

C++

Básico ao Avançado

Sequencial

Heitor Rodrigues Savegnago

UFABC Rocket Design

2017.3

1 Várias variáveis

2 Vetores

3 Matrizes

4 Strings

5 Preprocessador

6 Hora de brincar

Começando de outra forma

Começando de outra forma

- Vamos começar com $n = 5$

Começando de outra forma

- Vamos começar com $n = 5$
- Faça um programa que lê n valores

Começando de outra forma

- Vamos começar com $n = 5$
- Faça um programa que lê n valores
- Exiba os n valores na tela

Começando de outra forma

- Vamos começar com $n = 5$
- Faça um programa que lê n valores
- Exiba os n valores na tela
- Calcule a média dos n valores

Começando de outra forma

- Vamos começar com $n = 5$
- Faça um programa que lê n valores
- Exiba os n valores na tela
- Calcule a média dos n valores

```
int A1;  
int A2;  
int A3;  
int A4;  
int A5;
```


Começando de outra forma

- Vamos começar com $n = 5$
- Faça um programa que lê n valores
- Exiba os n valores na tela
- Calcule a média dos n valores

<code>int A1;</code>	<code>cin >> A1;</code>
<code>int A2;</code>	<code>cin >> A2;</code>
<code>int A3;</code>	<code>cin >> A3;</code>
<code>int A4;</code>	<code>cin >> A4;</code>
<code>int A5;</code>	<code>cin >> A5;</code>

Começando de outra forma

- Vamos começar com $n = 5$
- Faça um programa que lê n valores
- Exiba os n valores na tela
- Calcule a média dos n valores

```
int A1;    cin >> A1;  
int A2;    cin >> A2;  
int A3;    cin >> A3;  
int A4;    cin >> A4;  
int A5;    cin >> A5;
```

```
float media = 0.0f;  
;  
media += A1;  
media += A2;  
media += A3;  
media += A4;  
media += A5;  
media /= 5;
```

Começando de outra forma

- Vamos começar com $n = 5$
- Faça um programa que lê n valores
- Exiba os n valores na tela
- Calcule a média dos n valores

```
int A1;  
int A2;  
int A3;  
int A4;  
int A5;
```

```
cin >> A1;  
cin >> A2;  
cin >> A3;  
cin >> A4;  
cin >> A5;
```

```
float media = 0.0f;  
;  
media += A1;  
media += A2;  
media += A3;  
media += A4;  
media += A5;  
media /= 5;
```

```
cout << A1 << endl;  
cout << A2 << endl;  
cout << A3 << endl;  
cout << A4 << endl;  
cout << A5 << endl;
```

Começando de outra forma

- Vamos começar com $n = 5$
- Faça um programa que lê n valores
- Exiba os n valores na tela
- Calcule a média dos n valores

```
int A1;    cin >> A1;  
int A2;    cin >> A2;  
int A3;    cin >> A3;  
int A4;    cin >> A4;  
int A5;    cin >> A5;
```

```
float media = 0.0f;  
;  
media += A1;  
media += A2;  
media += A3;  
media += A4;  
media += A5;  
media /= 5;
```

```
cout << A1 << endl;  
cout << A2 << endl;  
cout << A3 << endl;  
cout << A4 << endl;  
cout << A5 << endl;
```

- Agora com $n = 6$,

Começando de outra forma

- Vamos começar com $n = 5$
- Faça um programa que lê n valores
- Exiba os n valores na tela
- Calcule a média dos n valores

```
int A1;    cin >> A1;  
int A2;    cin >> A2;  
int A3;    cin >> A3;  
int A4;    cin >> A4;  
int A5;    cin >> A5;
```

```
float media = 0.0f;  
;  
media += A1;  
media += A2;  
media += A3;  
media += A4;  
media += A5;  
media /= 5;
```

```
cout << A1 << endl;  
cout << A2 << endl;  
cout << A3 << endl;  
cout << A4 << endl;  
cout << A5 << endl;
```

- Agora com $n = 6$, $n = 7$,

Começando de outra forma

- Vamos começar com $n = 5$
- Faça um programa que lê n valores
- Exiba os n valores na tela
- Calcule a média dos n valores

```
int A1;
int A2;
int A3;
int A4;
int A5;

cin >> A1;
cin >> A2;
cin >> A3;
cin >> A4;
cin >> A5;
```

```
float media = 0.0f;

;
media += A1;
media += A2;
media += A3;
media += A4;
media += A5;
media /= 5;
```

```
cout << A1 << endl;
cout << A2 << endl;
cout << A3 << endl;
cout << A4 << endl;
cout << A5 << endl;
```

- Agora com $n = 6$, $n = 7$, $n = 10$,

Começando de outra forma

- Vamos começar com $n = 5$
- Faça um programa que lê n valores
- Exiba os n valores na tela
- Calcule a média dos n valores

```
int A1;    cin >> A1;
int A2;    cin >> A2;
int A3;    cin >> A3;
int A4;    cin >> A4;
int A5;    cin >> A5;
```

```
float media = 0.0f;
;
media += A1;
media += A2;
media += A3;
media += A4;
media += A5;
media /= 5;
```

```
cout << A1 << endl;
cout << A2 << endl;
cout << A3 << endl;
cout << A4 << endl;
cout << A5 << endl;
```

- Agora com $n = 6$, $n = 7$, $n = 10$, $n = 100...$

Começando de outra forma

- Vamos começar com $n = 5$
- Faça um programa que lê n valores
- Exiba os n valores na tela
- Calcule a média dos n valores

```
int A1;
int A2;
int A3;
int A4;
int A5;

cin >> A1;
cin >> A2;
cin >> A3;
cin >> A4;
cin >> A5;
```

```
float media = 0.0f;

;
media += A1;
media += A2;
media += A3;
media += A4;
media += A5;
media /= 5;
```

```
cout << A1 << endl;
cout << A2 << endl;
cout << A3 << endl;
cout << A4 << endl;
cout << A5 << endl;
```

- Agora com $n = 6$, $n = 7$, $n = 10$, $n = 100...$
- Existe uma forma de generalizar isso...

Motivacional

Motivacional

```
1  int  V00(0), V01(0), V02(0), V03(0), V04(0), V05(0), V06(0), V07(0),  
    V08(0), V09(0);  
2  int  V10(0), V11(0), V12(0), V13(0), V14(0), V15(0), V16(0), V17(0),  
    V18(0), V19(0);  
3  int  V20(0), V21(0), V22(0), V23(0), V24(0), V25(0), V26(0), V27(0),  
    V28(0), V29(0);  
4  int  V30(0), V31(0), V32(0), V33(0), V34(0), V35(0), V36(0), V37(0),  
    V38(0), V39(0);  
5  int  V40(0), V41(0), V42(0), V43(0), V44(0), V45(0), V46(0), V47(0),  
    V48(0), V49(0);  
6  int  V50(0), V51(0), V52(0), V53(0), V54(0), V55(0), V56(0), V57(0),  
    V58(0), V59(0);  
7  int  V60(0), V61(0), V62(0), V63(0), V64(0), V65(0), V66(0), V67(0),  
    V68(0), V69(0);  
8  int  V70(0), V71(0), V72(0), V73(0), V74(0), V75(0), V76(0), V77(0),  
    V78(0), V79(0);  
9  int  V80(0), V81(0), V82(0), V83(0), V84(0), V85(0), V86(0), V87(0),  
    V88(0), V89(0);  
10 int  V90(0), V91(0), V92(0), V93(0), V94(0), V95(0), V96(0), V97(0),  
    V98(0), V99(0);
```

Sequência padrão

Sequência padrão

- A forma que utilizamos é chamada *vetor*

Sequência padrão

- A forma que utilizamos é chamada *vetor*
- Conjuntos de números sequenciados em relação a uma base

Sequência padrão

- A forma que utilizamos é chamada *vetor*
- Conjuntos de números sequenciados em relação a uma base
- Para nós, variáveis sequenciadas

Sequência padrão

- A forma que utilizamos é chamada *vetor*
- Conjuntos de números sequenciados em relação a uma base
- Para nós, variáveis sequenciadas
- Acessíveis por um índice

Sequência padrão

- A forma que utilizamos é chamada *vetor*
- Conjuntos de números sequenciados em relação a uma base
- Para nós, variáveis sequenciadas
- Acessíveis por um índice
- Um operador novo!

Sequência padrão

- A forma que utilizamos é chamada *vetor*
- Conjuntos de números sequenciados em relação a uma base
- Para nós, variáveis sequenciadas
- Acessíveis por um índice
- Um operador novo! `[]`

Sequência padrão

- A forma que utilizamos é chamada *vetor*
- Conjuntos de números sequenciados em relação a uma base
- Para nós, variáveis sequenciadas
- Acessíveis por um índice
- Um operador novo! []

```
<tipo> <nome>[<tamanho>];
```

Sequência padrão

- A forma que utilizamos é chamada *vetor*
- Conjuntos de números sequenciados em relação a uma base
- Para nós, variáveis sequenciadas
- Acessíveis por um índice
- Um operador novo! []

```
<tipo> <nome>[<tamanho>];
```

- Inicialização diferente

Sequência padrão

- A forma que utilizamos é chamada *vetor*
- Conjuntos de números sequenciados em relação a uma base
- Para nós, variáveis sequenciadas
- Acessíveis por um índice
- Um operador novo! `[]`

```
<tipo> <nome>[<tamanho>];
```

- Inicialização diferente

```
<tipo> <nome>[<tam>]{<val1>, ..., <valn>}; //Sintaxe de
    Inicialização
//...
int V1[5]{}; //Todos os valores como 0
int V2[5]{1,2,3,4,5}; //Valores conforme sequência
int V3[] {1,2,3,4,5}; //Tamanho do vetor implícito
```

Sequência padrão

Sequência padrão

- Acessar os valores por índices entre colchetes [i]

Sequência padrão

- Acessar os valores por índices entre colchetes [i]
- Começa em 0

Sequência padrão

- Acessar os valores por índices entre colchetes `[i]`
- Começa em 0
- Acaba em $n - 1$

Sequência padrão

- Acessar os valores por índices entre colchetes $[i]$
- Começa em 0
- Acaba em $n - 1$

Tabela: Representação de vetor de n valores

0	1	2	3	...	$n - 1$

Sequência padrão

- Acessar os valores por índices entre colchetes `[i]`
- Começa em 0
- Acaba em $n - 1$

Tabela: Representação de vetor de n valores

0	1	2	3	...	$n - 1$

```
//...  
float valores[10];  
//...  
float media(0.0f);  
for(int i=0; i<10; i++) media +=valores[i];  
media /= 10;  
//...
```

Sequência de sequências

Sequência de sequências

- Semelhante ao vetor

Sequência de sequências

- Semelhante ao vetor
- Pode ser considerada uma *matriz*

Sequência de sequências

- Semelhante ao vetor
- Pode ser considerada uma *matriz*
- Consistem em um vetor com vetores

Sequência de sequências

- Semelhante ao vetor
- Pode ser considerada uma *matriz*
- Consistem em um vetor com vetores
- Sintaxe de declaração semelhante à do vetor

Sequência de sequências

- Semelhante ao vetor
- Pode ser considerada uma *matriz*
- Consistem em um vetor com vetores
- Sintaxe de declaração semelhante à do vetor

```
<tipo> <nome>[<tamanho1>][<tamanho2>;
```


Sequência de sequências

- Semelhante ao vetor
- Pode ser considerada uma *matriz*
- Consistem em um vetor com vetores
- Sintaxe de declaração semelhante à do vetor

```
<tipo> <nome>[<tamanho1>][<tamanho2>;
```

- Inicialização muito diferente

Sequência de sequências

- Semelhante ao vetor
- Pode ser considerada uma *matriz*
- Consistem em um vetor com vetores
- Sintaxe de declaração semelhante à do vetor

```
<tipo> <nome>[<tamanho1>][<tamanho2>;
```

- Inicialização muito diferente

```
<tipo> <nome>[<tam1>][<tam2>]{<val11>, ..., <val1n>},  
    ..., {<valm1>, ..., <valmn>}}; //Sintaxe de  
    Inicialização  
//...  
int V1[2][3]{}; //Todos os valores como 0  
int V2[2][2]{{1,2,3},{4,5,6}}; //Valores separador por  
    vetor  
int V3[2][2]{1,2,3,4,5,6}; //Valores distribuídos  
    automaticamente  
int V3[][2]{1,2,3,4,5,6}; //Distribuição com um  
    tamanho implícito
```

Sequência de sequências

Sequência de sequências

- Um colchete para cada índice

Sequência de sequências

- Um colchete para cada índice
- Também começa em 0

Sequência de sequências

- Um colchete para cada índice
- Também começa em 0
- O passo de vetor para matriz é o passo de matriz para tensor

Sequência de sequências

- Um colchete para cada índice
- Também começa em 0
- O passo de vetor para matriz é o passo de matriz para tensor
- O `for` foi feito pra isso

Tabela: Representação de matriz de $m \times n$

	0	1	2	3	...	$n - 1$
0						
1						
2						
3						
\vdots						
$m - 1$						

Sequência de sequências

Sequência de sequências

```
//...  
float valores[8][12];  
//...  
float media[8]{};  
for(int i=0; i<8; i++) for(int j=0; j<12; j++) media[i] +=  
    valores[i][j];  
for(int i=0; i<8; i++) media[i] /=8;  
//...
```

Sequências de caracteres

Sequências de caracteres

- Vetor especial de `char`

Sequências de caracteres

- Vetor especial de `char`
- A forma de inicialização diferente na master

Sequências de caracteres

- Vetor especial de `char`
- A forma de inicialização diferente na master

```
char fraseA [] ( "Hello World!" );  
char fraseB [] = "Hello World!" ;
```

Sequências de caracteres

- Vetor especial de `char`
- A forma de inicialização diferente na master

```
char fraseA [] ( "Hello World!" );  
char fraseB [] = "Hello World!" ;
```

- Tamanho implícito?

Sequências de caracteres

- Vetor especial de `char`
- A forma de inicialização diferente na master

```
char fraseA [] ( "Hello World!" );  
char fraseB [] = "Hello World!" ;
```

- Tamanho implícito?
- Aspas duplas

Sequências de caracteres

- Vetor especial de `char`
- A forma de inicialização diferente na master

```
char fraseA [] ( "Hello World!" );  
char fraseB [] = "Hello World!" ;
```

- Tamanho implícito?
- Aspas duplas e o caractere nulo (`'\0'`)

Sequências de caracteres

- Vetor especial de `char`
- A forma de inicialização diferente na master

```
char fraseA [] ( "Hello World!" );  
char fraseB [] = "Hello World";
```

- Tamanho implícito?
- Aspas duplas e o caractere nulo (`'\0'`)
- Podemos contar tamanho de *string* com isso?

Sequências de caracteres

Sequências de caracteres

```
//...
char A[]("string");
char B[]{'s','t','r','i','n','g','\0'}; //Equivale a A
char C[]{'s','t','r','i','n','g'};      //Não equivale a A
char D[]{'H','e','l','l','o','\0',' ','W','o','r','l','d','!'};
};
//Espaço em branco não é nulo
//...
int size(0);
for(int i=0; A[i]!='\0'; i++) size++;    //Contagem tamanho da
    string
size = 0;
for(int i=0; D[i]!='\0'; i++) size++;    //Note que a contagem
    falha
//...
```

Preprocessador

Preprocessador

- Tamanhos precisam ser literais

Preprocessador

- Tamanhos precisam ser literais
- E se eu pudesse criar literais com nomes

Preprocessador

- Tamanhos precisam ser literais
- E se eu pudesse criar literais com nomes
- Agora você pode

Preprocessador

- Tamanhos precisam ser literais
- E se eu pudesse criar literais com nomes
- Agora você pode
- *Se ligar agora, 75% de desconto*

Preprocessador

- Tamanhos precisam ser literais
- E se eu pudesse criar literais com nomes
- Agora você pode
- *Se ligar agora, 75% de desconto*
- Define pra mim o que você quer...

Preprocessador

- Tamanhos precisam ser literais
- E se eu pudesse criar literais com nomes
- Agora você pode
- *Se ligar agora, 75% de desconto*
- Define pra mim o que você quer... `#define`

Preprocessador

- Tamanhos precisam ser literais
- E se eu pudesse criar literais com nomes
- Agora você pode
- *Se ligar agora, 75% de desconto*
- Define pra mim o que você quer... `#define`

```
#define SIZE 3  
//...  
int V1[SIZE][SIZE]{};  
//...
```

Vamos testar!