BCC702 - Programação de Computadores II

Apresentação da disciplina

Prof José Romildo Malaquias
Profª Valéria de Carvalho Santos

UFOP

decom
departamento de computação

BCC702 - Programação de Computadores II

Pré-requisito: BCC701

- ❑ Introdução a ambientes de programação.
- ❑ Conceitos de algoritmo.
- ❑ Conceitos básicos de programação: valores e expressões de tipos primitivos, variáveis, comando de atribuição, comandos de controle de fluxo, entrada e saída padrão, procedimentos e funções, tipos de dados compostos.

BCC702 - Programação de Computadores II

Pré-requisito: BCC701

- ❑ Introdução a ambientes de programação.
- ❑ Conceitos de algoritmo.
- ❑ Conceitos básicos de programação: valores e expressões de tipos primitivos, variáveis, comando de atribuição, comandos de controle de fluxo, entrada e saída padrão, procedimentos e funções, tipos de dados compostos.

Ementa (BCC702):

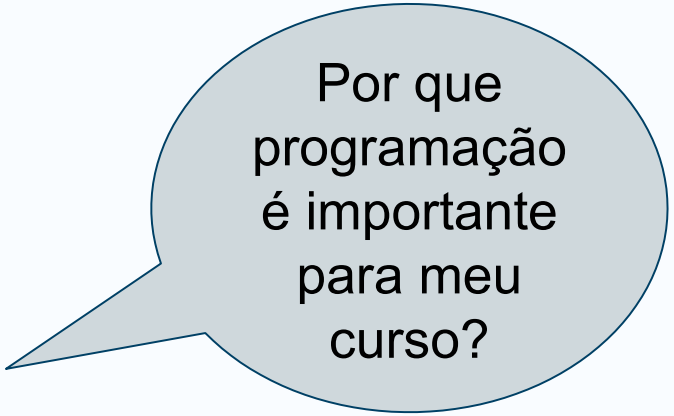
- ❑ Processamento de arquivos.
- ❑ Modularização de programas e abstração de dados.
- ❑ Conceitualização e utilização de estruturas de dados.
- ❑ Algoritmos de pesquisa e ordenação.
- ❑ Desenvolvimento de programas com utilização de uma biblioteca de algoritmos e estruturas de dados.

BCC702 - Programação de Computadores II

- Ver plano de curso no site da disciplina em

http://www.decom.ufop.br/romildo/2020-1/bcc702/

BCC702 - Programação de Computadores II



BCC702 - Programação de Computadores II

Por que programação é importante para meu curso:

- ❑ Graduação: formação base
- ❑ Aplicações
  - ❑ Problemas de otimização
  - ❑ Automatização de processos
  - ❑ Programação de sistemas embutidos e robóticos

BCC702 - Programação de Computadores II

Por que programação é importante para meu curso:

- Cada vez mais Engenharia (e várias áreas) precisa de programação
  - https://www.bbc.com/portuguese/geral-42145774
  - https://www.ctrlplay.com.br/conheca-5-beneficios-da-programacao-para-criancas/
  - https://noticias.portaldaindustria.com.br/listas/8-razoes-para-ensinar-programacao-a-criancas-e-adolescentes/
  - https://www.youtube.com/watch?v=MXw8YXusoPg&list=PL5vSn8ej1b0vJmPJI6DAcoigpF8TB_y1b&index=2

BCC702 - Programação de Computadores II

Metodologia:

- ❑ Aulas teóricas expositivas
- ❑ Aulas práticas em laboratório
- ❑ Atividades extraclasse para resolução de exercícios
  - ❑ Participação na tutoria

BCC702 - Programação de Computadores II

Avaliações:

- ❑ Prova Teórica I (08/04) - Valor: 10,0 pts - Peso 0,33
- ❑ Prova Teórica II (20/05) - Valor: 10,0 pts - Peso 0,33
- ❑ Prova Teórica III (01/07) - Valor: 10,0 pts - Peso 0,33

BCC702 - Programação de Computadores II

Outras informações:

- ❑ Aulas presenciais: 75% de presença
  - ❑ Abono de faltas: PROGRAD - ver CEPE nº 2.880

- ❑ Início da aula: 15 min de tolerância

BCC702 - Programação de Computadores II

Dúvidas:

Email: malaquias@ufop.edu.br

http://www.decom.ufop.br/decom/pessoal/planos_trabalho_publico/

Horários de atendimento:

- Terça: 13h30 às 17h30
- Quarta: 15h30 às 17h30
- Quinta: 13h30 às 17h30

BCC702 - Programação de Computadores II

Introdução à programação em C++

Profª Valéria de Carvalho Santos

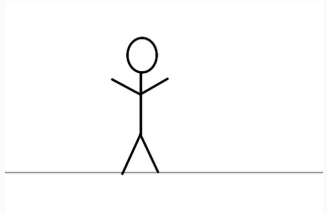
Linguagem de Programação

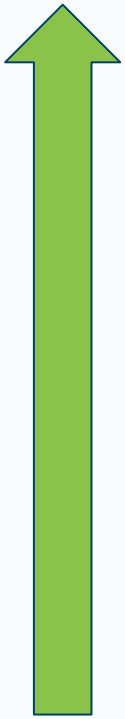
- Método padronizado para comunicar instruções a um computador

- Possui um conjunto de regras (léxicas, sintáticas e semânticas) para representar um programa de computador.

Linguagem de Programação

Linguagem de Programação





Alto nível

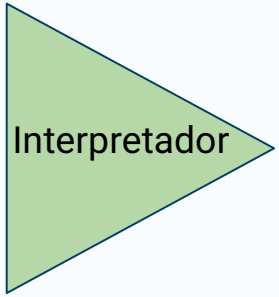
Baixo nível

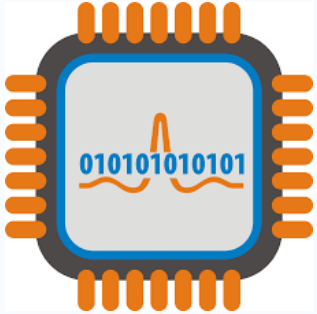
Linguagem de Programação

Linguagem interpretada: código fonte é executado pelo interpretador e em seguida é executado pelo processador ou sistema operacional.

Scilab

```
printf("Soma de dois inteiros\n");
n1 = input("Digite o primeiro número: ");
n2 = input("Digite o segundo número: ");
soma = n1 + n2;
printf("\nA soma de %g + %g é igual a %g", n1, n2, soma);
```



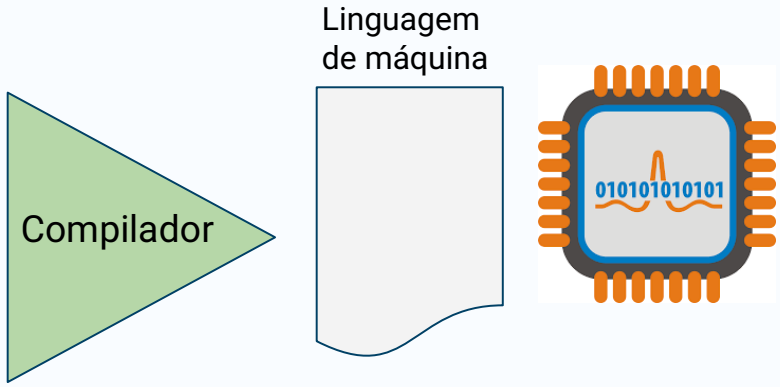


Linguagem de Programação

Linguagem compilada: código fonte é convertido para linguagem de máquina antes de ser executado pelo processador ou sistema operacional.

C++

```
#include <iostream>

using namespace std;
int main(){
        int soma, n1, n2;
        cout << "Soma de dois inteiros << endl;
        cout << "Digite o primeiro número: " << endl;
        cin >> n1;
        cout << "Digite o segundo número: " << endl;
        cin >> n2;
        soma = n1 + n2;
        cout << "A soma de" << n1 << "+" << n2 << "é igual a" << soma;
        return 0;
}
```



Linguagem de Programação

Scilab

```
1.      printf("Soma de dois inteiros\n");
2.      n1 = input("Digite o primeiro número: ");
3.      n2 = input("Digite o segundo número: ");
4.      soma = n1 + n2;
5.      printf("\nA soma de %g + %g é igual a %g", n1, n2,
soma);
```

C++

```
1       #include <iostream>

2       using namespace std;
3       int main(){
4               int soma, n1, n2;
5               cout << "Soma de dois inteiros << endl;
6               cout << "Digite o primeiro número: " << endl;
7               cin >> n1;
8               cout << "Digite o segundo número: " << endl;
9               cin >> n2;
10              soma = n1 + n2;
11              cout << "A soma de" << n1 << "+" << n2 << "é igual a"
                << soma;
                return 0;
12      }
```

Linguagem C++

- Desenvolvida por Bjarne Stroustrup na década de 80
- Extensão da linguagem C para suportar Programação Orientada a Objetos
- Suportar os dois paradigmas: procedural e orientado a objetos
- Linguagem compilada
- Tipagem estática: os tipos são verificados pelo compilador

Linguagem C++

Algoritmos

- **Algoritmo** corresponde a uma descrição de um padrão de comportamento, expresso em termos de um conjunto finito de ações.

- Informalmente, um algoritmo é qualquer procedimento computacional bem definido que toma algum valor ou conjunto de valores como entrada e os transforma em saída(s).

Algoritmos

- **Programas** são formulações concretas de algoritmos abstratos, baseados em representações e estruturas específicas de dados.

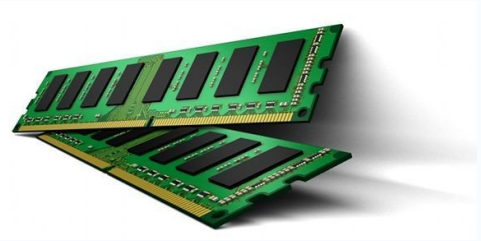
- Todo programa pode ser escrito como uma combinação de comandos primitivos envolvendo três estruturas básicas de controle:
  - Estrutura sequencial;
  - Estrutura de seleção;
  - Estrutura de repetição.

Programa em C++

```
#include <iostream>

using namespace std;

//programa principal
int main(){
        cout << “Olá mundo” << endl;
        return 0;
}
```

Comandos	Significado
#include <iostream>	Inclui biblioteca com funções de entrada e saída de dados
using namespace std;	Em C++ as bibliotecas são divididas em namespaces. Usando o namespace std (*standard*) da biblioteca *iostream*
//programa principal	Comentário. Essa linha não é executada pelo compilador.
int main()	Função principal. Início da execução.
cout << “Olá mundo!” << endl;	Função que escreve na tela.
return 0;	Retorno da função main.
;	Indica o fim de uma instrução.
{	Indica o início de um bloco de instruções.
}	Indica o fim de um bloco de instruções.

Variáveis



- Locais de armazenamento da informação gerada
  - Exemplo: notas, soma, média, idade, etc
- Os valores variam de acordo com o contexto e ocupam um espaço em memória
- Definição formal:
  - Objeto ou entidade situada na memória que representa um valor ou uma expressão. Esta representação existe apenas em tempo de execução.
- As variáveis são referenciadas por um nome ou identificador.

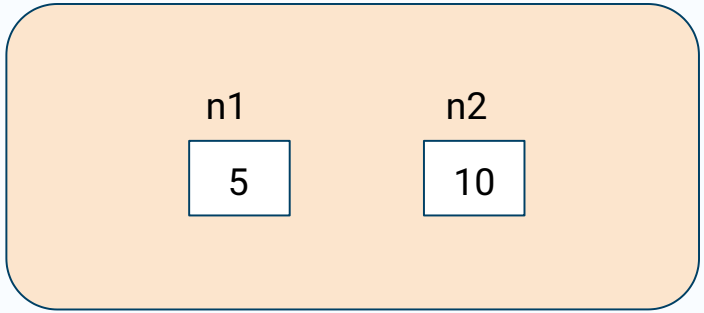
Variáveis

- Um identificador deve iniciar por uma letra ou por um "_" (*underline*);
- A partir do segundo caractere, pode conter letras (ç e acentos não são válidos), números e *underline*;
- Deve-se usar nomes significativos dentro do contexto do programa;
- C é uma linguagem *case-sensitive*, ou seja, faz diferença entre nomes com letras maiúsculas e nomes com letras minúsculas. *Idade* e *idade* são diferentes;
- Exemplos:
  - Idade, contador, taxaMatricula, aluno_1, valorMaximo

Variáveis

- Declaração de variável: reserva um espaço em memória;
- Atribuição de valor: altera o conteúdo da variável.

```
int main(){
	int n1, n2;

	n1 = 5;
	n2 = 2*n1;
	return 0;
}
```



Tipos de dados

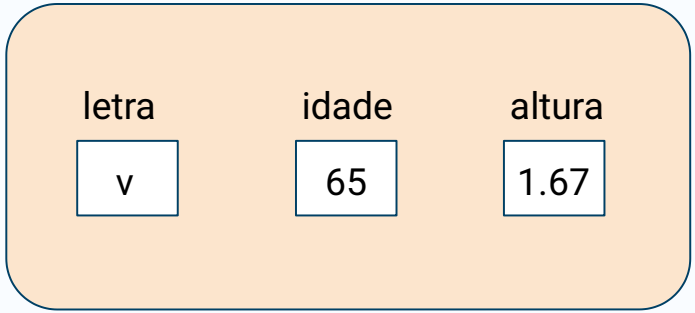
- C++ é uma linguagem tipada estaticamente: os tipos de todas as variáveis são fixados quando são declaradas em tempo de compilação.
- Cada variável tem apenas um tipo de dado associado quando é declarada.
- Tipos: número inteiro, texto, caractere, número real, etc
- Cada tipo define os valores que a variável pode armazenar e ocupa um tamanho de espaço em memória.

Tipos de dados

Tipo	Valores
número inteiro	short, int, long
número real	float, double
caractere	char
booleano	bool

Tipos de dados

```
int main(){
        char letra;
        int idade;
        float altura;

        letra = ‘v’;
        idade = 65;
        altura = 1.67;
        return 0;
}
```

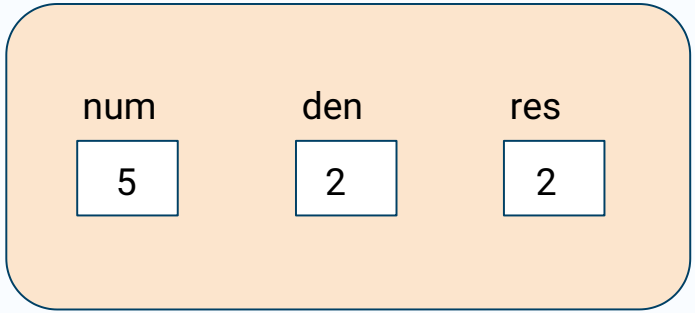


Operadores aritméticos

Operação	Exemplo
soma (+)	x = y + 5
subtração (-)	diferenca = g - 2;
multiplicação (*)	total = diferenca*3;
divisão (/)	quociente = x/3;
resto da divisão (%)	resto = total%2;

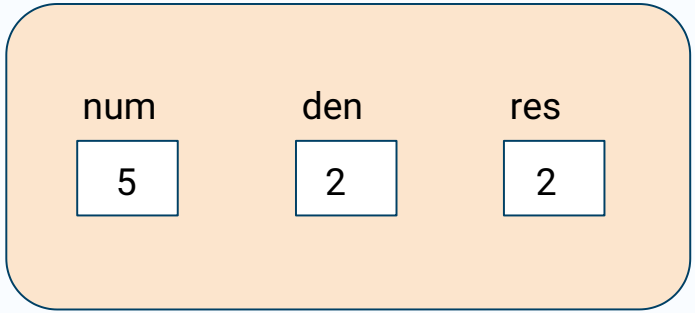
Operadores aritméticos

```
int main(){
        int num=5, den=2;
        int res = num/den;
        return 0;
}
```



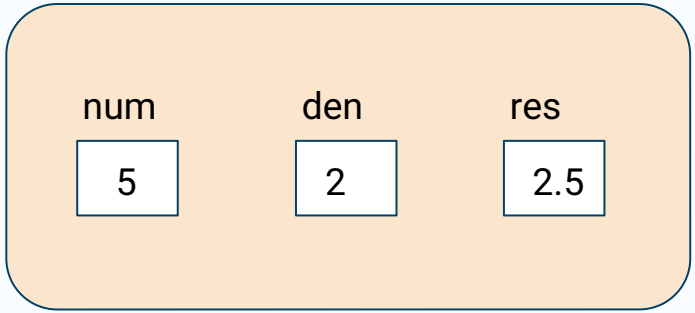
Operadores aritméticos

```
int main(){
        int num=5, den=2;
        float res = num/den;
        return 0;
}
```



Operadores aritméticos

```
int main(){
        int num=5, den=2;
        float res = ((float)num)/den;
        return 0;
}
```



Saída de dados

- Biblioteca iostream
- Variável **cout**
  - representa o fluxo (stream) de saída padrão (tela)
- Operador **<<**
  - Envia um dado para um fluxo de saída
  - Operador binário infixo
  - Operando da esquerda: fluxo de saída que vai receber o dado
  - Operando da direita: dado a ser inserido no fluxo de saída
  - O dado pode ser uma constante, variável, texto, etc.
  - O resultado é o próprio fluxo de saída
- Sintaxe:
  - cout << *valor*;
  - cout << *valor* << endl;

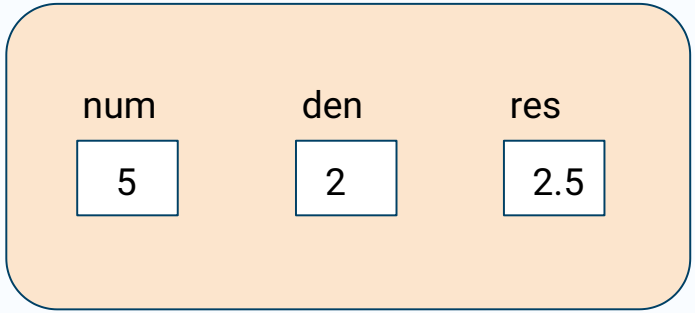
Saída de dados

```
#include <iostream>

using namespace std;

int main(){
	int num=5;
	cout << 120 << endl;
	cout << num << endl;
	cout << “Oi” << endl;
	return 0;
}
```

```
120
5
Oi
```

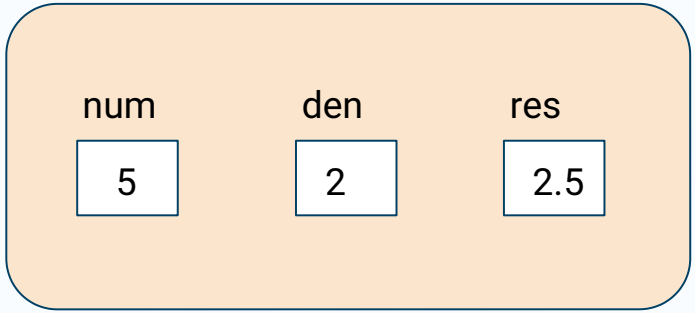
Saída de dados

```
#include <iostream>

using namespace std;

int main(){
	int num=5, den=2;
	float res = ((float)num)/den;
	cout << res << endl;
	return 0;
}
```



Saída de dados

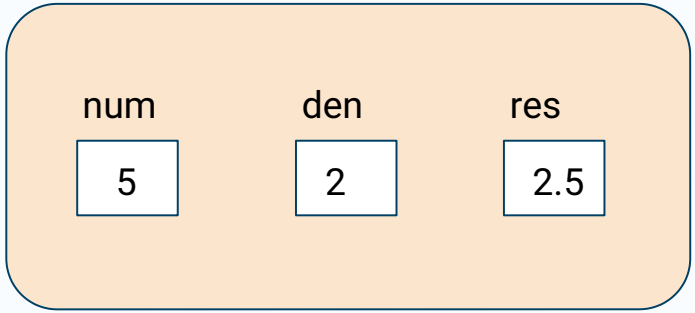
```
#include <iostream>

using namespace std;

int main(){
	int num=5, den=2;
	float res = ((float)num)/den;
	cout << “Resultado = ” << res << endl;
	return 0;
}
```




```
Resultado = 2.5
```

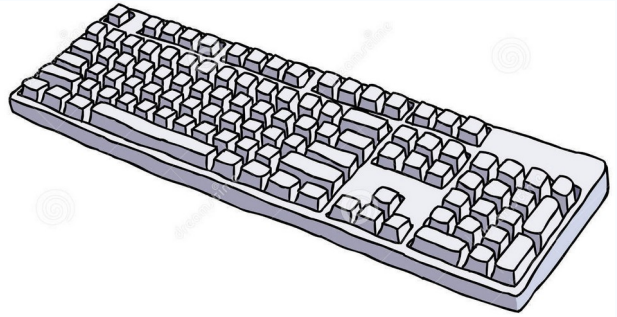
Saída de dados

```
#include <iostream>

using namespace std;

int main(){
	int num=5, den=2;
	float res = ((float)num)/den;
	cout << Resultado =  << res << endl;
	return 0;
}
```



‘Resultado’ was not declared in this scope

Entrada de dados



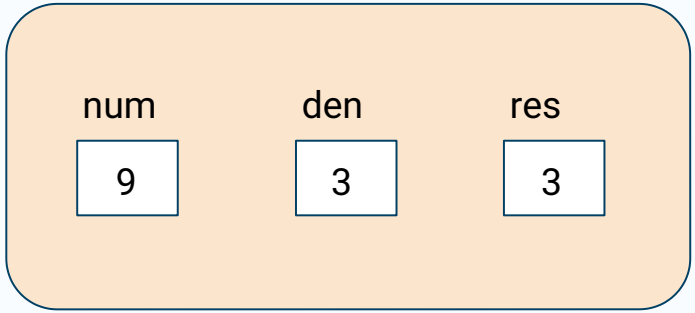
- Biblioteca iostream

- Variável cin
  - Representa o fluxo (stream) de entrada padrão (teclado)

- Operador >>
  - Extrai um dado de um fluxo de entrada
  - Operador binário infixo
  - Operando da esquerda: fluxo de entrada de onde o dado será extra
  - Sintaxe:
    - cin >> *variável*;

Entrada de dados

```
#include <iostream>

using namespace std;

int main(){
        int num, den;
        cout << “Digite o numerador: ”;
        cin >> num;
        cout << “Digite o denominador: ”;
        cin >> den;
        res = num/den;
        cout << “Resultado =”  << res << endl;
        return 0;
}
```




```
Digite o numerador: 9
Digite o denominador: 3
Resultado = 3
```

Exemplo completo

Escreva um programa em C++ que recebe como entrada a quantidade de dias e os converta em semanas. A conversão deve considerar como respostas apenas semanas completas.

Exemplo:
Digite a quantidade de dias: 22

Saída: 22 dias são 3 semanas

Exemplo completo

```
#include <iostream>

using namespace std;

int main(){
	int dias, semanas;
	cout << “Digite a quantidade de dias: ”;
	cin >> dias;
	semanas = dias/7;
	cout << dias << “ são ”  << semanas << “semanas” <<  endl;
	return 0;
}
```