.x = 4.0; b.y = 5.0;
```

Como argumento de função:

```
float distance(struct point a, struct point b)
  { float dx = a.x - b.x, dy = a.y - b.y;
    return sqrt(dx*dx + dy*dy);
  }
```

Interface e implementação

```
/* Point.h */
typedef struct { float x; float y; } point;
float distance(point a, point b);

/* Point.c */
#include <math.h>
#include "Point.h"
float distance(point a, point b)
  { float dx = a.x - b.x, dy = a.y - b.y;
    return sqrt(dx*dx + dy*dy);
  }
```

Exemplo de uso

```
#include <stdio.h>
#include <stdlib.h>
#include "Point.h"

#define NN 1000

float randFloat()
  { return 1.0*rand()/RAND_MAX; }
```

Exemplo de uso

```
(cont.)
int main(int argc, char *argv[])
 { float d = atof(argv[2]);
   int i, j, cnt = 0, N = atoi(argv[1]);
   point a[NN];
   for (i = 0; i < N; i++)
     { a[i].x = randFloat(); a[i].y = randFloat(); }
   for (i = 0; i < N; i++)
     for (j = i+1; j < N; j++)
       if (distance(a[i], a[j]) < d) cnt++;
   printf("%d edges shorter than %f\n", cnt, d);
   return 0;
 }
```

Exemplo de uso

```
yoshi@erdos:~/Main/www/2006ii/mac122a/exx
$ gcc -g -I. -Wall -pedantic -ansi -c prog3.8a.c Point.c
yoshi@erdos:~/Main/www/2006ii/mac122a/exx
$ gcc -lm prog3.8a.o Point.o
yoshi@erdos:~/Main/www/2006ii/mac122a/exx
$ a.out 1000 .1
14549 edges shorter than 0.100000
yoshi@erdos:~/Main/www/2006ii/mac122a/exx
$
```

Mais sobre `structs`:

`http://www.ime.usp.br/~pf/algoritmos/aulas/stru.html`

Outra possibilidade

```
/* Point2.h */
typedef struct { float x; float y; } point;
float distance(point a, point b);
point randPoint();
```

Outra possibilidade

```
/* prog3.2a2.c */
#include <stdio.h>
#include <stdlib.h>
#include "Point2.h"

#define NN 1000
```

Outra possibilidade

```
/* prog3.2a2.c */
(cont.)
int main(int argc, char *argv[])
 { float d = atof(argv[2]);
   int i, j, cnt = 0, N = atoi(argv[1]);
   point a[NN];
   for (i = 0; i < N; i++)
     a[i] = randPoint();
   for (i = 0; i < N; i++)
     for (j = i+1; j < N; j++)
       if (distance(a[i], a[j]) < d) cnt++;
   return 0;
 }
```

Outra possibilidade

```
/* Point2.c */
#include <math.h>
#include <stdlib.h>
#include "Point2.h"
float distance(point a, point b)
  { [...] }
float randFloat()
  { return 1.0*rand()/RAND_MAX; }
point randPoint()
  { point p; p.x = randFloat(); p.y = randFloat();
    return p;
  }
```

Ponteiros

- Em geral, **difícil** no primeiro contato! (E por algum tempo mais, de acordo com alguns)

Você já pode imaginar minha recomendação:

`http://www.ime.usp.br/~pf/algoritmos/aulas/pont.html`

Ponteiros

```
int a;
int *b;  /* b é um ponteiro para inteiro */
[...]
b = &a;
printf("%d %d\n", *b, a);
a = 314;
printf("%d %d\n", *b, a);
*b = a + 7;
printf("%d %d\n", *b, a);
[...]
```

Em argumento de funções

```
polar(double x, double y, double *r, double *theta)
{ *r = sqrt(x*x + y*y);  *theta = atan2(y,x); }
```

Chamada:

```
double a, b;
[...]
polar(-1.0, 1.0, &a, &b);
```

Resulta em `a = 1.41421` ($\sqrt{2}$) e `b = 2.35619` ($3\pi/4$).

Mais um exemplo

```
void troca (int x, int y) /* errado! */
{
   int temp;
   temp = x; x = y; y = temp;
}
```

Correto:

```
void troca (int *x, int *y)
{
   int temp;
   temp = *x; *x = *y; *y = temp;
}
```

Mais um exemplo

Troca conteúdos de `x` e `y`:

```
troca (&x, &y);
```

Outro exemplo:

```
int *p, *q;
p = &a;
q = &b;
troca (p, q);
```

Exercícios

Exercício 1. *Determine experimentalmente a área esperada de um triângulo cujos vértices são escolhidos aleatoriamente de um quadrado* 1×1.

Exercício 2. *Determine experimentalmente a área esperada de um triângulo cujos vértices são escolhidos aleatoriamente de um disco de raio* 1.

Veja

`http://www.ime.usp.br/~yoshi/2006ii/mac122a/exx/Trigs1`

`http://www.ime.usp.br/~yoshi/2006ii/mac122a/exx/Trigs2`

Veja também

`http://mathworld.wolfram.com/SquareTrianglePicking.html`

`http://mathworld.wolfram.com/DiskTrianglePicking.html`

[**Observação:** se você fizer o Exercício 2 e vir a 4a. página acima, você notará certa inconsistência; qual? Talvez isso valha um bônus (você tem de resolver a inconsistência!).]