

Apostila: Algoritmos e Lógica de Programação

Linguagem de Programação C/C++

Índice

Sumário

1. Cronograma Aula de Laboratório	3
2. Orientações sobre essa apostila	4
3. Instruções para Elaboração/Entrega dos roteiros	4
4. Distribuição de Pontos	4
5. Referência bibliográfica	5
Roteiro 1 - Como iniciar uma aplicação de console no Dev-C++	7
Roteiro 2 – Estrutura Sequencial em C/C++	11
Exemplo	14
Exercícios	14
Roteiro 3 – Estrutura Condicional em C/C++	16
Exemplos	18
Exercícios	19
Roteiro 4 – Estrutura de Repetição em C/C++	21
Exemplos	22
Exercícios	23
Roteiro 5 – Vetor em C/C++	25
Exemplos	26
Exercícios	27
Roteiro 6 – Matriz em C/C++	29
Exemplos	30
Exercícios	31
Lista 1 – Estrutura Sequencial em C/C++	33
Lista 2 – Estrutura Condicional em C/C++	34
Lista 3 – Estrutura de Repetição em C/C++	36
Lista 4 – Vetor em C/C++	38
Lista 5 – Matriz em C/C++	40

1. Cronograma Aula de Laboratório

Semana	Aula	Roteiro	Unidade de Ensino /Roteiro	Referência Bibliográfica
14 a 16/fev.		-	Recepção aos calouros	
18 a 23/fev.	1	-	Conceitos iniciais de Algoritmos	
25/fev. a 02/mar.	2	Roteiro 1	Como iniciar uma aplicação no Dev-C++	Apostila - Roteiro 1
04 a 09/mar.	3	Roteiro 2	Estrutura Sequencial	Apostila - Roteiro 2, cap. 3 [Ascencio], cap. 2 [Mizrahi, 1990]
11 a 16/mar.	4	Roteiro 2	Estrutura Sequencial	Apostila - Roteiro 2, cap. 3 [Ascencio], cap. 2 [Mizrahi, 1990]
18 a 23/mar.	5	Roteiro 3	Estrutura Condicional	Apostila - Roteiro 3, cap. 4 [Ascencio], cap. 3 [Mizrahi, 1990]
25 a 30/mar.		-	27 a 30/03 - Semana Santa e Recesso Aula de Exercício (lista 1 e 2)	Apostila páginas: 33 e 34
01 a 06/abr.	6	Roteiro 3	Estrutura Condicional	Apostila - Roteiro 3, cap. 4 [Ascencio], cap. 3 [Mizrahi, 1990]
08 a 13/abr.	7	Roteiro 4	Estrutura de Repetição	Apostila - Roteiro 4, cap. 5 [Ascencio], cap. 4 [Mizrahi, 1990]
15 a 20/abr.	8	Roteiro 4	Estrutura de Repetição	Apostila - Roteiro 4, cap. 5 [Ascencio], cap. 4 [Mizrahi, 1990]
22 a 27/abr.	9	Roteiro 4	Estrutura de Repetição	Apostila - Roteiro 4, cap. 5 [Ascencio], cap. 4 [Mizrahi, 1990]
29/abr a 04/mai.		-	01/05 – Dia do trabalho (feriado) Aula de Exercício (lista 3)	Apostila páginas: 36
06 a 11/mai.	10	Roteiro 5	Vetor	Apostila - Roteiro 5, cap. 6 [Ascencio], cap. 6 [Mizrahi, 1990]
13 a 18/mai.	11	Roteiro 5	Vetor	Apostila - Roteiro 5, cap. 6 [Ascencio], cap. 6 [Mizrahi, 1990]
20 a 25/mai.	12	Roteiro 6	Matriz	Apostila - Roteiro 6, cap. 7 [Ascencio], cap. 6 [Mizrahi, 1990]
27/mai. a 01/jun.		-	30 e 31/05 – Feriado e Recesso Aula de Exercício (lista 4 e 5)	Apostila páginas: 38 e 40
03 a 08/jun.	13	Roteiro 6	Matriz	Apostila - Roteiro 6, cap. 7 [Ascencio], cap. 6 [Mizrahi, 1990]
10 a 15/jun.	14	Roteiro 6	Matriz	Apostila - Roteiro 6, cap. 7 [Ascencio], cap. 6 [Mizrahi, 1990]
17 a 22/jun.	15	-	Avaliação Prática	

2. Orientações sobre essa apostila

A apostila possui roteiros que deverão ser desenvolvidos em laboratório e caso seja necessário terminar em casa. Os roteiros foram desenvolvidos com o intuito de não apenas permitir a prática da programação de computadores, como também apresentar um material de apoio a disciplina teórica. Dessa forma, cada roteiro tem uma descrição da matéria que será praticada (estrutura da linguagem C/C++), bem como, exemplos práticos da linguagem.

Sendo assim, espera-se que todo aluno inicie o roteiro a partir da leitura do material e, em seguida, tente compilar e entender os exemplos. Dessa forma, muitas dúvidas serão sanadas. Mesmo assim, caso tenha necessidade solicite a ajuda do professor da disciplina. Além dos roteiros práticos, a apostila apresenta listas de exercícios que serão trabalhadas em sala, mas a maior parte delas deverá ser desenvolvida em casa. As listas de exercícios são um complemento de estudo e deverão ser entregues ao final da disciplina para o professor.

3. Instruções para Elaboração/Entrega dos roteiros

- O desenvolvimento dos roteiros deverá seguir o cronograma apresentado anteriormente. Caso o aluno não consiga terminar o roteiro em sala de aula, deverá levar para casa e fazer os exercícios faltantes.
- O aluno deverá entregar seu roteiro individualmente, podendo realizar um trabalho em equipe, mas cada aluno deve ser capaz de fazer e apresentar o seu trabalho. Os alunos ausentes nas datas previstas para entregas dos roteiros serão premiados com nota ZERO no roteiro em questão. Cópias dos roteiros práticos e listas de exercícios serão penalizadas com a nota ZERO para ambos os alunos (o que fez e o que copiou).
- Os roteiros práticos devem ser apresentados para o professor da disciplina prática SEMPRE na **próxima** aula de laboratório. O professor poderá sortear o aluno que irá apresentar o trabalho.
- Cada aluno tem como obrigação realizar o backup de seus roteiros que foram desenvolvidos no laboratório, bem como em casa (através de pen drive ou via email) para que possa apresentar na aula indicada.
- Para desenvolver os roteiros o aluno precisará instalar o “DEV C++” em seu computador. Faça o download do programa no endereço: http://prdownloads.sourceforge.net/dev-cpp/devcpp-4.9.9.2_setup.exe.

4. Distribuição de Pontos

Roteiro	Pontos
Roteiro 2 – Estrutura seqüencial	1,0
Roteiro 3 – Estrutura condicional	2,0
Roteiro 4 – Estrutura de Repetição	2,0
Roteiro 5 – Vetor	2,0
Roteiro 6 – Matriz	3,0
Avaliação Prática	10,0
Total	20 pontos

5. Referência bibliográfica

Referência Básica

[ASCENCIO] Ascencio, Ana Fernanda Gomes e Campos, Edilene Aparecida Veneruchi de. . Fundamentos da programação de computadores: algoritmos, Pascal e C/C++ e Java. 2.ed. São Paulo: Pearson Pascal e C/C++ e Java. 2.ed. São Paulo: Pearson Prentice Hall, 2008.

[CORMEN] Cormen, Thomas H. [et al.]. Algoritmos: teoria e prática. Rio de Janeiro: Elsevier, 2002.

[DEITEL] Deitel, H.M; Deitel, P.J. Como Programar C++. Porto Alegre:. Bookmen. 2006

Referência Complementar

[FORBELLONE] Forbellone, André Luiz Villar e Eberspächer, Henri Frederico. Lógica de programação : a construção de algoritmos e estruturas de dados. Editora Pearson, Prentice Hall. 3ª Ed.

[MEDINA] Medina, Marco. Algoritmos e programação : teoria e prática. . 2ª Ed. São Paulo. Novatec. 2006 . 2a. Ed. São Paulo. Novatec. 2006

[MIZRAHI, 1990] Mizrahi, Victorine Viviane. Treinamento em Linguagem C++: Módulo I.São Paulo. Makron Books do Brasil Editora Ltda. 1990

[SILVA] Silva, Osmar Quirino da. . Estrutura de dados e algoritmos usando c: fundamentos e aplicações. Rio de Janeiro: Ciência Moderna, 2007.

[SOUZA] Souza, Marco Antonio Furlan de, [et al.]. Algoritmos e lógica de programação. São Paulo : Thomson Learning, 2005.

Curso de C da Engenharia Elétrica da UFMG. Disponível em:
<http://www.ead.cpdee.ufmg.br/cursos/C/c.html>

Roteiros Aulas Práticas (Laboratório)

Roteiro 1 - Como iniciar uma aplicação de console no Dev-C++

Uma aplicação do tipo console é uma aplicação que roda no Prompt de comando. Ela não utiliza **interface gráfica** do Windows. Este documento tem o objetivo de ensinar passo-a-passo como iniciar o desenvolvimento de uma aplicação de linha de comando usando o Dev-C++.

1. Inicie o Dev-C++ clicando no ícone Dev-C++ do Menu Iniciar. Ao iniciar o programa, ele abrirá uma tela parecida com a exibida na **Figura 1**. Feche a tela com a Dica do dia clicando no X.

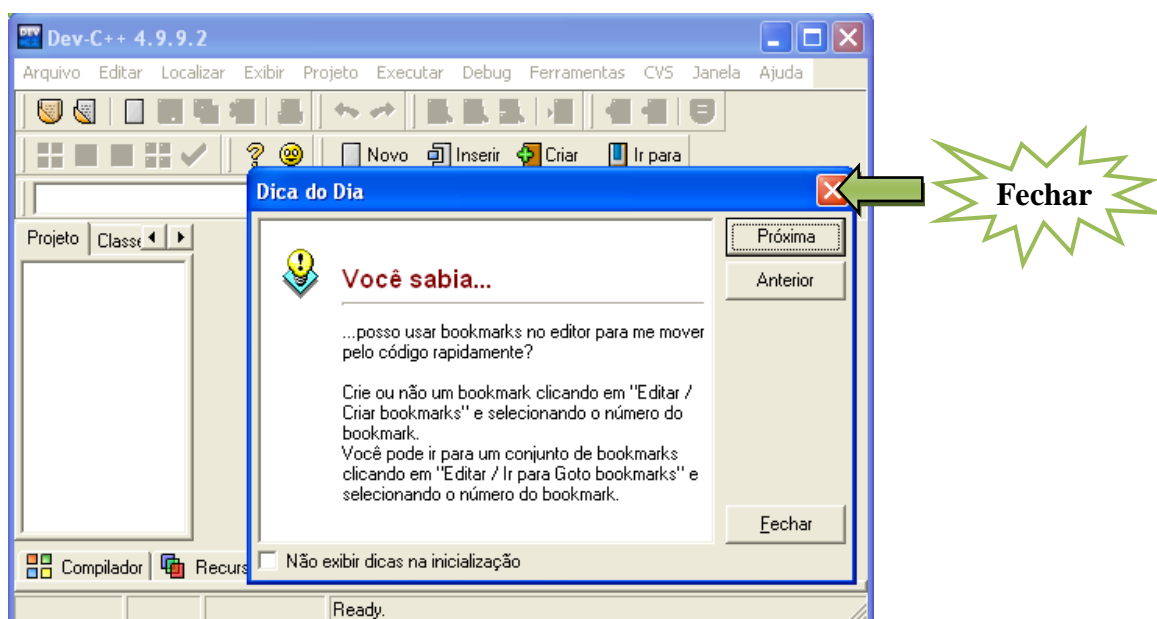


Figura 1 - Tela Inicial do Dev-C++

2. Crie um novo arquivo fonte pressionando o botão **Arquivo – Novo – Arquivo Fonte**. Em seguida aparecerá a tela mostrada na **Figura 2**.

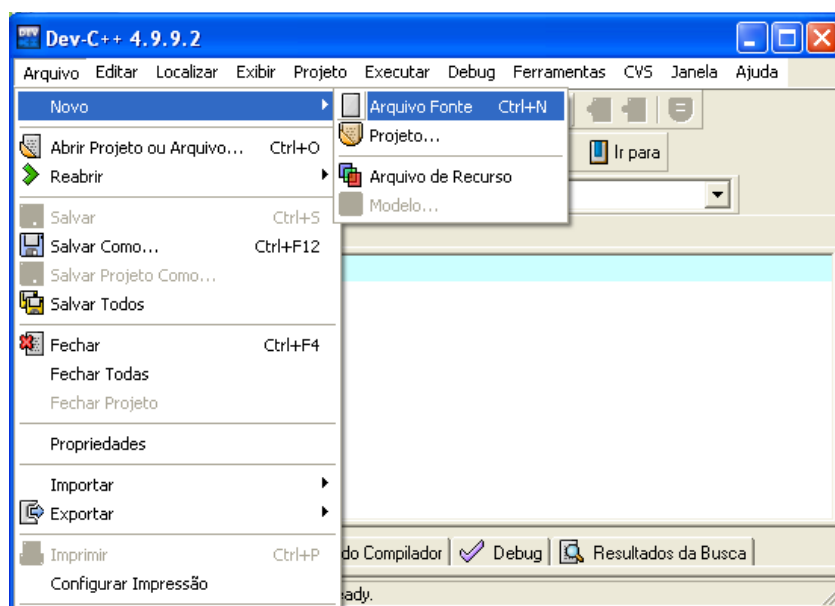


Figura 2 - Tela Arquivo fonte

- Em seguida será aberta uma tela no qual você deverá digitar o programa em C/C++ (**figura 3**).

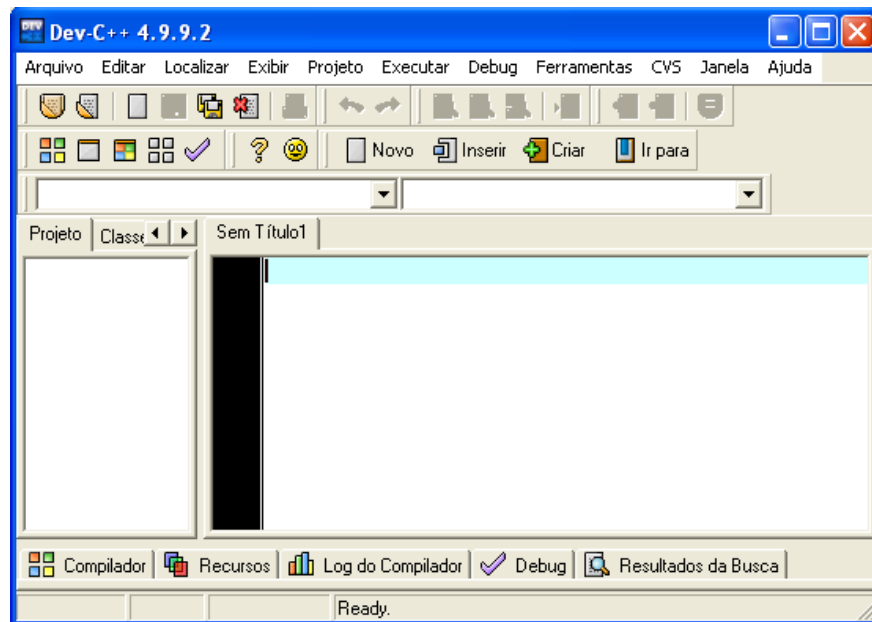


Figura 3 – Tela inserção do código


- Não se esqueça de colocar as bibliotecas necessárias para os comandos e funções que for usar. A seguir é apresentado nosso primeiro exemplo, digite o código abaixo.

Exemplo 1:

```
/*
Programador : <coloque o seu Nome>
Data : <coloque a data>
Descrição: Imprime na tela a mensagem de boas vindas.
*/

#include <iostream.h>

int main()
{
    cout <<"Bem vindo a UNA!!!";
    cout<<"\n";
    cout <<"Aula pratica da disciplina: Algoritmos e Logica de Programacao";
    cout<< "\n\n";
    system("PAUSE");
}
```

- Após digitar o código acima no Dev-C++, compile e execute o programa. Para compilar e executar o programa, basta clicar no botão  ou (F9).

Observe que no exemplo 1, foram usados:

- **Comentários de programa**, que são informações a respeito do programa. Para isso, foram usados `/*` (**para indicar o início do comentário**) e `*/` (**para indicar o fim do comentário**). É possível também usar comentários com apenas uma linha através de `//`.
 - **Bibliotecas** que servem para fazer com que o compilador permita a utilização de funções de outros arquivos. Para incluir uma biblioteca é necessário usar `#include <nome da biblioteca>`. Em geral, esse recurso é usado para incluir definições de dados e código que serão utilizados por nosso programa, mas já foram compilados e estão disponíveis em uma biblioteca. No exemplo, usamos apenas a biblioteca `iostream.h`.
 - **Função principal `main()`**, já que todos os comandos da linguagem C/C++ devem estar dentro de funções, o que implica, portanto, que um programa deve ter no mínimo uma função (a principal).
 - Todas as declarações e comandos da linguagem devem ser terminados por `;` (ponto e vírgula). Esse sinal serve apenas como separador nas declarações, mas serve para identificar a composição de sequência entre os comandos, isto é, primeiro é executado um e depois o outro.
 - `cout<<` é o comando utilizado para imprimir mensagens na Tela.
6. Uma tela será aberta para salvar o arquivo (dê um nome para o arquivo principal do seu programa em seguida salve). Use sempre nomes sugestivos para os seus programas, ou seja, nomes que lhe remetam a idéia ou objetivo do programa. Para o nosso exemplo vamos dar o seguinte nome **primeiro_programa**. Não use acentos e caracteres especiais nos nomes dos programas
7. O seu programa executará em seguida mostrará uma mensagem na tela (**figura 4**)

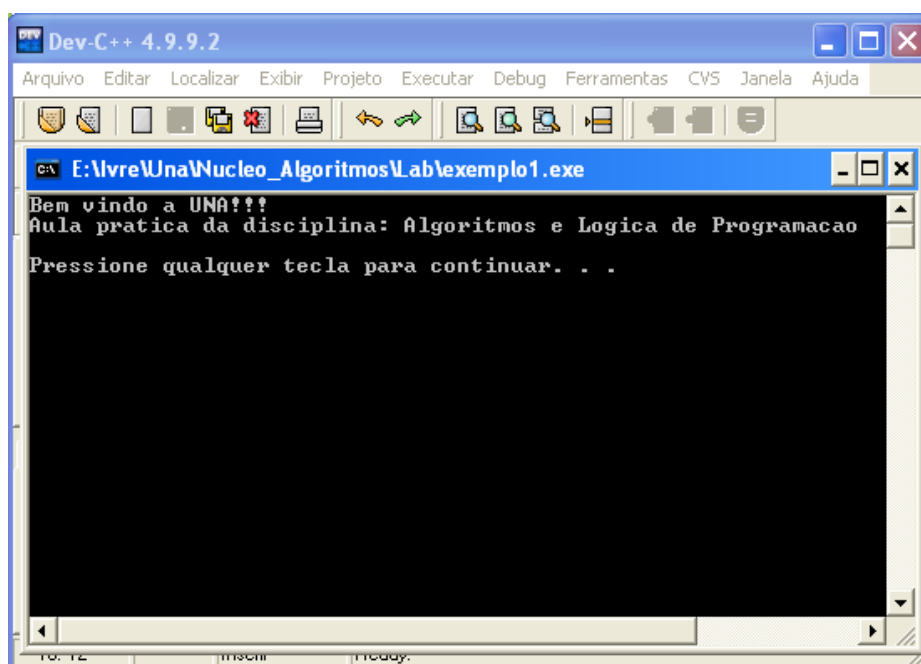


Figura 4 – Tela de resultado de compilação

8. Agora experimente retirar as linhas do programa `system("PAUSE");` e `return 0;` e em seguida, compile novamente (F9). O que acontece?

Como exercício, digite os exemplos abaixo no Dev-C++ e tente entender o que está ocorrendo:

Exemplo 2:

```
/*
Programador : <Nome>
Data :
Descrição:
*/

#include <iostream.h>

int main()
{
    int num1;
    cout << "\n Digite um numero: ";
    cin >> num1;
    cout << "\n Vai imprimir o numero digitado: ";
    cout << num1;
    system("PAUSE");
}
```

Exemplo 3:

```
/*
Programador : <Nome>
Data :
Descrição:
*/

#include <iostream.h>

int main()
{
    char mensagem[20];
    cout << "\n Digite alguma coisa: ";
    gets(mensagem);
    cout << "\n Vai imprimir o texto digitado: "<<mensagem;
    system("PAUSE");
}
```

Roteiro 2 – Estrutura Sequencial em C/C++

A estrutura sequencial na linguagem C/C++ consiste em:

```
# include <nome da biblioteca>
int main()
{
    Bloco de comandos;
}
```

- ✓ Bibliotecas são arquivos contendo várias funções que podem ser incorporadas aos programas escritos em C/C++. A diretiva **# include** faz com que o texto contido dentro da biblioteca especificada seja inserido no programa.
- ✓ As bibliotecas **iostream.h** e **conio.h** permitem a utilização de diversos comandos de entrada e saída.
- ✓ A linguagem C/C++ é sensível a letras maiúsculas e minúsculas, ou seja, considera que letras maiúsculas são **diferentes** de minúsculas (por exemplo, 'a' é diferente de 'A')
- ✓ Todos os **comandos** devem, obrigatoriamente, ser escritos com letras minúsculas.

Declaração de variáveis:

```
int main()
{
    //Declaração de variáveis
    int Y;
    float X;
    char sexo, nome[40];
}
```

- ✓ As variáveis são declaradas após a especificação de seus **tipos** (int, float ou char).
- ✓ C/C++ não possui tipo especial para armazenar cadeias de caracteres (strings).

Comando de atribuição:

```
int main()
{
    int y, x, soma;

    //Atribuição de valores
    y = 2;
    x = 3;
    soma = y + x;
}
```

- ✓ Utilizado para atribuir valores ou operações a variáveis, sendo representado por = (sinal de igualdade).

Comando de Entrada e Saída:

```
int main()
{
    //Declaração de variáveis
    int num, x, soma;

    cout<<"Digite um numero: "; //Comando de Saída
    cin>>num; //Comando de Entrada

    //Atribuição de valores
    x = 3;
    soma = num + x;

    //Comando de Saída
    cout<<"A soma dos numeros e: "<<soma;
}
```

- ✓ Para entrada de dados em C/C++, o comando que será utilizado é o **cin>>**. Assim, com o comando (`cin>>num;`) o valor digitado pelo usuário será armazenado na variável num.
- ✓ Para saída de dados em C/C++, o comando que será utilizado é o **cout<<**. Assim, com o comando (`cout<<"Digite um numero: ";`) o texto Digite um número aparecerá na tela do computador para o usuário.

Operadores e funções predefinidas:

A linguagem C/C++ possui alguns operadores e funções predefinidas destinadas a cálculos matemáticos e à manipulação de caracteres.

Operadores matemáticos:

Operador	Exemplo	Comentário
+	<code>x + y</code>	Soma o conteúdo de X e de Y.
-	<code>x - y</code>	Subtrai o conteúdo de Y do conteúdo de X
*	<code>x * y</code>	Multiplica o conteúdo de X pelo conteúdo de Y
/	<code>x / y</code>	Obtém o quociente da divisão de X por Y
%	<code>x % y</code>	Obtém o resto da divisão de X por Y
++	<code>x ++</code>	Aumenta o conteúdo de X em uma unidade (é o mesmo que <code>x = x + 1</code>)
--	<code>x --</code>	Diminui o conteúdo de X em uma unidade (é o mesmo que <code>x = x - 1</code>)

Operadores matemáticos de atribuição:

Operador	Exemplo	Comentário
<code>+=</code>	<code>x += y</code>	Equivale a <code>X = X + Y</code> .
<code>-=</code>	<code>x -= y</code>	Equivale a <code>X = X - Y</code> .
<code>*=</code>	<code>x *= y</code>	Equivale a <code>X = X * Y</code> .
<code>/=</code>	<code>x /= y</code>	Equivale a <code>X = X / Y</code> .
<code>%=</code>	<code>x %= y</code>	Equivale a <code>X = X % Y</code> .

Expressões aritméticas:

A linguagem C/C++ possui algumas funções matemáticas prontas para serem usadas. Todas elas podem ser observadas detalhadamente na documentação da biblioteca **math.h**. Para se utilizar as funções dessa biblioteca deve-se adicionar a cláusula: `#include <math.h>`. Algumas das funções disponíveis nessa biblioteca são:

Função	Finalidade
<code>abs(i)</code>	Retorna o valor absoluto de i .
<code>ceil(d)</code>	Arredonda para cima, para o próximo valor inteiro maior que d .
<code>cos(d)</code>	Retorna o cosseno de d .
<code>floor(d)</code>	Arredonda para baixo, para o próximo valor inteiro menor que d .
<code>log(d)</code>	Calcula o logaritmo neperiano <code>log(d)</code> .
<code>pow(d1, d2)</code>	Retorna d1 elevado a d2 .
<code>rand()</code>	Retorna um inteiro positivo aleatório.
<code>sin(d)</code>	Retorna o seno de d .
<code>sqrt(d)</code>	Retorna a raiz quadrada de d .
<code>tan(d)</code>	Retorna a tangente de d .

- ✓ As funções acima que possuem retorno, devem ser usadas com cuidado, e precisam de uma variável para receber esse retorno conforme exemplo:

```
potencia = pow (b, 2);
```

- ✓ No exemplo a variável `potencia` vai receber o retorno da função **pow**. Essa função vai retornar o resultado de `b` elevado a dois.

Exemplo

O programa abaixo calcula a hipotenusa de um triângulo retângulo, dados os seus catetos, pelo Teorema de Pitágoras.



Busque nas aulas do nivelamento a fórmula do Teorema de Pitágoras!

```
/*
Programador:<Nome>
Descricao: Calcula a hipotenusa de um triangulo retângulo dados os
seus catetos.
Entrada: Lados b e c de um triângulo retângulo.
Saida: impressao da mensagem.
*/

#include <iostream.h>
#include <math.h>

int main()
{
    float a , b , c;
    cout<<"\n Digite o valor de b: ";
    cin>>b;
    cout<<"\n Digite o valor de c: ";
    cin>>c;
    a = sqrt ( pow(b , 2 ) + pow( c , 2 ) ) ;
    cout << "\n O valor da hipotenusa e: "<< a;
    cout<< "\n\n";

    system("PAUSE");
}
```

Digite e compile o código fonte do exemplo acima. Observe o uso das funções matemáticas usadas no exemplo. Use para o exemplo os seguintes valores **b = 4** e **c = 3**.

Exercícios



Busque nas aulas do nivelamento a fórmula da equação do segundo grau e a fórmula para encontrar suas raízes!

Exercício 1: Considere a equação do segundo grau. Faça um algoritmo (**fluxograma**), em seguida, um programa em C/C++ que encontre as raízes de uma equação do segundo grau dados os coeficientes a, b e c. O usuário deverá **obrigatoriamente** entrar com os coeficientes a, b e c.

Retorne para o usuário as raízes da equação. Na tela, os valores de x1 e x2 deverão ser exibidos alinhados, um embaixo do outro.

Por exemplo:

x1:	1
x2:	0

Exercício 2: Faça um programa em C/C++ para ler três números inteiros do teclado. A saída na tela deve ser o primeiro número ao cubo, o triplo do segundo número e a raiz quadrada do terceiro número.

Exercício 3: Faça um programa em C/C++ para calcular a área de um trapézio. O programa deve ler do teclado o valor da base menor, base maior e a altura. Em seguida, imprimir na tela o valor da área do trapézio.



• *Veja a aula 12 do nivelamento: Equacionando Problemas (A Letra como Variável)!*

Exercício 4: Faça um programa em C/C++ que receba o preço de um produto, calcule e mostre o novo preço, sabendo –se que:

- a) o preço do produto sofreu um desconto de **10%**
- b) o preço do produto sofreu um aumento de **20%**



• *Veja a aula do nivelamento sobre porcentagens!*

Roteiro 3 – Estrutura Condicional em C/C++

Uma das tarefas fundamentais de qualquer programa é decidir o que deve ser executado, para isso, temos comandos de decisão que permitem determinar qual a ação deve ser tomada a partir de um resultado de uma expressão condicional. Em C/C++ temos os comandos de decisão:

if
if – else

Estrutura condicional simples:

```
# include <nome da biblioteca>
int main()
{
    if (condição)
    {
        Bloco de comandos;
    }
}
```

- ✓ Observe que o **bloco de comandos** só será executado se a **condição** for verdadeira, Uma condição é uma comparação que possui dois valores possíveis, verdadeiro ou falso.
- ✓ A condição deve estar entre **parênteses ()**.
- ✓ Em C/C++, torna-se obrigatória a utilização de chaves quando houver mais de um comando a ser executado. Os comandos entre as chaves { } só serão executados se a **condição** for verdadeira.

Estrutura condicional composta:

```
# include <nome da biblioteca>
int main()
{
    if (condição)
    {
        Bloco de comandos1;
    }

    else
    {
        Bloco de comandos2;
    }
}
```


- ✓ Se a **condição** for *verdadeira*, será executado o **Bloco de comandos1**, caso contrário, se a **condição** for *falsa*, será executado o **Bloco de comandos2**.

Operadores relacionais:

Operador	Exemplo	Comentário
=	x == y	O conteúdo de X é igual ao conteúdo de Y
!=	x != y	O conteúdo de X é diferente do conteúdo de Y
<=	x <= y	O conteúdo de X é menor ou igual ao conteúdo de Y
>=	x >= y	O conteúdo de X é maior ou igual ao conteúdo de Y
<	x < y	O conteúdo de X é menor que o conteúdo de Y
>	x > y	O conteúdo de X é maior que o conteúdo de Y

Operadores lógicos:

Operador	Exemplo	Comentário
&&	if (x >= 3 && x < 7)	Significa que a condição verificada será X >= 3 E X < 7, ou seja, a condição será verdadeira se X estiver entre 3 e 7
	if (x >= 3 x >= 7)	Significa que a condição verificada será X >= 3 OU X <= 7, ou seja, tanto se X for maior que 3 como se X for maior que 7 a condição será verdadeira
!	!x	A ! (NÃO) é a <i>negação</i> , ou seja, muda um valor lógico de verdadeiro para falso e de falso para verdadeiro - Resulta 1 (verdadeiro) somente se x for falsa. - Resulta 0 (falso) somente se x for verdadeiro.

Tabela verdade:

Tabela E (&&)	Tabela OU ()	Tabela NÃO (!)
V e V = V	V ou V = V	Não V = F
V e F = F	V ou F = V	Não F = V
F e V = F	F ou V = V	
F e F = F	F ou F = F	

Exemplos

Exemplo 1 - Digite e compile o código fonte abaixo:

```
//Lembre-se de colocar as bibliotecas

int main()
{
    float valor;
    cout<<"\n Digite um numero: ";
    cin>> valor;
    if(valor > 0)
    {
        cout<<"\n o numero digitado e maior que ZERO";
    }
    if(valor == 0)
    {
        cout<<"\n o numero digitado e igual a ZERO";
    }
    if(valor < 0)
    {
        cout<<"\n o numero digitado e menor que ZERO";
    }
    cout<<"\n\n";
    system("PAUSE");
}
```

O que o programa faz? Coloque **comentários** no programa para explicar o que cada comando faz.

Exemplo 2 - Digite e compile o código fonte abaixo:

```
int main()
{
    float valor;
    cout<<"\n Digite um numero: ";
    cin>> valor;
    if(valor > 0)
    {
        cout<<"\n o numero digitado e maior que ZERO";
    }
    else if(valor == 0)
    {
        cout<<"\n o numero digitado e igual a ZERO";
    }
    else
    {
        cout<<"\n o numero digitado e menor que ZERO";
    }
    system("PAUSE");
}
```

O que o programa faz? Qual a diferença entre o **exemplo 1** e o **exemplo 2** ?

Exemplo 3 - Digite e compile o código fonte abaixo:

```
int main()
{
    float num1, num2;
    cout<<"\n Digite um numero: ";
    cin>> num1;
    cout<<"\n Digite outro numero: ";
    cin>> num2;

    if(num1> num2 )
    {
        cout<<"\n o maior numero digitado e: "<<num1;
    }
    else if(num2> num1)
    {
        cout<<"\n o maior numero digitado e: "<<num2;
    }
    else
    {
        cout<<"\n os numeros digitados sao iguais";
    }
    system("PAUSE");
}
```

O que o programa faz? O **exemplo 3** poderia usar a **estrutura condicional simples**?

Exercícios

Exercício 1 - Faça um fluxograma, em seguida, um programa em C/C++ que leia o primeiro e o último número da matrícula, em seguida escreva na tela qual é o maior número, se o primeiro ou o último número.

Exercício 2 - Criar um programa em C/C++ que solicite a entrada de um número, em seguida imprime na tela se o número digitado é **positivo**, **negativo** ou **nulo**.

Exercício 3 - Faça um programa em C/C++ que mostre o menu de opções a seguir, receba a opção do usuário e os dados necessários para executar cada operação.

Menu de opções:

1- Somar dois números

2- Raiz quadrada de número

Observação: Para calcular a raiz, será necessário validar se o número digitado é maior que Zero.

Exercício 4 - Faça um programa em C/C++ que receba o código correspondente ao cargo de um funcionário e seu salário atual. O programa deverá calcular o aumento e mostrar na tela o cargo, o aumento e o novo salário. Os cargos e o percentual de aumento estão na tabela a seguir:

Código	Cargo	Percentual de aumento
1	Escriturário	50%
2	Secretário	35%
3	Caixa	20%
4	Gerente	20%
5	Diretor	Não tem aumento

Roteiro 4 – Estrutura de Repetição em C/C++

Estrutura de Repetição em C++ **FOR**

Essa estrutura é utilizada quando se sabe o número de vezes que um trecho do programa deve ser repetido. O formato geral do comando **for** é composto por três partes:

```
for (i = valor inicial; condição; incremento ou decremento de i)
{
    Bloco de comandos;
}
```

Estrutura de Repetição em C++ **WHILE**

O **while** é uma estrutura de repetição que pode ser utilizada quando o número de repetições necessárias não é fixo. Os comandos serão repetidos até a condição assumir o valor falso. Nesse tipo de estrutura, o teste condicional ocorre no início. Isto significa que existe a possibilidade da repetição não ser executada quando a condição assumir valor falso logo na primeira verificação.

```
while (condição)
{
    Bloco de comandos;
}
```

Estrutura de Repetição em C++ **DO-WHILE**

O **do-while** é uma estrutura de repetição que pode ser utilizada quando o número de repetições necessárias não é fixo. Os comandos serão repetidos até a condição assumir o valor falso. Nesse tipo de estrutura, o teste condicional ocorre no fim. Isto significa que a repetição será executada no mínimo uma vez, quando todo o bloco for executado uma vez e, ao final, a condição assumir valor falso.

```
do{
    Bloco de comandos;
} while (condição);
```

Exemplos

Exemplo 1 - Digite e compile o código fonte abaixo:

```
//Lembre-se das bibliotecas

int main()
{
    int quant=0, num, i;
    for (i = 1; i <= 5; i++)
    {
        cout <<"\n Entre um valor inteiro: ";
        cin >> num;
        if(num>5)
        {
            quant = quant + 1;
        }
    }
    cout<<"\n Foram digitados "<<quant<<" numeros maiores que 5";
    cout<<"\n\n";
    system("PAUSE");
}
```

O que o programa faz? Coloque comentários no programa para explicar o que cada comando faz.

Exemplo 2 - Digite e compile o código fonte abaixo:

```
//Bibliotecas...

int main()
{
    int num, N, i=0, soma=0;
    cout<<"\nQuantos numeros deseja digitar: ";
    cin>>N;
    while(i < N)
    {
        cout <<"\n Entre um valor inteiro: ";
        cin >> num;
        soma = soma + num;
        i++;
    }
    cout<<"\n Foram digitados "<<N<<" numeros";
    cout<<"\n A soma dos numeros digitados e: "<<soma;
    cout<<"\n\n";
    system("PAUSE");
}
```

O que o programa faz? Coloque comentários no programa para explicar o que cada comando faz.

Exemplo 3 - Digite e compile o código fonte abaixo:

```
//Bibliotecas...

int main()
{
    int contador, numero, n;
    contador = 1;

    cout << " Entre com a quantidade de vezes para repetir: ";
    cin >> n;

    do {
        cout << "\n Digite um numero inteiro: ";
        cin >> numero;
        cout << "\n O numero digitado foi " << numero;
        contador = contador + 1;
    } while (contador <= n);

    cout << "\n\n => Digite ENTER para terminar";
    system("PAUSE");
}
```

O que o programa faz? Coloque comentários no programa para explicar o que cada comando faz.

Exemplo 4 - Digite e compile o código fonte abaixo:

```
//Bibliotecas...
int main()
{
    int saida, numero, cont=0;

    do
    {
        cout<<"\n Digite um numero inteiro: ";
        cin>> numero;
        cont++;
        cout<<"\n Para sair digite -1 e para continuar outro numero ";
        cin>>saida;
    } while (saida != -1);

    cout << "\n\n Foram digitados "<<cont<<" numeros";

    system("PAUSE");
}
```

O que o programa faz? Coloque comentários no programa para explicar o que cada comando faz.

Exercícios



Veja a aula do nivelamento de matemática sobre porcentagens e média aritmética.

Exercício 1 - Faça um programa em C/C++ que receba **dez** números, calcule e mostre a soma dos números pares e a quantidade de números ímpares digitados.

Exercício 2 - Faça um programa que receba a idade de **oito** pessoas, calcule e mostre:

- A quantidade de pessoas em cada faixa etária;
- A porcentagem de pessoas na primeira faixa etária com relação ao total de pessoas;
- A porcentagem de pessoas na última faixa etária com relação ao total de pessoas.

Use a tabela a seguir para as faixas etárias:

Faixa etária	Idade
1ª	Até 15 anos
2ª	De 16 a 30 anos
3ª	De 31 a 45 anos
4ª	De 46 a 60 anos
5ª	Acima de 60 anos

Exercício 3 - Faça um programa que receba uma seqüência de números e retorne o **maior** número e o **menor** número da seqüência digitada. A quantidade de números n é fornecida pelo usuário. Use a estrutura **WHILE**.

Exercício 4 - Faça um programa que leia números inteiros, calcule e mostre a **quantidade** de números divisíveis por 3 e a **quantidade** de números divisíveis por 7 dos números informados. O programa é finalizado ao usuário informar um número negativo. Use a estrutura **DO-WHILE**.

Exercício 5 - Em uma eleição presidencial existem quatro candidatos. Os votos são informados por meio de código. Os códigos utilizados são:

1, 2, 3, 4	Votos para os respectivos candidatos
5	Voto nulo
6	Voto em branco

Faça um programa que calcule e mostre: o **total** de votos para cada candidato; o **total** de votos nulos; a **porcentagem** de votos em branco sobre o total de votos.

Para finalizar o conjunto de votos digitados, tem-se o valor zero e, para códigos inválidos, o programa deverá mostrar uma mensagem "Voto inválido!".

Roteiro 5 – Vetor em C/C++

- ✓ Vetor são variáveis compostas homogêneas unidimensionais capazes de armazenar vários valores.
- ✓ Cada um desses valores é identificado pelo mesmo nome sendo diferenciado apenas por um índice.
- ✓ Os índices utilizados para identificar as posições de um vetor em C/C++ começam sempre em **0 (zero)** e vão até o **tamanho do vetor menos uma unidade**.

índice	→	0	1	2	3
valor	→	20	12	3	13

- ✓ Para acessar os elementos do vetor, deve utilizar o valor do índice desejado, juntamente com o nome da variável, por exemplo, **peso[2]** está associado ao terceiro elemento do vetor pois o primeiro elemento está relacionado ao índice **0**

Declaração de um vetor:

- ✓ Para declarar um vetor:

Tipo **nome_Vetor** [tamanho_do_vetor];

```
# include <nome da biblioteca>

int main()
{
    int peso[10];
    float nota[41];
    char nome[80];
}
```

Preenchendo um vetor:

- ✓ Preencher um vetor significa atribuir valores para todas as posições de um vetor.

```
for (int i = 0; i < tamanho do vetor; i++)
{
    cin >> nome_vetor[i];
}
```

Mostrando um vetor:

- ✓ Para mostrar todas as posições de um vetor.

```
for (int i = 0; i < tamanho do vetor; i++)  
{  
    cout << nome_vetor[i];  
}
```

Exemplos

Exemplo 1 - Digite e compile o código fonte abaixo:

```
#include <iostream.h>  
  
int main()  
{  
    float notas[5] = {10,5,8,2,8};  
    int i=0;  
  
    cout<< "\n Impressao dos elementos do vetor";  
  
    for(i=0; i < 5; i++)  
    {  
        cout<<"\nNota "<<(i+1)<<": "<<notas[i];  
    }  
  
    cout<<"\n\t\tFim do programa\n";  
    system("PAUSE");  
}
```

O que o programa faz? Coloque comentários no programa para explicar o que cada comando faz.

Exemplo 2 - Digite e compile o código fonte abaixo:

```
//Bibliotecas...  
  
int main()  
{  
    float notas[5];  
    float media=0, soma=0, menor=0;  
  
    int i=0;
```

```
for(i=0; i < 5; i++)
{
    cout<<"\nDigite a nota " <<(i+1)<<": ";

    cin>>notas[i];

}
menor = notas[0];

for(i=0; i < 5; i++)
{
    soma = soma + notas[i];

    if (notas[i] < menor)
    {
        menor = notas[i];
    }

}

media = soma / 5;
cout<<"\n Soma total = "<<soma;
cout<<"\n Média      = "<<media;
cout<<"\n Menor nota = "<<menor;
system("PAUSE");
}
```

O que o programa faz? Coloque comentários no programa para explicar o que cada comando faz.

Exercícios

Exercício 1 - Faça um programa em C/C++ que receba **vinete** números, calcule e mostre a soma dos números positivos e a quantidade de números negativos digitados. **Lembre-se de usar vetor.**

Exercício 2 - Faça um programa em C/C++ que preencha **dois** vetores de cinco elementos numéricos cada um e mostre o vetor resultante da intercalação deles. Como no exemplo abaixo:

	0	1	2	3	4
Vetor1	3	5	4	2	2

	0	1	2	3	4
Vetor2	7	15	20	0	18

	0	1	2	3	4	5	6	7	8	9
Vetor Resultante	3	7	5	15	4	20	2	0	2	18

Exercício 3 - Faça um programa que preencha um vetor com 15 números inteiros, calcule e mostre:

- A quantidade de posições com elementos iguais a 2
- Os elementos (números) múltiplos de 3
- As posições que possuem elementos (números) múltiplos de 2

Exercício 4 - Faça um programa que solicite que sejam digitados e armazenados 10 números em um vetor de inteiros chamado **vetorOriginal**. Logo em seguida o programa deve guardar os 10 números de maneira invertida em **outro vetor** chamado **vetorInvertido**. Mostrar na tela o vetor **vetorOriginal** e o vetor **vetorInvertido**.

	0	1	2	3	4	5	6	7	8	9
Vetor Original	3	7	5	15	4	20	2	0	2	18

	0	1	2	3	4	5	6	7	8	9
Vetor Invertido	18	2	0	2	20	4	15	5	7	3

Observação: Nesse exercícios são criados dois vetores (original e invertido)

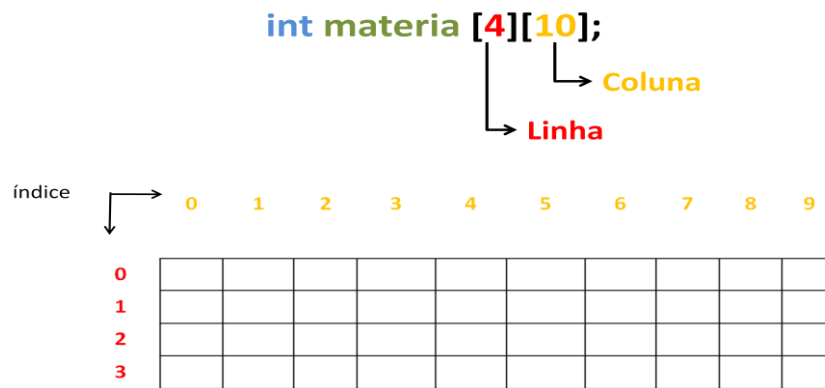
Exercício 5 - Faça um programa que carregue um vetor com 10 números inteiros digitados pelo usuário. Em seguida, calcule e **mostre** o **mesmo vetor** ordenado de maneira **decrescente**.

	0	1	2	3	4	5	6	7	8	9
Vetor início	3	7	5	15	4	20	2	0	1	18

	0	1	2	3	4	5	6	7	8	9
Vetor fim	20	18	15	7	5	4	3	2	1	0

Roteiro 6 – Matriz em C/C++

- ✓ Uma **matriz** pode ser definida como um conjunto de variáveis de mesmo tipo e identificadas pelo mesmo nome (variável composta homogênea multidimensional).
- ✓ Essas variáveis são diferenciadas por meio da especificação de suas posições dentro dessa estrutura.
- ✓ C/C++ permite a declaração de: Matrizes **unidimensionais** (mais conhecidas como *vetores*) e Matrizes **bidimensionais** e **multidimensionais** (o limite de dimensões fica por conta da quantidade de recurso disponível pelo compilador. Entretanto, as matrizes mais utilizadas possuem duas dimensões, Linhas e Colunas).
- ✓ Da mesma maneira como ocorre com os vetores, os índices começam sempre em **0 (zero)**. O exemplo abaixo apresenta a declaração de uma variável chamada *materia* contendo 4 linhas (0 a 3) e 10 colunas (0 a 9), capazes de armazenar números inteiros.



Declaração de uma matriz:

- ✓ Para declarar uma matriz:

Tipo **nome_Matriz** [qtdes de Linhas] [qtdes de Colunas];

```
# include <nome da biblioteca>
int main()
{
    int nota[10][5];
    float peso[4][30];
}
```

Preenchendo uma matriz:

- ✓ Preencher uma matriz significa atribuir valores para todas as posições (linhas e colunas) de uma matriz.

```
for (int L = 0; L < qtde de linhas; L++)  
{  
    for (int C = 0; C < qtde de colunas; C++)  
    {  
        cin >> nome_matriz[L][C];  
    }  
}
```

Mostrando uma matriz:

- ✓ Para mostrar todas as posições (linhas e colunas) de uma matriz.

```
for (int L = 0; L < qtde de linhas; L++)  
{  
    for (int C = 0; C < qtde de colunas; C++)  
    {  
        cout << nome_matriz[L][C];  
    }  
}
```

Exemplos

Exemplo 1 - Digite e compile o código fonte abaixo:

```
//Bibliotecas  
  
int main ()  
{  
    int preco[3][4],linha,coluna,cont=0, maior;  
    for (linha=0;linha<3;linha++)  
    {  
        for (coluna=0;coluna<4;coluna++)  
        {  
            cout<<"\n Digite o preco do produto "<<linha<<" da loja "<<coluna<<" : ";  
            cin>>preco[linha][coluna];  
        }  
    }  
    maior = preco[0][0];  
    for (linha=0;linha<3;linha++)  
    {  
        for (coluna=0;coluna<4;coluna++)  
        {  
            if(preco[linha][coluna]>100)  
            {  
                cont++;  
            }  
        }  
    }  
}
```

```
        if(preco[linha][coluna]>maior)
        {
            maior = preco[linha][coluna];
        }
    }
}
cout<<"\nMaior preco: "<<maior;
cout<<"\nA quantidade de produtos com preco maior que 100 e: "<<cont;
system("PAUSE");
}
```

O que o programa faz? Coloque comentários no programa para explicar o que cada comando faz.

Exercícios

Exercício 1 - Faça um programa que carregue uma matriz 4 X 5, calcule e mostre um vetor com cinco posições, onde cada posição contém a soma dos elementos de cada **coluna** da matriz. Em seguida, mostre o vetor e a matriz na tela.

Exercício 2 - Faça um programa C/C++ que preencha uma matriz de dimensões digitadas pelo usuário e mostre o número de elementos maiores que 15 e menores que 25.

Exercício 3 - Faça um programa C/C++ que preencha uma matriz **8 x 6** de inteiros, calcule e mostre a média dos elementos das **linhas pares** da mesma.

Exercício 4 - Faça um programa C/C++ que carregue uma matriz **5 X 5** com números inteiros, calcule e mostre a soma:

- a) dos elementos da linha 4;
- b) dos elementos da coluna 2;
- c) de todos os elementos da matriz.

Exercício 5 - Faça um programa que receba o estoque atual de três produtos, armazenados em quatro armazéns e coloque esses dados em uma matriz 4 x 3. Em seguida, o programa deverá calcular e mostrar:

- a) a quantidade de itens armazenados em cada armazém
- b) qual armazém possui maior estoque do produto 2
- c) qual armazém possui menor estoque do produto 1

	Produto 1	Produto 2	Produto 3
Armazém 1			
Armazém 2			
Armazém 3			
Armazém 4			

Listas de Exercícios

Lista 1 – Estrutura Sequencial em C/C++

Antes de começar a fazer essa lista de exercícios, veja a aula de Raciocínio Lógico do nivelamento de matemática.

- 1- Responda as perguntas a seguir:
 - a. O que é um algoritmo?
 - b. Quais são os tipos mais utilizados de algoritmos? Apresente a vantagem e desvantagem do uso de cada um.
 - c. O que é uma variável e qual o seu objetivo em um programa de computador?
- 2- Faça um algoritmo (**fluxograma e pseudocódigo**), em seguida, um programa que leia uma temperatura dada na escala Celsius (C) e imprima na tela o equivalente em Fahrenheit (F).

Fórmula de conversão: $F = C * 1.8 + 32$

- 3- Faça um algoritmo (**fluxograma e pseudocódigo**), em seguida, um programa que calcule o quadrado de um número qualquer, ou seja, o produto desse número por ele mesmo. Imprima na tela o resultado final.
- 4- Faça um algoritmo (**fluxograma e pseudocódigo**), em seguida, um programa que calcule a área de um triângulo. O usuário deverá digitar o valor da base e da altura. Em seguida deverá ser apresentado na tela o valor final da área do triângulo.

Fórmula da área do triângulo: $(base * altura) / 2$
--

- 5- Um funcionário recebe um salário fixo mais 4% de comissão sobre as vendas. Faça um algoritmo (**fluxograma e pseudocódigo**), em seguida, um programa que receba o salário fixo do funcionário e o valor de suas vendas no mês, calcule e mostre a comissão e seu salário final.

Lista 2 – Estrutura Condicional em C/C++

- 1- Faça um algoritmo (pseudocódigo e fluxograma), em seguida, um programa que receba dois números e mostre na tela o menor.
- 2- Faça um algoritmo (pseudocódigo e fluxograma), em seguida, um programa que receba três números e mostre na tela o maior.
- 3- Faça um algoritmo (pseudocódigo e fluxograma), em seguida um programa que receba quatro notas de um aluno, calcule e mostre a média aritmética das notas e a mensagem de aprovado ou reprovado, considerando para aprovado média maior ou igual a 7.
- 4- Faça um algoritmo (pseudocódigo e fluxograma), em seguida um programa que leia dois números inteiros. Se um deles for menor que 20, realize a soma dos mesmos e imprima na tela o resultado. Se os dois forem maiores que 30, realize a subtração do maior pelo menor e imprima na tela o resultado. Senão, se nenhum dos casos solicitados for válido, imprima na tela os números que foram digitados.
- 5- Faça um algoritmo (pseudocódigo e fluxograma), em seguida um programa que receba um número inteiro e verifique se esse número é par ou ímpar.
- 6- Faça um programa que receba a idade de um nadador e mostre a sua categoria usando as regras da tabela abaixo:

Categoria	Idade
Infantil	5 a 7
Juvenil	8 a 10
Adolescente	11 a 15
Adulto	16 a 30
Sênior	Acima de 30

- 7- Faça um programa que receba o salário de um funcionário, calcule e mostre o novo salário, acrescido de bonificação e de auxílio escola conforme tabela abaixo:

Salário	Bonificação
Até R\$ 500,00	5% do salário
Entre R\$ 500,00 e R\$ 1200,00	12% do salário
Acima de R\$ 1200,00	Sem bonificação

Salário	Auxílio Escola
Até R\$ 600,00	R\$ 150,00
Mais que R\$ 600,00	R\$ 100,00

Lista 3 – Estrutura de Repetição em C/C++

1- Responda:

- Por que usamos estruturas de repetição no desenvolvimento de programas?
- Quando devemos usar a estrutura de repetição for e quando devemos usar as estruturas while e do-while?

2- Faça um algoritmo (pseudocódigo e fluxograma), em seguida, um programa para imprimir na tela os 10 primeiros números inteiros maiores que 100 utilizando as estruturas de repetição (PARA, ENQUANTO, REPITA-ATÉ) e (for, while, do-while).

3- Faça um algoritmo (pseudocódigo e fluxograma), em seguida, um programa que receba um número inteiro maior que 1, verifique se o número fornecido é primo ou não e mostre a mensagem de número primo ou de número não primo. Use estrutura de repetição. (*obs. Um número é primo quando é divisível apenas por 1 e por ele mesmo*).

4- Fazer um programa que calcula e mostre na tela o *N*-ésimo termo da sequência de Fibonacci (0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, ...) utilizando a estrutura de repetição for.

5- Dado o código abaixo indique o resultado do mesmo para cada um dos valores de “val”.

```
int main()
{
    int i, n = 3;
    int val = ??;

    for (i=0; i<=5; i++)
    {
        val = val + n;
    }

    n = 5;
    cout<< val;

    getch();
}
```

Substitua o valor do símbolo “??” no código acima por cada um dos valores apresentados abaixo. E mostre o resultado final do programa para cada dos valores de val.

a)	val = -1	Resposta =
b)	val = 0	Resposta =
c)	val = 2	Resposta =
d)	val = 3	Resposta =

6- Faça um programa que receba **vários números**, finalize a entrada de números no programa com a digitação do número **-1**, calcule e mostre na tela:

- A soma dos números digitados;
- A quantidade de números digitados;
- A média dos números digitados;
- O maior número digitado;
- O menor número digitado.

7- Construa um programa que lê uma opção conforme abaixo e o salário atual do funcionário, calcula e exibe o novo salário. Deve-se repetir o cálculo para diversos funcionários, enquanto for informado um salário válido (maior do que zero).

A – Para aumento de 8%

B – Para aumento de 11%

C – Para aumento fixo de R\$ 450,00

8- Faça um programa para calcular a área de um triângulo, que **NÃO** permita a entrada de dados inválidos, ou seja, medidas menores ou iguais a zero.

9- Faça um programa que receba duas notas de 6 alunos, calcule e mostre :

- A média aritmética das duas notas de cada aluno;
- A mensagem de acordo com a tabela abaixo:

Média aritmética	Mensagem
Até 3	Reprovado
Entre 3 e 7	Exame
Acima de 7	Aprovado

- O total de alunos aprovados;
- O total de alunos de exame;
- O total de alunos reprovados;
- A média da classe.

Lista 4 – Vetor em C/C++

- 1- Faça um algoritmo (pseudocódigo e fluxograma), em seguida, um programa que carregue um vetor com 15 elementos inteiros e verifique a existência de elementos iguais a 30, mostrando na tela as posições em que esses elementos aparecem no vetor.
- 2- Faça um algoritmo (pseudocódigo e fluxograma), em seguida, um programa que carregue um vetor de seis elementos numéricos inteiros, calcule e mostre na tela:
 - A quantidade de números pares e quais são os números pares
 - A quantidade de números ímpares e quais são os números ímpares
- 3- Faça um programa que leia dois vetores (**A** e **B**) de cinquenta posições de números inteiros. O programa deve, então, subtrair o primeiro elemento de A do último de B, acumulando o valor, subtrair o segundo elemento de A do penúltimo de B, acumulando o valor e assim por diante. Mostre na tela o resultado da soma de todas as subtrações.
- 4- Faça um programa que carregue um vetor com 10 números inteiros digitados pelo usuário. Em seguida, calcule e mostre o mesmo vetor ordenado de maneira **crescente**.

	0	1	2	3	4	5	6	7	8	9
<i>vetor</i>	3	5	4	2	1	6	8	7	11	9

<i>vetor ordenado</i>	0	1	2	3	4	5	6	7	8	9
	1	2	3	4	5	6	7	8	9	11

- 5- Faça um programa que lê N números informados pelo usuário e armazena em um vetor. O valor de N é informado pelo usuário, ao final exibir o conteúdo armazenado no vetor, a quantidade de números positivos e a quantidade de números negativos.
- 6- Dado o programa em C++ abaixo:

<pre>int main() { int vetor[6]; for(int i=0; i<6; i++) { vetor[i] = i + 2; } cout<<"\n"<<vetor[0]; cout<<"\n"<<vetor[2]; cout<<"\n"<<vetor[4]; getch(); }</pre>	<p>Quais são os valores que serão mostrados na tela para as seguintes posições do vetor?</p> <p>a) vetor[0] = ____</p> <p>b) vetor[2] = ____</p> <p>c) vetor[4] = ____</p>
---	--

7- Uma academia tem 30 alunos. Faça um programa para ler o peso de todos os alunos e logo em seguida imprimir:

- Total de alunos com peso maior que 70 kg
- Média dos pesos
- Maior peso (peso do aluno mais gordo)
- Menor peso (peso do aluno mais magro)

Lista 5 – Matriz em C/C++

- 1- Faça um algoritmo (pseudocódigo e fluxograma), em seguida, um programa que carregue uma matriz **2 x 4** com números inteiros, calcule e mostre na tela:
 - a quantidade de elementos entre **12 e 20** em **cada linha**;
 - a média dos **elementos pares** da matriz.
- 2- Faça um algoritmo (pseudocódigo e fluxograma), em seguida, um programa que declare uma matriz de tamanho **10 x 10**. Logo em seguida, o programa deve em cada posição i, j armazena o valor (i * j). Logo em seguida imprima o conteúdo da matriz.
- 3- Faça um algoritmo (pseudocódigo e fluxograma), em seguida, um programa que carregue uma matriz **3 x 5** com números inteiros. Em seguida, verifique a quantidade de elementos digitados entre 15 e 20. Mostre na tela essa quantidade.
- 4- Observe o código abaixo. Quais são os valores da matriz que serão impressos na tela?

```
int main ()
{
    int mtrx [3][4],i,j,cont=1;
    for (i=0;i<3;i++)
    {
        for (j=0;j<4;j++)
        {
            mtrx[i][j]=cont+2;
            cont++;
        }
    }
    cout<<"\n"<<mtrx[0][1];
    cout<<"\n"<<mtrx[1][2];
    cout<<"\n"<<mtrx[2][3];
    getch();
}
```

- 5- Faça um programa que preencha uma matriz **8 x 6** de números inteiros. Em seguida, calcule e mostre a **média dos elementos das linhas** da matriz.
- 6- Faça um programa que preencha uma matriz **5 x 3** de números inteiros. Em seguida, calcule e mostre a **quantidade de elementos pares nas colunas** da matriz.

Núcleo de Algoritmos

Ana Paula Ladeira (ana.ladeira@prof.una.br)

Ivre Marjorie Ribeiro Machado (ivre.machado@prof.una.br)

Roberto de Oliveira Campos Júnior (roberto.junior@prof.una.br)