

INSTITUTO FEDERAL

Rio Grande do Sul

Campus Porto Alegre

Instituto Federal de Educação, Ciência e Tecnologia
do Rio Grande do Sul

Professor Fabio Okuyama

Email: fabio.okuyama@poa.ifrs.edu.br

Retomando...

onde foi que paramos?

Funções do nosso dia a dia

Acordar

Comer

Dormir

Escovar os dentes

Tomar banho


Trabalhar

Funções – o que é?

Conjunto de comandos que realiza uma tarefa específica.

- Bloco de código de programa;
- Pode ser utilizado várias vezes;
- Melhorar legibilidade do código;
- Estrutura do código;
- Possibilitar **REUSO**;
- Pode conter: Valores de Entrada e um valor de Saída (Retorno)

Funções - estrutura

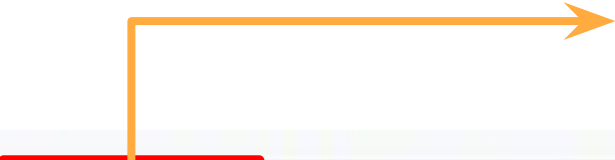


Tipo de retorno

```
1 int media (int a, int b){  
2     printf ("Calculando...");  
3     return((a+b)/2);  
4 }
```

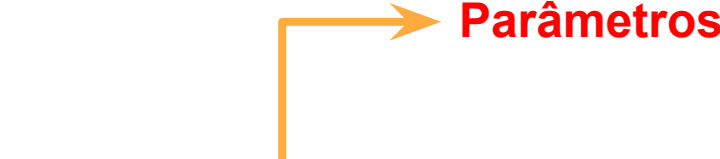
Funções - estrutura

**Nome da
função**



```
1 int media (int a, int b){  
2     printf ("Calculando...");  
3     return((a+b)/2);  
4 }
```

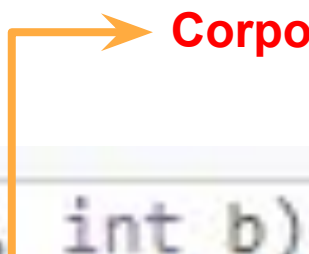
Funções - estrutura



Parâmetros

```
1 int media (int a, int b){  
2     printf ("Calculando...");  
3     return((a+b)/2);  
4 }
```

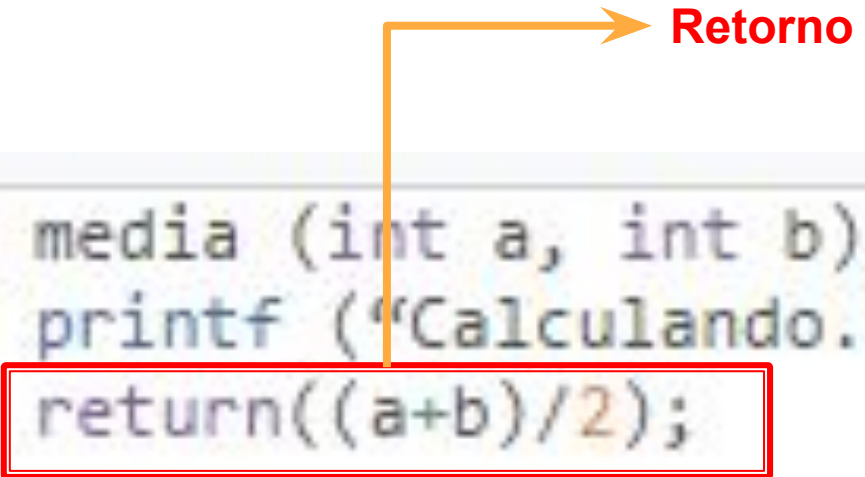
Funções - estrutura



Corpo

```
1 int media (int a, int b){  
2     printf ("Calculando...");  
3     return((a+b)/2);  
4 }
```


Funções - estrutura



```
1 int media (int a, int b){  
2     printf ("Calculando...");  
3     return((a+b)/2);  
4 }
```

Retorno

Nomes de funções

como nomear suas funções

Regras para nomes de funções

- **Mesma das variáveis**
 - Deve começar com letra ou _ (Underscore)
 - O nome de uma Função/variável não pode ser igual a uma palavra reservada (palavra-chave);
 - nem igual ao nome de outra função declarada pelo programador, ou pelas bibliotecas do C;
 - Limite de 32 caracteres

Tipos de retorno

o que sua função deve retornar?

Tipos de retorno de uma função

- As funções deverão ter um tipo de retorno
 - São válidos os mesmo tipos disponíveis para variáveis (INT, CHAR, FLOAT, ...)
 - Além disso é possível definir uma função como VOID (sem retorno).
 - A variável ou o valor a ser retornado é definido pelo comando RETURN:
 - `return (x);`
 - `return(0);`

Parâmetros de uma função

alimentando sua função

Parâmetros de uma função

- Passagem de parâmetros por valor
 - As variáveis são definidas em parênteses:
 - `int func(int a, int b);`
 - `char func(char a, int b);`
 - Os parâmetros podem ser acessados dentro da função
 - A mudança de valor de um parâmetro não altera o valor do original (é feita uma cópia)

Exemplos de funções

funções na prática

Exemplo de função

```
1 ▾ int func (char msg[], int a, int b){  
2     printf ("Ola %s!\n", msg);  
3     return a+b;  
4 }  
5  
6 ▾ void bomDia (){  
7     printf("Bom Dia\n");  
8 }  
9  
10 ▾ float pi(){  
11     return 3.141516;  
12 }  
13  
14 ▾ int main(){  
15     bomDia();  
16     func("Abobora",1,2);  
17     printf("Pi = %f",pi());  
18     printf("Func = %d", func("Abobora",1,2));  
19 }
```

Escopo de variáveis

além dos blocos

Escopo de variáveis

Globais

- Podem ser acessadas de qualquer ponto
- Declaradas no fora de funções
- Ou no arquivo de header (cabeçalho)

Locais

- Declarados em um subprograma
- Podem ser acessadas apenas dentro do subprograma

Encontre a(s) linha(s) que darão erro:

```
1  #include <stdio.h>
2  int a;
3  int mensagem (char msg[]){
4      int b;
5      printf ("Ola %s!", msg);
6      printf("%d", a);
7      printf("%d", b);
8      printf("%d", c);
9  }
10
11 int main(){
12     int c;
13     printf("%d", a);
14     printf("%d", b);
15     printf("%d", c);
16 }
```

Cuidado com variáveis Globais

- Podem ser acessada de qualquer ponto
- Risco de erros lógicos
 - Difíceis de corrigir

Exemplo com funções

```
1  #include <stdio.h>
2
3  int quadrado (int iArgumento){
4      return iArgumento * iArgumento;
5  }
6
7  int main (){
8      int iNum;
9      printf ("Entre com um numero: ");
10     scanf ("%d",&iNum);
11     printf("O quadrado eh %d",quadrado(iNum));
12 }
```

Assinatura de funções

a identidade de uma função

Assinaturas de funções

- Servem para informar ao compilador a existência de uma função que vai ser utilizada (chamar a função antes da implementação)
- Normalmente ficam no arquivo Header (.h) – Cabeçalho
- Os identificadores dos parâmetros podem ser omitidos

Exemplo de funções com assinatura

```
1  #include <stdio.h>
2
3  int soma (int, int);
4  int um();
5
6  int main(){
7      soma(1,2);
8      printf("Um = %d",um());
9  }
10
11 int soma (int a, int b){
12     return a+b;
13 }
14
15 int um(){
16     return 1;
17 }
```


Mais exemplos de funções

praticar é o melhor caminho

Mais exemplos...

```
1  #include <stdio.h>
2
3  void mult () {
4      float x,y;
5      printf("Entre com um número :");
6      scanf("%f",&x);
7      printf("Entre com outro número :");
8      scanf("%f",&y);
9      printf("%.4f",x*y);
10 }
11
12 int main(){
13     mult();
14     return(0);
15 }
```

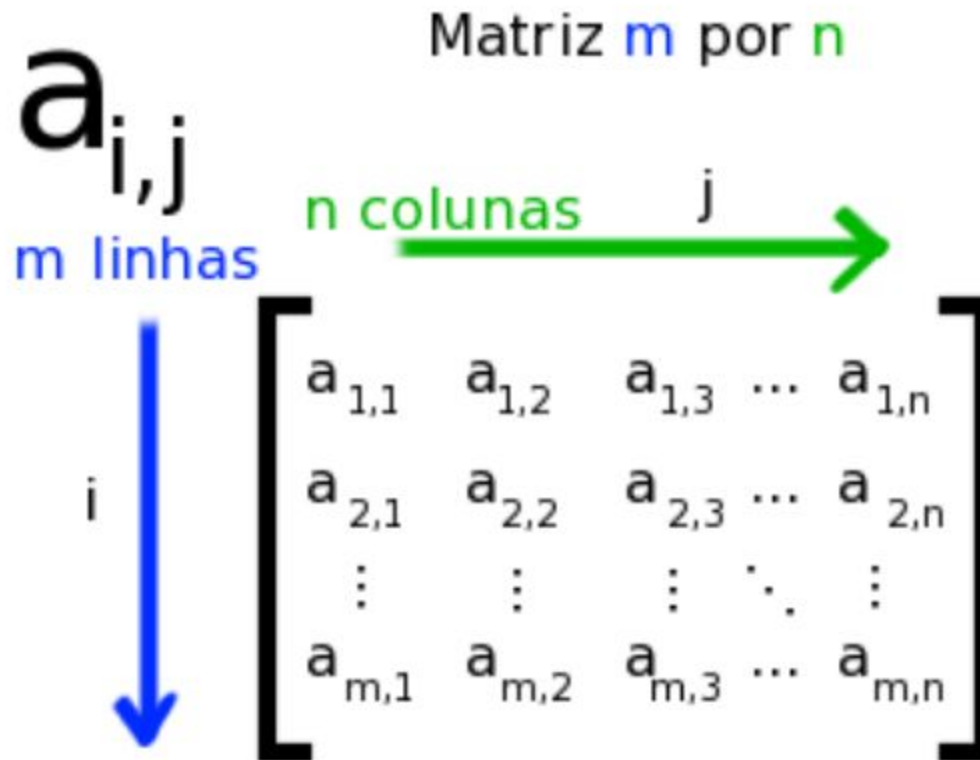
Mais exemplos...

```
1  #include <stdio.h>
2
3  int prod (int x,int y){
4      return (x*y);
5  }
6
7  int main (){
8      int iSaida;
9      iSaida = prod(12,3);
10     printf ("A saida eh: %d\n",iSaida);
11 }
```

Matrizes e Vetores

grandes aliados

O que é uma matriz?



- A matriz é um tipo de dado usado para representar uma certa quantidade de variáveis que são referenciados pelo mesmo nome.
- Consiste em locações contíguas de memória.

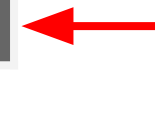
Matriz unidimensional: Vetores

índices



0 1 2 3 4 8 6 7 8

10	34	12	66	45	70	51	18	6
----	----	----	----	----	----	----	----	---



Conteúdo da Matriz

Para declarar uma matriz unidimensional, use a sintaxe:

Tipo nome [**tamanho**];

Declaração da matriz unidimensional do exemplo acima:

int idade[**9**];

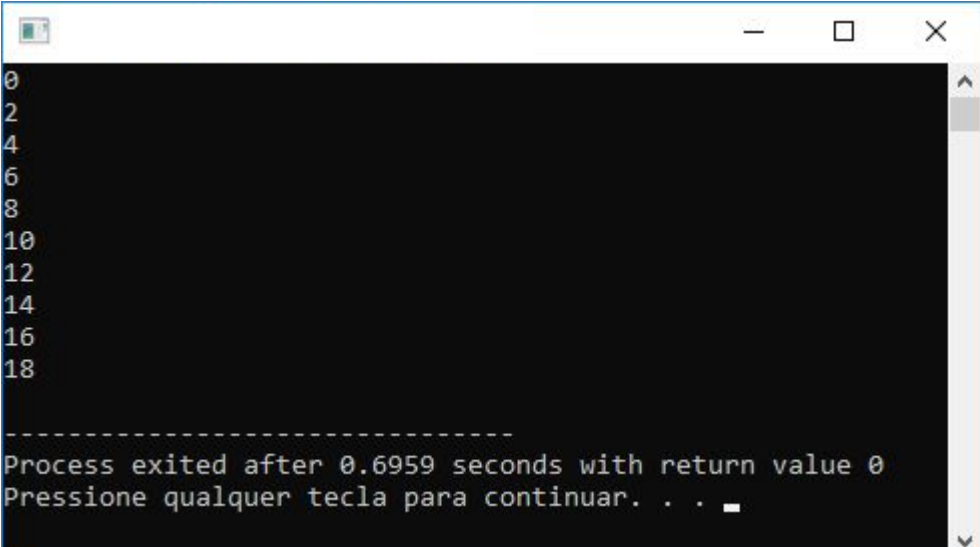
Exemplo 1 - Utilizando um Vetor

```
#include <stdio.h>

int main(void) {
    int meuVetor[10];
    int posicao;

    for(posicao = 0; posicao < 10; posicao++){
        meuVetor[posicao] = posicao*2;
        printf("%d\n", meuVetor[posicao]);
    }

    return 0;
}
```



```
0
2
4
6
8
10
12
14
16
18

-----
Process exited after 0.6959 seconds with return value 0
Pressione qualquer tecla para continuar. . .
```

Exemplo 2 - Cálculo da média da Turma

```
#include <stdio.h>

int main(void) {
    int notas[50], posicao, soma;

    // leitura dos alunos diretamente na posição do vetor
    for(posicao = 0; posicao < 50; posicao++){
        printf("\nDigite a Nota do aluno %d: ", posicao);
        scanf("%d", &notas[posicao]);
    }

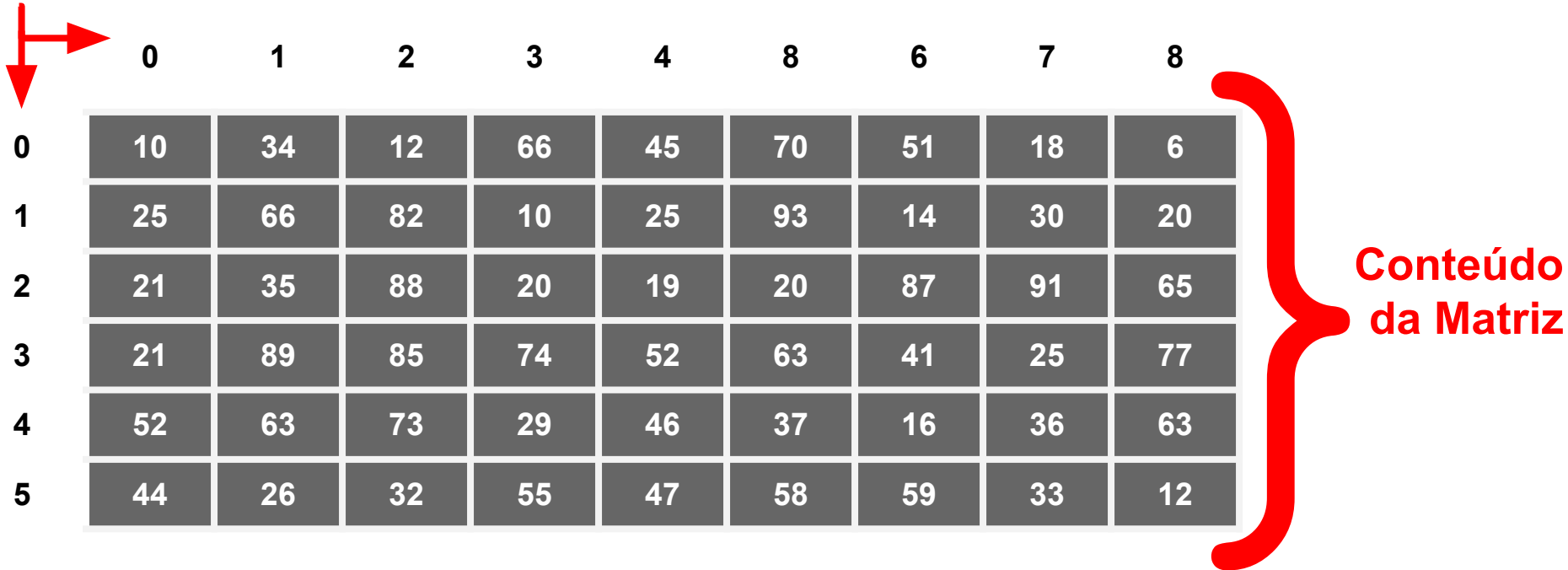
    soma = 0;
    // percorre o vetor somando o valor das posições
    for(posicao = 0; posicao < 50; posicao++){
        soma = soma + notas[posicao];
    }

    printf("\nMédia da Turma: %d", soma / 50);

    return 0;
}
```


Matriz multidimensional

índices



	0	1	2	3	4	8	6	7	8
0	10	34	12	66	45	70	51	18	6
1	25	66	82	10	25	93	14	30	20
2	21	35	88	20	19	20	87	91	65
3	21	89	85	74	52	63	41	25	77
4	52	63	73	29	46	37	16	36	63
5	44	26	32	55	47	58	59	33	12

Para declarar uma matriz unidimensional, use a sintaxe:

Tipo nome [**tamanho 1**][**tamanho 2**] .. [**tamanho**];

Declaração da matriz unidimensional do exemplo acima:

```
int idade[9][6];
```

Exemplo 3 - Utilizando um Matriz

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

```
int main(void) {
```

```
    int minhaMatriz[10][10];
```

```
    int posicao1, posicao2= 0;
```

```
    for(posicao1 = 0; posicao1 < 10; posicao1++, posicao2++){
```

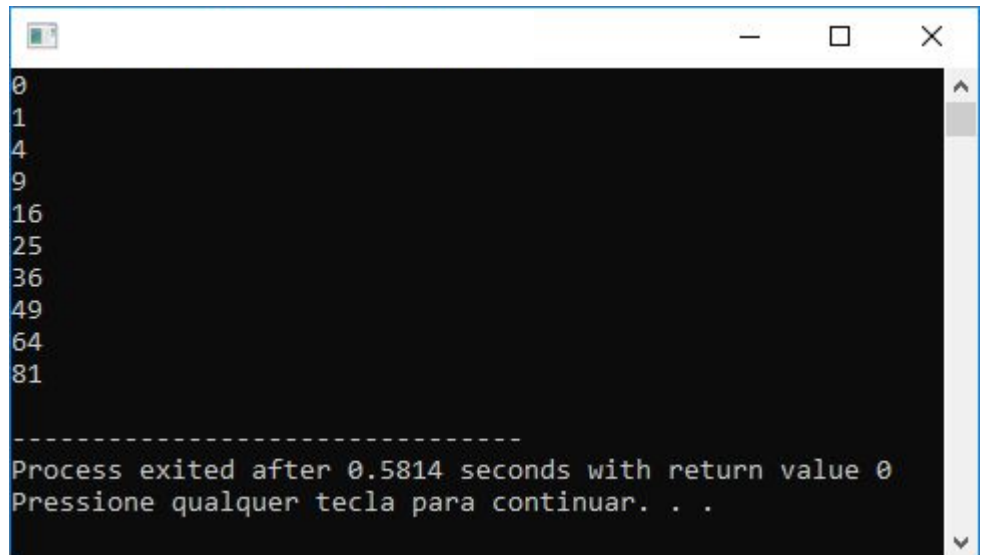
```
        minhaMatriz[posicao1][posicao2] = posicao1*posicao2;
```

```
        printf("%d\n", minhaMatriz[posicao1][posicao2]);
```

```
    }
```

```
    return 0;
```

```
}
```



```
0
1
4
9
16
25
36
49
64
81

-----
Process exited after 0.5814 seconds with return value 0
Pressione qualquer tecla para continuar. . .
```

Exemplo 4 - Cálculo da média dos Alunos

```
#include <stdio.h>

int main(void) {
    int notas[50][3], posicao1, posicao2, soma;
    // leitura dos alunos diretamente nas posições do vetor
    for(posicao1 = 0; posicao1 < 50; posicao1++){
        for(posicao2 = 0; posicao2 < 3; posicao2++){
            printf("\nDigite Nota %d aluno %d: ", posicao2, posicao1);
            scanf("%d", &notas[posicao1][posicao2]);
        }
    }
    for(posicao1 = 0; posicao1 < 50; posicao1++){
        soma = 0; // "zera" soma a cada aluno
        for(posicao2 = 0; posicao2 < 3; posicao2++){
            soma = soma + notas[posicao1][posicao2];
        }
        printf("\nMédia do aluno %d: %d", posicao1, soma / 3);
    }
    return 0;
}
```

ATENÇÃO

- Em C, os índices iniciam em **ZERO**.
- No índice, deve obrigatoriamente ser um inteiro
- Limites das matrizes não são verificados pelo compilador
- A responsabilidade no uso é do programador

```
#include <stdio.h>

int main(void) {
    int erro[10], posicao;
    for(posicao = 0; posicao < 100; posicao++){
        erro[posicao] = posicao;
        printf("%d\n", erro[posicao]);
    }

    return 0;
}
```

Atribuição de valores na declaração

```
//Cria e inicializa os valores
```

```
int x[10] = {5, 120, 28, 38, 489, 596, 68, 78, 86, 95};
```

```
//Cria vetor de 3 posições e inicializa
```

```
char a[] = {'a', 'b', 'c'};
```

Importante: esse tipo de atribuição é válido apenas na declaração da variável!

Matrizes Globais

```
#include <stdio.h>

int notas[3]; // vetor declarado globalmente

int media();

int main(void) {
    int posicao;
    // leitura dos alunos diretamente na posição do vetor
    for(posicao = 0; posicao < 3; posicao++){
        printf("\nDigite a Nota %d: ", posicao);
        scanf("%d", &notas[posicao]);
    }

    printf("\nMédia do aluno: %d", media());
    return 0;
}

int media() {
    int posicao, soma = 0;
    // percorre o vetor somando o valor das posições
    for(posicao = 0; posicao < 3; posicao++){
        soma = soma + notas[posicao];
    }
    return soma / 3;
}
```

Matrizes Locais

```
#include <stdio.h>

int media(int notas[3]);

int main(void) {
    int notas[3]; // vetor declarado localmente
    int posicao;
    // leitura dos alunos diretamente na posição do vetor
    for(posicao = 0; posicao < 3; posicao++){
        printf("\nDigite a Nota %d: ", posicao);
        scanf("%d", &notas[posicao]);
    }

    printf("\nMédia do aluno: %d", media(notas));
    return 0;
}

int media(int notas[3]) {
    int posicao, soma = 0;
    // percorre o vetor somando o valor das posições
    for(posicao = 0; posicao < 3; posicao++){
        soma = soma + notas[posicao];
    }
    return soma / 3;
}
```

Batalha Naval

Objetivo

explorar o uso de matrizes e vetores

Roteiro

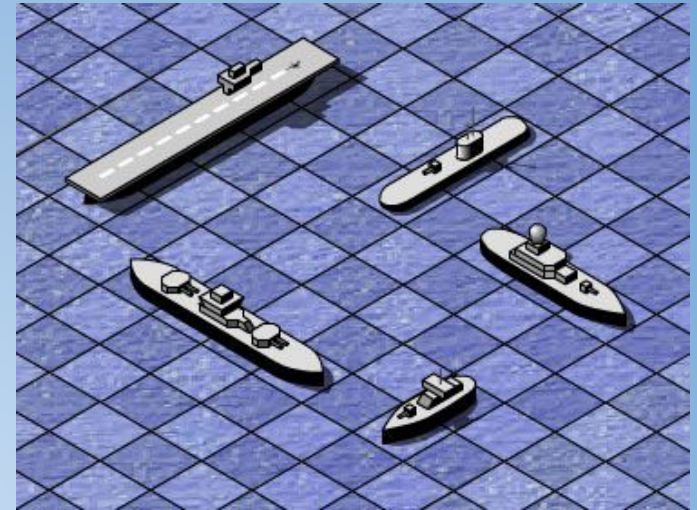
- conheça as regras do jogo
- desafie o seu colega

Tempo

10 min

Entrega

sem entrega

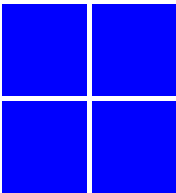


Batalha Naval

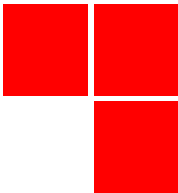
SEU JOGO

JOGO DO ADVERSÁRIO

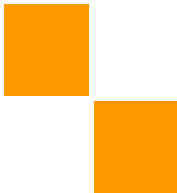
	A	B	C	D	E	F	G	H	I		A	B	C	D	E	F	G	H	I	
1										1										1
2										2										2
3										3										3
4										4										4
5										5										5
6										6										6
7										7										7
	A	B	C	D	E	F	G	H	I		A	B	C	D	E	F	G	H	I	



Submarino



Fragata



Couraçado



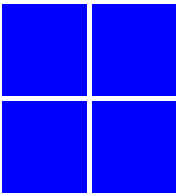
Crusador

Batalha Naval

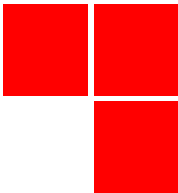
SEU JOGO

JOGO DO ADVERSÁRIO

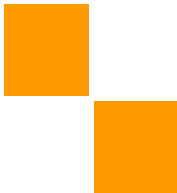
	A	B	C	D	E	F	G	H	I		A	B	C	D	E	F	G	H	I	
1										1		O								1
2										2										2
3										3										3
4										4										4
5										5										5
6										6										6
7										7										7
	A	B	C	D	E	F	G	H	I		A	B	C	D	E	F	G	H	I	



Submarino



Fragata



Couraçado



Crusador