ATIVIDADES DIDÁTICAS:

UM GUIA PARA PROFESSORES

CRIPTOGRAFIA

CIÊNCIA E ARTE DE OCULTAR MENSAGENS COM PRECISÃO MATEMÁTICA

JOÃO OTAVIO FURTADO DA SILVA

SÃO JOSÉ DO RIO PRETO - 2025

UNIVERSIDADE ESTADUAL PAULISTA – UNESP

Instituto de Biociências, Letras e Ciências Exatas

MESTRADO PROFISSIONAL EM MATEMÁTICA EM REDE NACIONAL

ATIVIDADES DIDÁTICAS COM CRIPTOGRAFIA: UM GUIA PARA PROFESSORES

JOÃO OTAVIO FURTADO DA SILVA

São José do Rio Preto
2025

PROFMAT

UNIVERSIDADE ESTADUAL PAULISTA – UNESP

Instituto de Biociências, Letras e Ciências Exatas

MESTRADO PROFISSIONAL EM MATEMÁTICA EM REDE NACIONAL

Nível: Mestrado Profissional

Área de concentração: Matemática do Ensino Básico

Autor: João Otavio Furtado Da Silva

Coautor (Orientador): Prof. Dr. Jéfferson Luiz Rocha Bastos

Produto Educacional: Guia prática para Professores

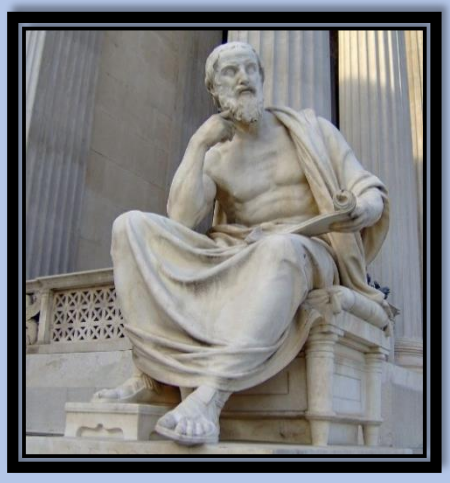
Nível de ensino: Ensino Básico

Área de conhecimento: Matemática e suas Tecnologias

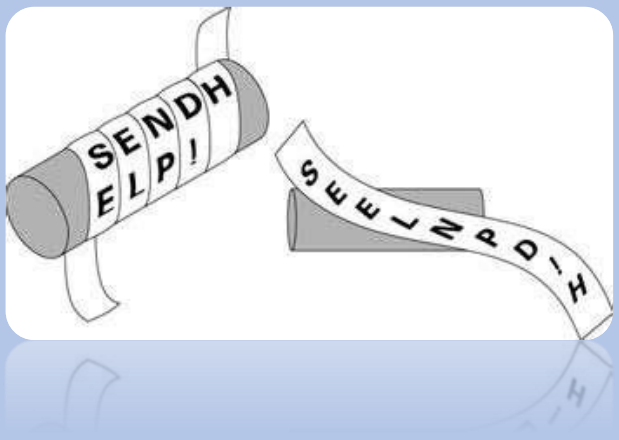
SUMÁRIO

INTRODUÇÃO

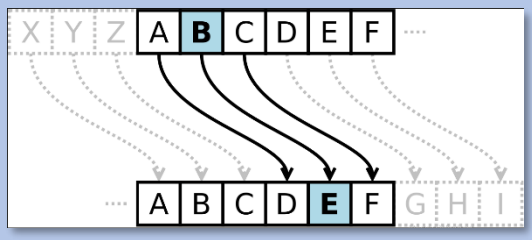
A **criptografia,** termo derivado do grego kryptós (**oculto**) e gráphein (**escrita**), é a **ciência de esconder mensagens** e garantir a **confidencialidade** da informação. Antes mesmo da escrita formal, povos antigos já utilizavam **métodos engenhosos** para ocultar conteúdos estratégicos, demonstrando uma **preocupação** ancestral com a **segurança** da informação. **Heródoto** (figura ao lado) em sua obra **História** descreve uma **estratégia** utilizada por **Histieu,** tirano de Mileto, que encontrou uma forma **curiosa** de enviar uma mensagem secreta a **Aristágoras**, líder da Revolta Jônica. Ele **raspou a cabeça de um escravo,** tatuou a **mensagem no couro cabeludo** e esperou o cabelo crescer. Quando o escravo chegou ao destino, **Aristágoras** raspou novamente sua cabeça **e leu as instruções ocultas**, uma forma **engenhosa** de **transmitir dados** sigilosos sem chamar atenção.



Entre os **dispositivos** históricos, destaca-se a **scytale espartana** (figura abaixo) , um **bastão cilíndrico** em torno do qual se enrolava uma **tira de couro** contendo o texto. A **mensagem** apenas podia ser lida corretamente quando a **fita** era **enrolada** em outro **bastão** de **mesmo diâmetro**, configurando um exemplo clássico de **cifra de transposição**.



No campo das cifras de substituição, destaca-se a **Cifra de César,** atribuída ao general romano **Júlio César** (figura ao lado), que, no século I a.C., utilizava essa **técnica** para **proteger comunicações** militares sigilosas. Trata-se de uma **cifra** de **substituição** monoalfabética, na qual cada letra do texto claro é **substituída** por outra **letra** do alfabeto, **deslocada** um número fixo de posições. O **deslocamento** tradicional utilizado pelo próprio **César,** é de **três letras** à frente, gerando um **sistema** simples, porém **eficiente**, para sua época.

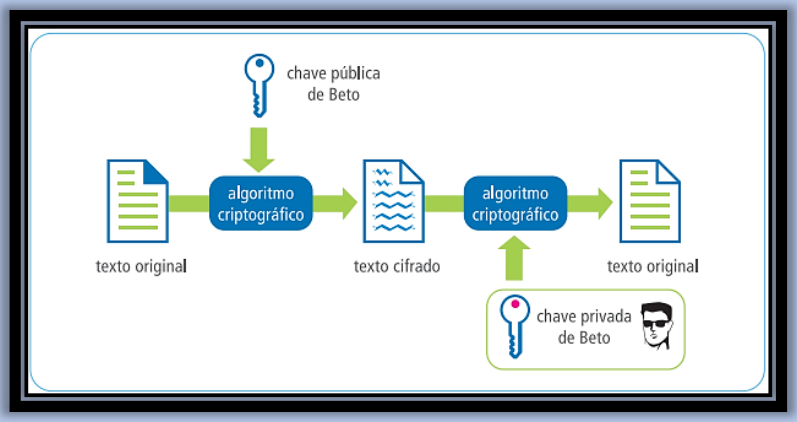


Durante a **Idade Média**, o mundo **árabe** foi responsável por **avanços** teóricos **significativos**, como a **análise de frequência**, atribuída a **Al-Kindi**, **filósofo** e **cientista** árabe que permitiu **quebrar cifras** monoalfabéticas por meio do **estudo estatístico** das **letras** mais recorrentes. Já no **Renascimento**, a **Cifra de Vigenère** trouxe um novo patamar de **segurança** ao empregar uma **palavra-chave** que define **múltiplos deslocamentos,** tornando-a uma **cifra polialfabética** mais **resistente.**

No **século XX**, a **criptografia** ganhou **relevância** decisiva durante as guerras mundiais: a **Máquina Enigma** (figura ao lado), usada pela **Alemanha** e **decifrada** por **Alan Turing**, matemático e lógico

britânico, que marcou um **ponto** de virada **histórico**, dando origem à era da **computação moderna**.

Com o **avanço tecnológico**, a **criptografia** consolidou-se como um **pilar** da **segurança digital** contemporânea. Surgiram **dois** grandes **modelos**: a **criptografia simétrica**, que utiliza uma mesma **chave** para **cifrar e decifrar**, e a **criptografia assimétrica**, baseada em pares de **chaves públicas** e **privadas**, como no algoritmo **RSA.** Hoje, a **criptografia** está **presente** em diversas áreas do cotidiano, como **transações bancárias, mensagens** instantâneas, **redes** sem **fio** e sistemas de **blockchain,** revelando-se **indispensável** à preservação da **privacidade** e da **integridade** das **informações.**



Além de sua **importância tecnológica**, a **criptografia** apresenta-se como um **recurso didático** capaz de **despertar** a **curiosidade** e o **raciocínio lógico** dos estudantes. Ao **decifrar mensagens** e explorar **padrões,** os alunos aplicam **conceitos matemáticos** de forma **prática** e **contextualizada**, desenvolvendo **competências** previstas na **Base Nacional Comum Curricular** (BNCC), como o **pensamento lógico**, a **argumentação** e a **resolução de problemas**. Assim, os **códigos** e **cifras** se tornam pontes entre a **Matemática** e outras áreas do **conhecimento**, como **Linguagens**, **História** e **Informática**, favorecendo a **interdisciplinaridade** e a **aprendizagem significativa**.

As propostas aqui reunidas buscam **atender** diferentes níveis de ensino, podendo ser **adaptadas** para turmas do **ensino fundamental** e **médio**. Os objetivos didáticos principais incluem:

- Introduzir os **conceitos básicos** de **criptografia** e suas aplicações históricas e contemporâneas;
- Desenvolver a compreensão de **operações aritméticas** modulares e funções **matemáticas** no contexto da **codificação** de mensagens;
- Estimular a **resolução** de **problemas,** a **análise** de padrões e a construção de **estratégias**;
- Promover o trabalho colaborativo e a **interdisciplinaridade,** articulando matemática com **linguagens, história, informática** e **segurança digital;**
- Encorajar o **protagonismo** dos estudantes em **desafios criptográficos** contextualizados.

Diversos **estudos** indicam que o uso de **desafios criptográficos** contribui para o **desenvolvimento** de **competências matemáticas**, atuando como **ferramenta** de mediação **pedagógica** que integra **lógica, criatividade e resolução de problemas.** Ao **desvendar** códigos e criar mensagens **cifradas,** os alunos **experimentam** uma **vivência** matemática **autêntica**, **ativa** e **instigante,** transformando-se em agentes do próprio **conhecimento.**

As **atividades** apresentadas a seguir foram organizadas de forma a **proporcionar** uma abordagem **lúdica** e **prática** da criptografia, combinando experiências no papel, com o uso de **rodas** de **codificação** físicas, e aplicações **computacionais** desenvolvidas na **linguagem C++** por meio do ambiente **Visual Studio.** Cada proposta contempla **objetivos** pedagógicos bem definidos, instruções detalhadas, trechos de código comentados e exemplos de **execução**, promovendo o **raciocínio lógico,** a **aprendizagem significativa** e o **desenvolvimento** do pensamento **computacional.**

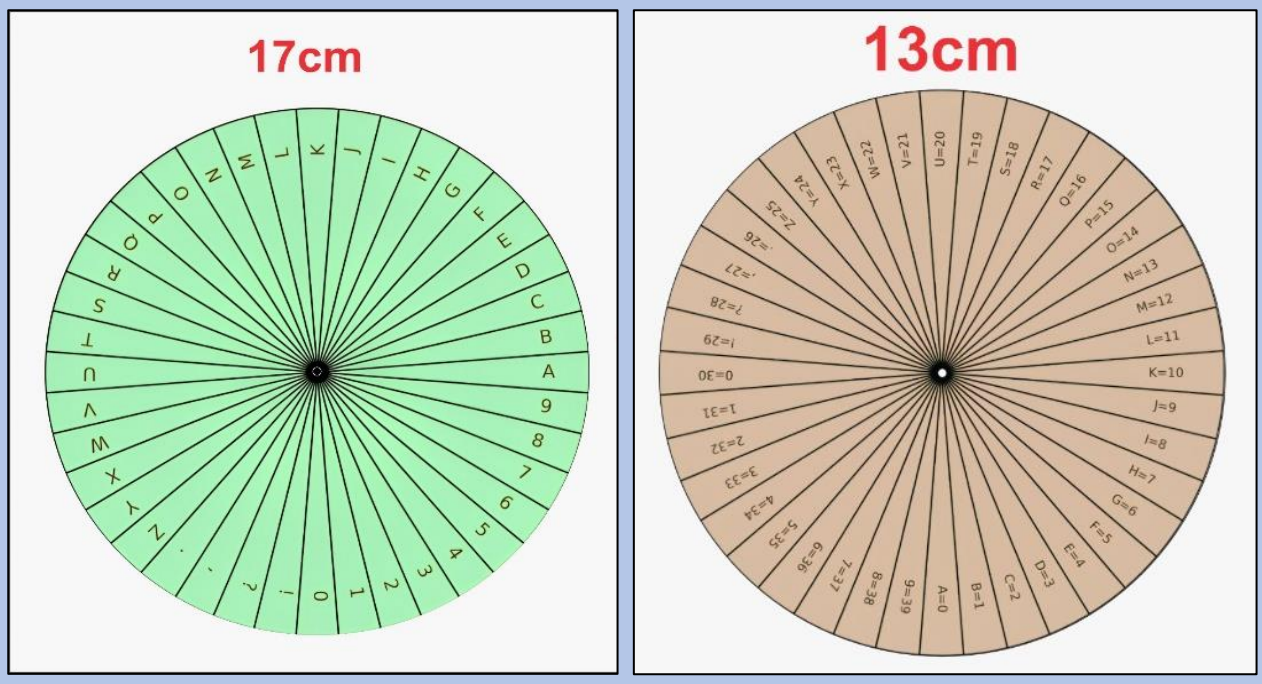
Atividade 1: Códigos Secretos com a Roda de Criptografia

🎯 **Objetivo:** Proporcionar aos alunos uma experiência prática com a Cifra de César, por meio da manipulação de uma roda de criptografia (criptodisco), reforçando conceitos como aritmética modular, congruência, padrões numéricos e lógica criptográfica.

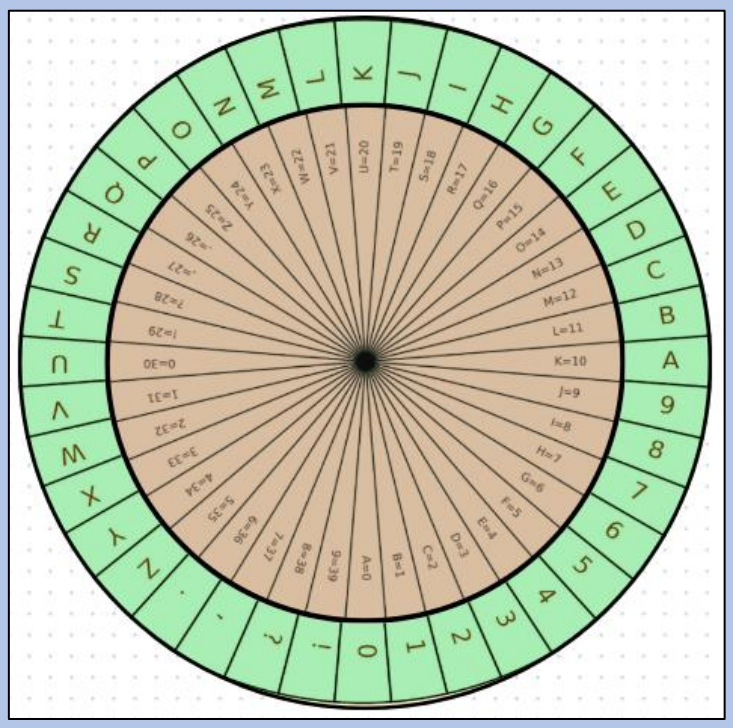
👣 Passo a Passo para o Professor

🛠Etapa 1: Montagem da Roda de Criptografia

- Distribua aos alunos as duas partes do criptodisco (roda externa e roda interna), disponíveis no anexo.
- A ferramenta é composta por 40 caracteres que integram um alfabeto expandido, abrangendo as letras do alfabeto latino (A–Z), os sinais de pontuação (.,?!), e os algarismos numéricos (0–9).



- Peça que recortem cuidadosamente ambas as rodas, respeitando os contornos circulares.
- Oriente para que centralizem a roda interna sobre a externa.



- Fixem as rodas no centro com uma tachinha, grampo ou prendedor, permitindo que a parte interna gire livremente.

🔁 Etapa 2: Configurando o Deslocamento (Chave)

- Explique que o deslocamento define a chave da cifra, variando entre 0 e 39 (alfabeto expandido com letras, números e pontuação).
- Para configurar a cifra, alinhe o caractere “A” da roda externa com o valor da chave na roda interna.
- Se alinhar o caractere “A” da roda externa com “0 = A” na roda interna, o deslocamento será zero, logo não haverá deslocamento.
- Exemplo para Chave 10: Alinhar “A” (externa) com “10” (interna).
- Com esse alinhamento, cada caractere da mensagem original será substituído pelo caractere situado 10 posições adiante, resultando, por exemplo, em A → K.

✉ Etapa 3: Codificando uma Mensagem

- Com a roda ajustada, oriente os alunos a:
- Procurar cada letra da mensagem original na roda externa.
- Substituí-la pela letra correspondente abaixo, na roda interna.

Exemplo prático:

- Mensagem: ESCOLA
- Chave: 5
- Resultado: JXHTQF

🔓 Etapa 4: Decodificando uma Mensagem

- Com a mesma chave usada na codificação:
- Alinhar novamente a roda: “A” da externa com a chave na interna.
- Procurar o caractere da mensagem cifrada na roda interna.

- Substituí-lo pelo correspondente da roda externa.

Exemplo prático:

- Mensagem cifrada: R !R6V!R6ZTR V UZ8V46ZUR
- Chave: 17
- Resultado: A MATEMATICA E DIVERTIDA

Exercícios – Codificação e Decodificação com a Roda de Criptografia

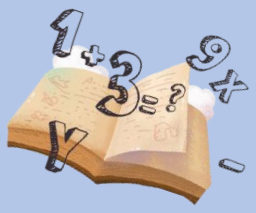
Codificando Mensagens

Utilize a roda de criptografia. Alinhe a letra A da roda externa com o valor da chave na roda interna. Depois, substitua cada caractere da mensagem original pelo correspondente na roda interna.

1. Mensagem: VAMOS ESTUDAR

Chave: 8

Mensagem criptografada: _____



2.Mensagem: CRIPTOGRAFIA É LEGAL!

Chave: 12

Mensagem criptografada: _____

3.Mensagem: HOJE TEM MATEMÁTICA

Chave: 5

Mensagem criptografada: _____

Decodificando Mensagens

Agora, alinhe a roda com a mesma chave usada na codificação e localize os caracteres da mensagem cifrada na roda interna. Substitua

pelo caractere correspondente na roda externa para recuperar a mensagem original.

4. Mensagem cifrada: **WK!OWK!SMK**

Chave: **10**

Mensagem original: _____

5. Mensagem cifrada: **T4Z261X4RWZR**

Chave: **17**

Mensagem original: _____

6. Mensagem cifrada: **W 4WB.4WB0YW ,0Y6C Z0D.9B0ZW**

Chave: **22**

Mensagem original: _____

💡 Desafio Final:

Crie sua própria mensagem, escolha uma chave entre 1 e 39, codifique com o criptodisco e troque com um colega para que ele tente decodificá-la.

🔎 Habilidades Desenvolvidas

A atividade proporciona aos alunos uma vivência concreta dos conceitos de congruência e deslocamentos modulares, permitindo a visualização prática desses conteúdos matemáticos por meio da manipulação de uma ferramenta criptográfica física. Ao utilizar a roda de criptografia, os estudantes compreendem, de forma intuitiva, como um simples deslocamento sistemático pode transformar uma

mensagem clara em um código ilegível, assimilando os fundamentos da criptografia clássica. Além disso, a proposta estimula o raciocínio lógico, a resolução de problemas, e promove uma integração entre Matemática, Linguagem e Tecnologia, favorecendo uma aprendizagem ativa, interdisciplinar e alinhada com as competências da BNCC.

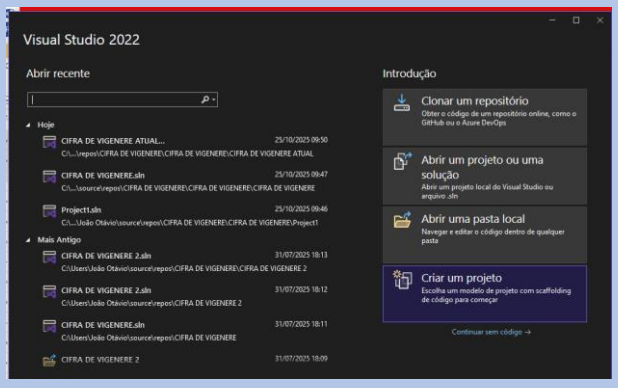
Ferramentas Práticas Aplicadas à Criptografia: Visual Studio

Para possibilitar a aplicação concreta dos algoritmos criptográfico, optou−se pela utilização do **Microsoft Visual Studio** como ambiente de desenvolvimento. Esta escolha deve−se à **segurança da plataforma**, ao seu suporte à **linguagem C++** e aos recursos integrados que favorecem o aprendizado estruturado da **lógica de programação**, aspecto essencial para a compreensão dos mecanismos matemáticos subjacentes à criptografia. Além disso, a possibilidade de **instalar** a **plataforma diretamente no computador** facilita seu uso em ambientes **educacionais** e **domésticos,** proporcionando maior autonomia e **acessibilidade ao usuário**.

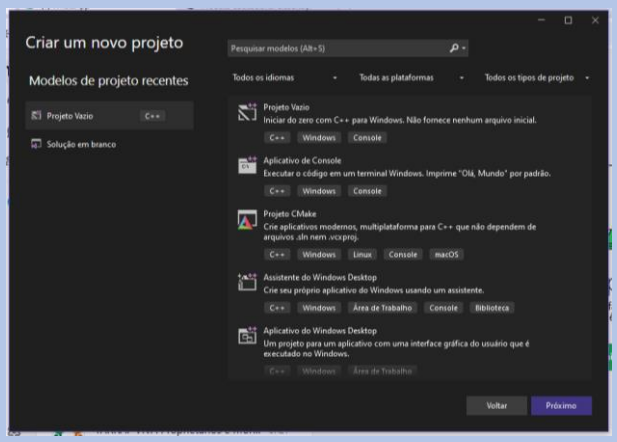
O **Visual Studio** permite a criação de **projetos modulares**, a **simulação** de **entradas** e **saídas** de dados em tempo real. Sua ampla **documentação** e o **apoio** de uma **grande comunidade** de **usuários** tornam a **ferramenta acessível** e **funcional** tanto para **docentes** quanto para **estudantes do Ensino Médio** que estejam ingressando nos estudos da **programação** e da **matemática aplicada.**

👣 Passo a Passo - Executando o código no Visual Studio (C++

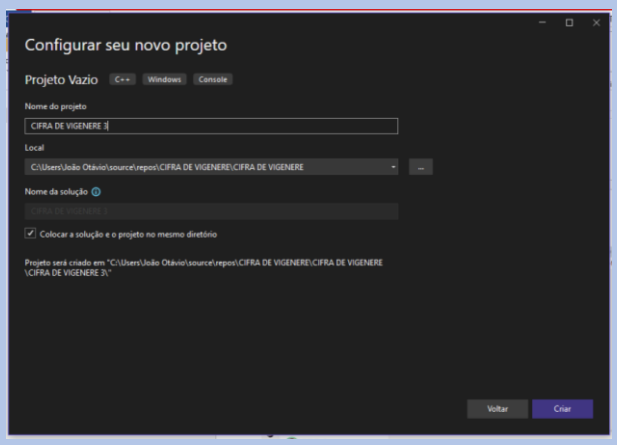
1. Abra o **Visual Studio**.



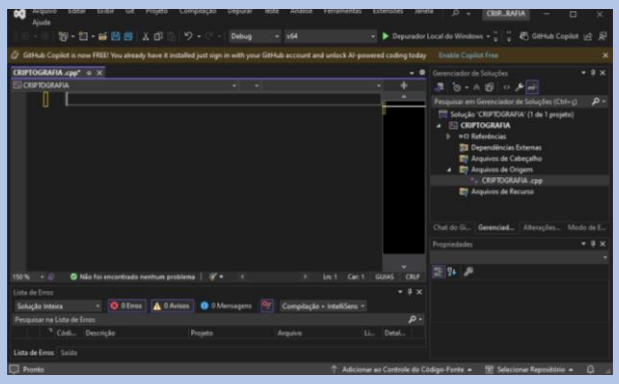
2. Clique em "**Criar um projeto**"
3. **Selecione "Projeto Vazio"** e depois clique em **"Próximo"**.



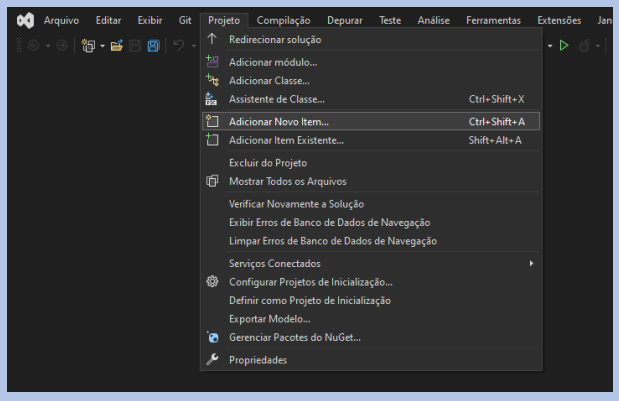
4. **Escolha um nome para o projeto** e clique em **"Criar"**.



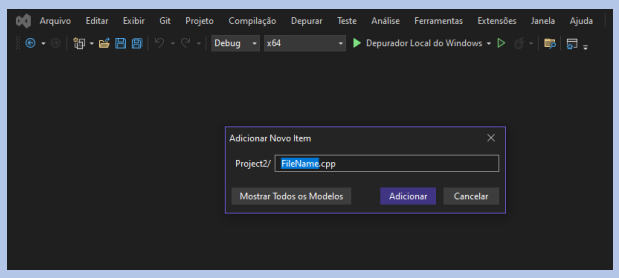
5. Observe a **interface inicial do Visual Studio**, com o projeto em C++ criado.



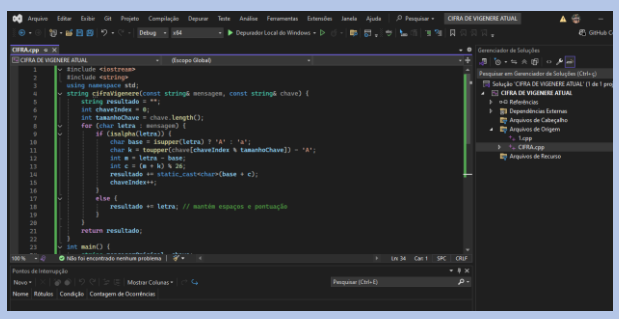
6. No menu superior, clique em "**Projeto → Adicionar Novo Item**", ou use o atalho (**Ctrl + Shift + A**).



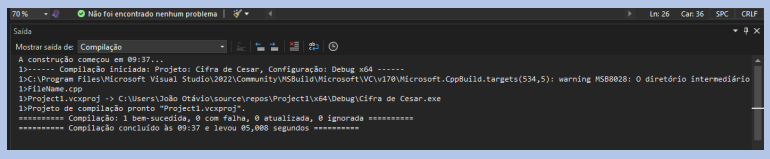
7. **Defina um nome para o novo item** e confirme a criação.



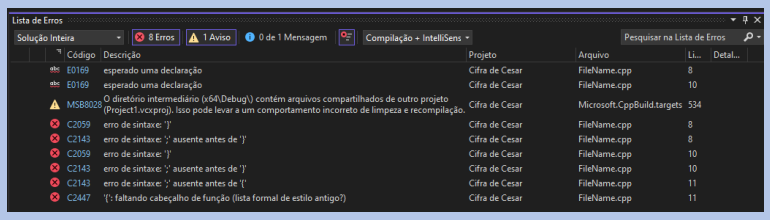
8. **Digite ou cole o código** da atividade apresentada neste material.



9. **Compile o código** e verifique se há erros. Pressione **(Ctrl + Alt + B)**

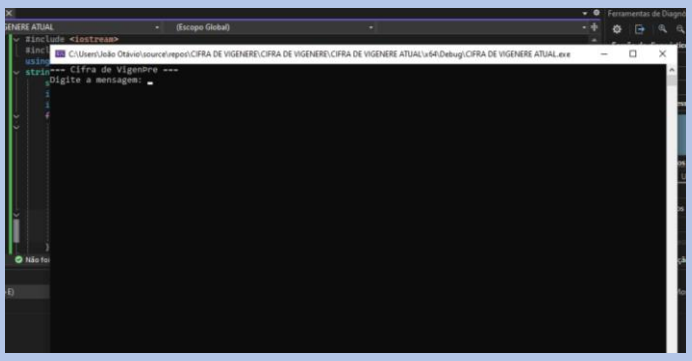


Caso **ocorram erros,** o compilador exibirá **mensagens** específicas na janela de saída, **indicando a linha** e o **tipo de problema** encontrado.

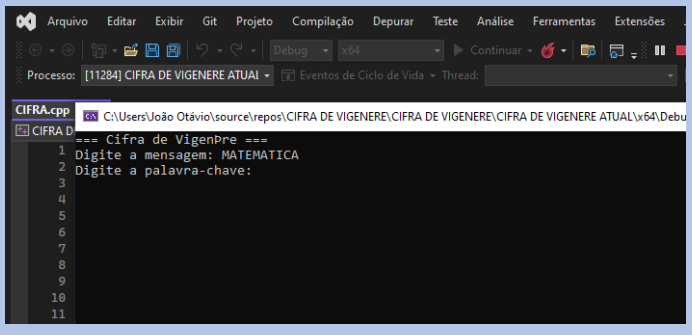


Nessa etapa, é **fundamental** que o **aluno** ou **professor** identifique e **corrija essas falhas** antes de prosseguir com a execução do programa. A presença de **erros impedirá** a **geração do arquivo** executável e, consequentemente, a **realização do teste da cifra.**

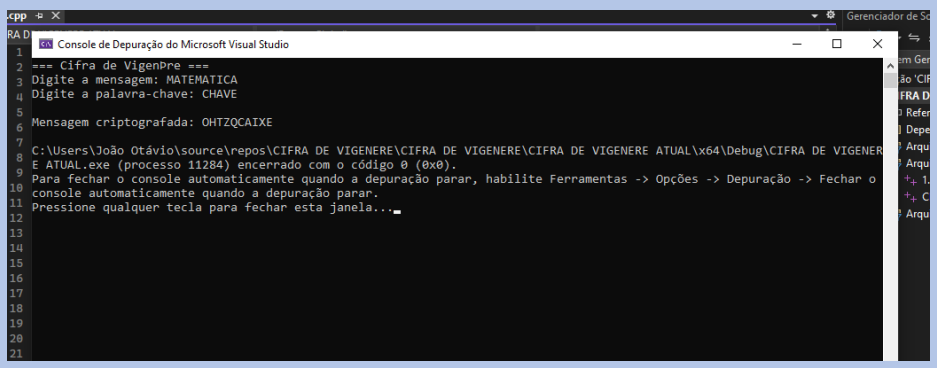
10. **Execute o programa** pressionando **F5**.
11. **Digite a mensagem** desejada e pressione **Enter**.



12. **Digite a chave** e pressione **Enter** novamente.



13. O console exibirá a **mensagem cifrada**.



A escolha do **Visual Studio** se **justifica** também pela sua **compatibilidade** com **projetos educacionais**, permitindo a **criação de aplicativos** simples e **personalizados**, nos quais os estudantes podem **manipular mensagens** criptografadas, **explorar cifras** clássicas como **César, Vigenère e Multiplicativa,** e compreender sua **lógica algorítmica** por meio da **implementação prática.**

No **anexo do arquivo** encontra-se uma tabela que apresenta a tradução e a explicação didática desses componentes, com o objetivo de ampliar a acessibilidade técnica do conteúdo e favorecer sua aplicação em contextos educacionais.

Atividade 2 – Desvendando Mensagens com a Cifra de César (Programação em C++)

🎯 **Objetivo da Atividade:** Explorar o conceito de aritmética modular por meio da implementação da Cifra de César em linguagem C++, permitindo que os alunos compreendam a lógica dos deslocamentos em sistemas de codificação clássicos. A atividade também visa desenvolver raciocínio lógico, leitura de algoritmos e noções básicas de programação.

👣 **Passo a Passo para o Professor**

Etapa 1 – Apresentação do Conceito

- Explique que a Cifra de César é uma técnica de criptografia em que cada letra da mensagem é deslocada um número fixo de posições no alfabeto.

Apresente a fórmula matemática usada:

$$c \equiv m + k \pmod{26}$$

onde:

- c é o valor da letra criptografada,
- m é o valor numérico da letra original (0 a 25),
- k é a chave de deslocamento (inteiro positivo),
- a operação é feita módulo 26 (número de letras do alfabeto latino).

Etapa 2 – Preparação do Ambiente

- Peça aos alunos que abram o Visual Studio no computador.
- Oriente-os a criar um novo projeto C++ em modo console.

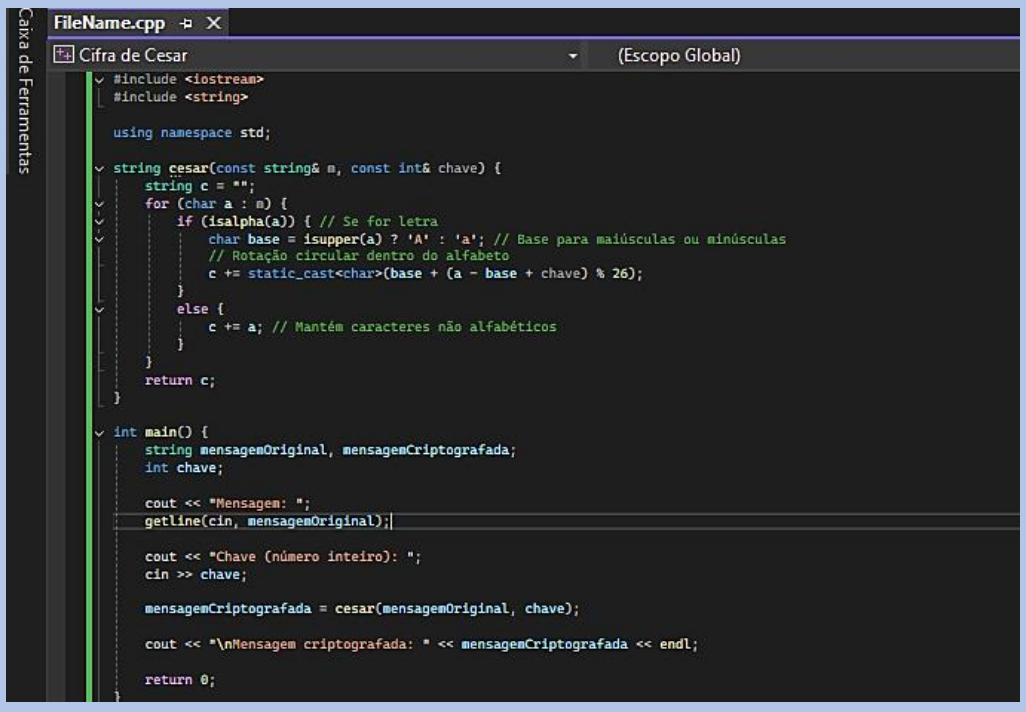
Etapa 3 – Digitação do Código

- Forneça o código abaixo e peça que os alunos copiem exatamente:

```
#include <iostream>
#include <string>
using namespace std;
string cesar(const string& m, const int& chave) {
   string c = "";
   for (char a : m) {
      if (isalpha(a)) { // Se for letra
         char base = isupper(a) ? 'A' : 'a'; // Base para maiúsculas ou minúsculas
         // Rotação circular dentro do alfabeto
         c += static_cast<char>(base + (a - base + chave) % 26);
      }
      else {
```

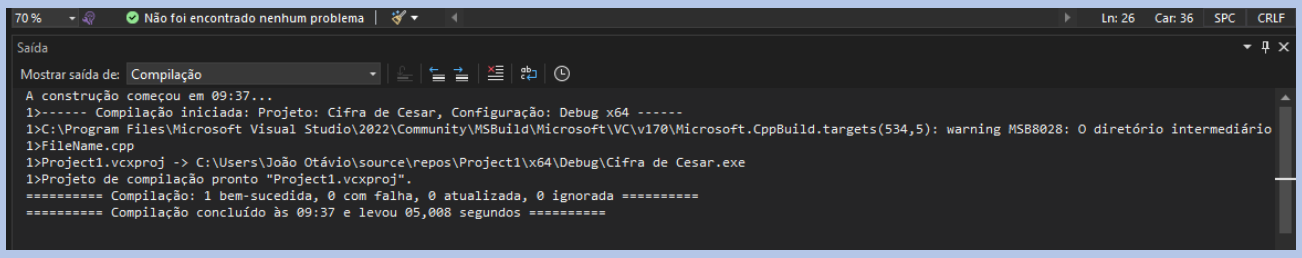
```
            c += a; // Mantém caracteres não alfabéticos
        }
    }
    return c;
}
int main() {
    string mensagemOriginal, mensagemCriptografada;
    int chave;
    cout << "Mensagem: ";
    getline(cin, mensagemOriginal);
    cout << "Chave (número inteiro): ";
    cin >> chave;
    mensagemCriptografada = cesar(mensagemOriginal, chave);
    cout << "\nMensagem criptografada: " << mensagemCriptografada << endl;
    return 0;
            }
```

- Esse código é ideal para uso didático, pois permite que os alunos visualizem diretamente como a fórmula matemática do deslocamento é aplicada a cada caractere da string, reforçando o conceito de aritmética modular. A estrutura do código também estimula a leitura lógica dos comandos e o desenvolvimento de habilidades básicas de programação.



Etapa 4 – Compilação do Código

- Oriente os alunos a compilar o programa pressionando Ctrl + Alt + B.

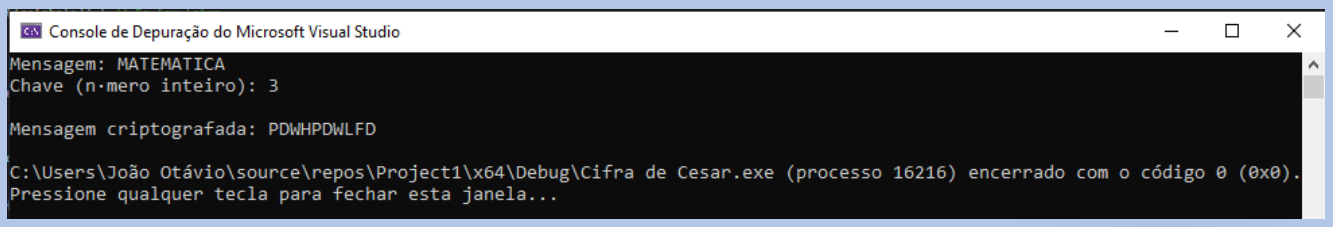


Etapa 5 – Testes e Análise

- Solicite que testem diferentes mensagens e valores de chave. Exemplos:

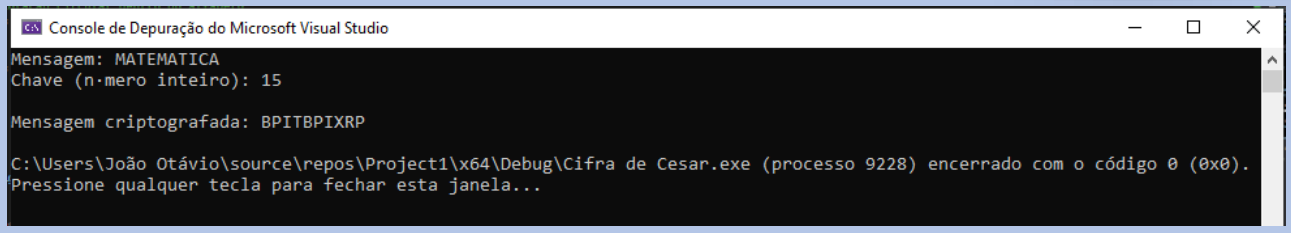
Exemplo 1

- Mensagem: MATEMATICA
- Chave: 3
- Saída: PDWHPDWLFD



Exemplo 2

- Mensagem: MATEMATICA
- Chave: 15
- Saída: BPITBPIXRP



- Peça que comparem os resultados e reflitam sobre como alterações na chave mudam totalmente a mensagem cifrada.
- A comparação entre os dois exemplos mostra como pequenas variações na chave podem alterar completamente a mensagem cifrada. Esse é um

dos princípios centrais da segurança na criptografia por chave simétrica.

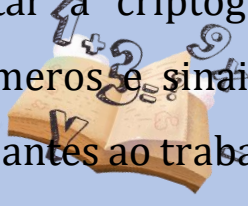
- Proponha que os alunos testem com outras palavras-chave e observem as diferenças na codificação

🔎 Habilidades Desenvolvidas

Essa atividade promove não apenas o aprendizado da aritmética modular de forma aplicada, mas também desenvolve competências importantes em lógica computacional e resolução de problemas. A associação entre conceitos matemáticos e sua aplicação em um programa real facilita a compreensão dos estudantes, além de incentivar a interdisciplinaridade entre matemática, história e tecnologia. Adicionalmente, o uso do exemplo clássico de Júlio César (com $k = 3$) fornece uma referência histórica de fácil assimilação pelos alunos, contribuindo para o aspecto lúdico e contextualizado da proposta.

Atividade 3 – Expandindo o Código com a Criptografia Aditiva

🎯 **Objetivo da Atividade:** Explorar a aritmética modular aplicada a alfabetos expandidos, demonstrando como adaptar a criptografia aditiva para mensagens compostas por letras, números e sinais de pontuação. A atividade amplia o repertório dos estudantes ao trabalhar com conjuntos personalizados de caracteres, aproximando a matemática de contextos reais como segurança digital e comunicação codificada.



👣 Passo a Passo para o Professor

Etapa 1 – Apresentação do Conceito

- Explique aos alunos que a criptografia aditiva consiste em deslocar cada caractere da mensagem por uma chave numérica fixa, utilizando um alfabeto definido.
- Diferente da cifra clássica (apenas letras A–Z), nesta atividade será utilizado um alfabeto expandido com 44 caracteres:
- ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789 .,:!?
- Apresente a fórmula matemática:

$$c \equiv m + k \pmod{n}$$

onde:

- c representa o caractere criptografado,
- m é a posição do caractere original no alfabeto expandido,
- k é a chave de deslocamento,
- n é o tamanho total do alfabeto utilizado.

Etapa 2 – Preparação do Ambiente

- Oriente os alunos a abrir o Visual Studio e criar um novo projeto C++ console.
- Solicite que nomeiem o projeto como CifraAditivaExpandida.

Etapa 3 – Digitação do Código

- Forneça o código a seguir para que os alunos digitem com atenção:

```
#include <iostream>
#include <string>
using namespace std;
string ALFABETO = "ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789 .,:!?";
int n = ALFABETO.length();
```

```
string cifraAditivaExpandida(const string& mensagem, int chave) {
    string resultado = "";
    for (char letra : mensagem) {
        letra = toupper(letra); // converte tudo para maiúsculo
        size_t pos = ALFABETO.find(letra);
        if (pos != string::npos) {
            int c = (pos + chave) % n;
            resultado += ALFABETO[c];
        }
        else {
            resultado += letra; // mantém acentos, emojis, etc.
        }
    }
    return resultado;
}
int main() {
    string mensagemOriginal;
    int chave;
    cout << "=== Cifra Aditiva com Alfabeto Expandido ===" << endl;
    cout << "Digite a mensagem: ";
    getline(cin, mensagemOriginal);
    cout << "Digite a chave (inteiro): ";
    cin >> chave;
    string criptografada = cifraAditivaExpandida(mensagemOriginal, chave);
    cout << "\nMensagem criptografada: " << criptografada << endl;
    return 0;
}
```

Etapa 4 – Visualização, Compilação e Execução

- Instrua os alunos a compilar o código com Ctrl + Alt + B e corrigir possíveis erros indicados pelo compilador.
- A estrutura do código também estimula a leitura lógica dos comandos e o desenvolvimento de habilidades básicas de programação

CODIGO2 (Escopo Global)

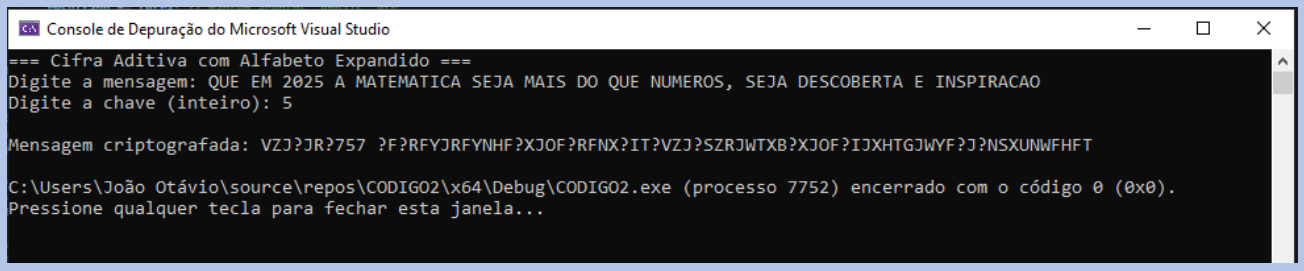
```cpp
#include <iostream>
#include <string>
using namespace std;

string ALFABETO = "ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789 .,:!?";
int n = ALFABETO.length();

string cifraAditivaExpandida(const string& mensagem, int chave) {
    string resultado = "";

    for (char letra : mensagem) {
        letra = toupper(letra); // converte tudo para maiúsculo
        size_t pos = ALFABETO.find(letra);

        if (pos != string::npos) {
            int c = (pos + chave) % n;
            resultado += ALFABETO[c];
        }
        else {
            resultado += letra; // mantém acentos, emojis, etc.
        }
    }

    return resultado;
}

int main() {
    string mensagemOriginal;
    int chave;

    cout << "=== Cifra Aditiva com Alfabeto Expandido ===" << endl;
    cout << "Digite a mensagem: ";
    getline(cin, mensagemOriginal);

    cout << "Digite a chave (inteiro): ";
    cin >> chave;

    string criptografada = cifraAditivaExpandida(mensagemOriginal, chave);

    cout << "\nMensagem criptografada: " << criptografada << endl;

    return 0;
}
```

- Após a compilação bem-sucedida, peça que executem o programa com Ctrl + F5.

Etapa 5 – Testes e Reflexão

- A execução prática da cifra com alfabeto expandido foi realizada com a seguinte configuração:

- Mensagem original: QUE EM 2025 A MATEMATICA SEJA MAIS DO QUE NUMEROS, SEJA DESCOBERTA E INSPIRACAO
- Chave utilizada: 5
- Mensagem criptografada: VZJ?JR?757??F?RFYJRFYNHF?XJOF ?RFNX?IT?VZJ?SZRJWTXB?XJOF?IJXHTGJWYF?J?NSXUNWFHFT



- Incentive os alunos a testar diferentes chaves e a analisar como pequenas variações no deslocamento resultam em mudanças significativas na mensagem final.

🔎 Habilidades Desenvolvidas

Este exemplo ilustra como o uso de alfabetos expandidos potencializa o ensino da aritmética modular, permitindo aos alunos visualizar a generalização da congruência para diferentes conjuntos de símbolos. Ao incluir letras, números e sinais de pontuação, a atividade aproxima os conceitos matemáticos do cotidiano digital, evidenciando aplicações práticas em áreas como segurança da informação, criptografia e linguagens formais.

Além do aprofundamento conceitual, a proposta promove o desenvolvimento de competências essenciais à formação contemporânea, como raciocínio lógico, abstração e pensamento algorítmico. Ao integrar matemática com elementos de informática, linguagem e cidadania digital, favorece uma aprendizagem ativa e significativa, em sintonia com os desafios educacionais da cultura digital

Atividade 4 – Função Afim em Ação: Codificando com Critério de MDC

🎯 **Objetivo da Atividade:** Apresentar a cifra afim como extensão da aritmética modular, integrando multiplicação, adição, congruência e o cálculo do máximo divisor comum (MDC). A atividade visa desenvolver a compreensão sobre inversos modulares e sua importância na segurança criptográfica, por meio da programação estruturada em linguagem C++.

👣 Passo a Passo para o Professor

Etapa 1 – Introdução Conceitual

- Explique que a cifra afim é uma generalização da cifra aditiva, combinando multiplicação e adição sobre o alfabeto (A–Z).

Apresente a fórmula utilizada:

$$c \equiv a \cdot m + b \pmod{26}$$

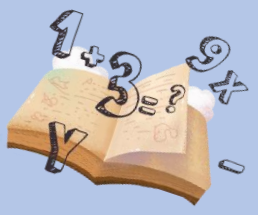
onde:

- c é o valor da letra criptografada,
- m é a posição da letra original no alfabeto (0 a 25),
- a é o fator multiplicativo,
- b é o deslocamento aditivo,
- a e 26 devem ser primos entre si, ou seja, mdc(a , 26) = 1.

- Destaque que a deve ser coprimo de 26 (ou seja, mdc(a, 26) = 1), para que a função seja invertível, condição essencial para decodificar a mensagem depois.

Etapa 2 – Ambiente de Programação

- Oriente os alunos a abrir o Visual Studio e criar um novo projeto C++ em modo console.
- Peça que nomeiem o projeto como CifraAfimMDC.

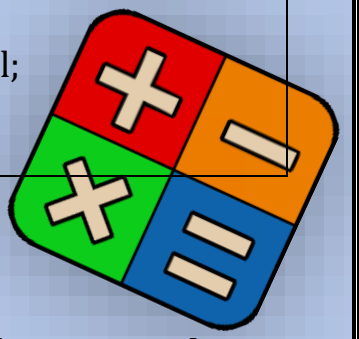


Etapa 3 – Digitação do Código

- Apresente o código abaixo para digitação e análise:

```
#include <iostream>
#include <string>
using namespace std;
// Função para calcular o máximo divisor comum (mdc)
int mdc(int a, int b) {
   while (b != 0) {
      int temp = b;
      b = a % b;
      a = temp;
   }
   return a;
}
// Função de criptografia afim
string cifraAfim(const string& mensagem, int a, int b) {
   string resultado = "";
   int n = 26;
      if (mdc(a, n) != 1) {
      return "Erro: o valor de 'a' deve ser primo com 26 (mdc(a, 26) = 1).";
   }
   for (char letra : mensagem) {
      if (isalpha(letra)) {
         char base = isupper(letra) ? 'A' : 'a';
         int m = letra - base;
         int c = (a * m + b) % n;
         resultado += static_cast<char>(base + c);
```

```
        }
        else {
            resultado += letra;
        }
    }
        return resultado;
}
int main() {
    string mensagemOriginal;
    int a, b;
        cout << "=== Cifra Afim: c = (a * m + b) mod 26 ===" << endl;
    cout << "Digite a mensagem original: ";
    getline(cin, mensagemOriginal);
        cout << "Digite o valor de 'a' (deve ser primo com 26): ";
    cin >> a;
        cout << "Digite o valor de 'b' (deslocamento): ";
    cin >> b;
        string resultado = cifraAfim(mensagemOriginal, a, b);
    cout << "\nMensagem criptografada: " << resultado << endl;
        return 0;
}
```



Etapa 4 – Visualização, Compilação e Execução

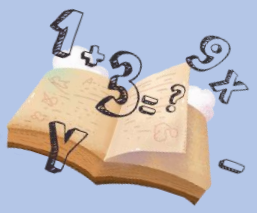
- A estrutura do código também estimula a leitura lógica dos comandos e o desenvolvimento de habilidades básicas de programação

CODIGO 3 (Escopo Global)

```cpp
#include <iostream>
#include <string>
using namespace std;
// Função para calcular o máximo divisor comum (mdc)
int mdc(int a, int b) {
    while (b != 0) {
        int temp = b;
        b = a % b;
        a = temp;
    }
    return a;
}
// Função de criptografia afim
string cifraAfim(const string& mensagem, int a, int b) {
    string resultado = "";
    int n = 26;
        if (mdc(a, n) != 1) {
        return "Erro: o valor de 'a' deve ser primo com 26 (mdc(a, 26) = 1).";
    }
    for (char letra : mensagem) {
        if (isalpha(letra)) {
            char base = isupper(letra) ? 'A' : 'a';
            int m = letra - base;
            int c = (a * m + b) % n;
            resultado += static_cast<char>(base + c);
        }
        else {
            resultado += letra;
        }
    }
        return resultado;
}
int main() {
    string mensagemOriginal;
    int a, b;
        cout << "=== Cifra Afim: c = (a * m + b) mod 26 ===" << endl;
    cout << "Digite a mensagem original: ";
    getline(cin, mensagemOriginal);
        cout << "Digite o valor de 'a' (deve ser primo com 26): ";
    cin >> a;
        cout << "Digite o valor de 'b' (deslocamento): ";
    cin >> b;
        string resultado = cifraAfim(mensagemOriginal, a, b);
    cout << "\nMensagem criptografada: " << resultado << endl;
        return 0;
}
```

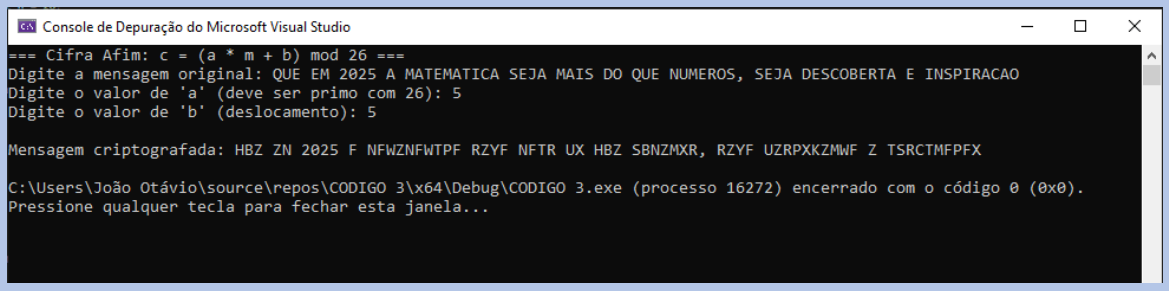
- Oriente os alunos a compilar o programa usando Ctrl + Alt + B e revisar atentamente os erros, se houver.
- Após compilação bem-sucedida, execute com Ctrl + F5.

Etapa 5 – Testes e Aplicações



- Solicite que testem com os seguintes dados:

- Mensagem original: **QUE EM 2025 A MATEMATICA SEJA MAIS DO QUE NUMEROS, SEJA DESCOBERTA E INSPIRACAO**
- Chave utilizada: **a = 5, b = 5**
- Mensagem criptografada: **HBZ ZN 2025 F NFWZFNFWTPF RZYF NFTR UX HBZ SBNZMXR, RZYF UZRPXKZMWF Z TSRCTMFPFX**



- Estimule a troca de mensagens entre os alunos com diferentes pares (a, b), verificando se o valor de a é válido (coprimo com 26).
- Proponha que os alunos testem com outras palavras-chave e observem as diferenças na codificação

🔎 Habilidades Desenvolvidas

Este exemplo destaca a relevância de verificar previamente as condições matemáticas necessárias à aplicação segura de algoritmos criptográficos. A utilização do máximo divisor comum (MDC) como critério de validação introduz, de maneira aplicada, o conceito de primos relativos e abre espaço para a discussão sobre a existência de inversos modulares e sua importância na codificação.

A atividade também promove uma integração efetiva entre os conhecimentos matemáticos e computacionais, pois exige dos alunos não apenas a implementação do algoritmo, mas a compreensão dos princípios teóricos que sustentam sua lógica. Essa abordagem estimula o desenvolvimento de competências como resolução de problemas, pensamento computacional e análise crítica, sendo especialmente indicada em propostas pedagógicas voltadas à alfabetização científica e ao uso significativo da tecnologia

Atividade 5 – Codificando com Multiplicação

🎯 **Objetivo da Atividade:** Aplicar, de forma prática, conceitos fundamentais da teoria dos números, como congruência modular, coprimalidade e inverso multiplicativo, por meio da cifra multiplicativa com alfabeto expandido. A proposta visa mostrar como esses princípios matemáticos sustentam a segurança da informação, ampliando a compreensão dos alunos sobre os critérios matemáticos exigidos para a reversibilidade de sistemas criptográficos.

👣 Passo a Passo para o Professor

Etapa 1 – Introdução Conceitual

Apresente a cifra multiplicativa, que transforma uma mensagem utilizando apenas a operação de multiplicação modular.

Explique a fórmula utilizada:

$$c \equiv a \cdot m \pmod{n}$$

onde:

- c representa o caractere cifrado,
- m é a posição do caractere original no alfabeto expandido,
- a é a chave multiplicativa,
- n é o tamanho do alfabeto expandido.

Destaque a condição essencial: A condição essencial é que a e n sejam coprimos, ou seja, mdc (a, n) = 1, para garantir que a cifra possa ser

invertida. Esse critério reforça o uso consciente e seguro dos algoritmos criptográficos.

Etapa 2 – Preparação do Ambiente

- Peça aos alunos que abram o Visual Studio e criem um novo projeto console C++.
- Sugira o nome do projeto: CifraMultiplicativaExpandida.

Etapa 3 – Digitação do Código

- Forneça o código a seguir e oriente os alunos para digitação cuidadosa:

```
#include <iostream>
#include <string>
using namespace std;
// Alfabeto expandido: letras, números e pontuação
string ALFABETO = "ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789 .,:!?";
int n = ALFABETO.length(); // Tamanho do alfabeto expandido
// Função para calcular o máximo divisor comum (mdc)
int mdc(int a, int b) {
   while (b != 0) {
      int t = b;
      b = a % b;
      a = t;
   }
   return a;
}
// Função da cifra multiplicativa com alfabeto expandido
string cifraMultiplicativaExpandida(const string& mensagem, int a) {
   string resultado = "";
   if (mdc(a, n) != 1) {
      return "Erro: o valor de 'a' deve ser primo com " + to_string(n) + " (mdc(a,
n) = 1).";
   }
   for (char letra : mensagem) {
      letra = toupper(letra);
      size_t pos = ALFABETO.find(letra);

      if (pos != string::npos) {
```

```
            int c = (a * pos) % n;
            resultado += ALFABETO[c];
        }
        else {
            resultado += letra; // mantém acentos, emojis, etc.
        }
    }
    return resultado;
}
int main() {
    string mensagemOriginal;
    int a;
    cout << "=== CIFRA MULTIPLICATIVA COM ALFABETO EXPANDIDO ===" 
<< endl;
    cout << "Alfabeto: " << ALFABETO << endl;
    cout << "Digite a mensagem: ";
    getline(cin, mensagemOriginal);
    cout << "Digite a chave 'a' (deve ser primo com " << n << "): ";
    cin >> a;
    string resultado = cifraMultiplicativaExpandida(mensagemOriginal, a);
    cout << "\nMensagem criptografada: " << resultado << endl;
    return 0;
            }
```

Etapa 4 – Visualização, Compilação e Execução

- A estrutura do código também estimula a leitura lógica dos comandos e o desenvolvimento de habilidades básicas de programação

codigo 4 (Escopo Global)

```cpp
#include <iostream>
#include <string>
using namespace std;

// Alfabeto expandido: letras, números e pontuação
string ALFABETO = "ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789 .,:!?";
int n = ALFABETO.length(); // Tamanho do alfabeto expandido

// Função para calcular o máximo divisor comum (mdc)
int mdc(int a, int b) {
    while (b != 0) {
        int t = b;
        b = a % b;
        a = t;
    }
    return a;
}

// Função da cifra multiplicativa com alfabeto expandido
string cifraMultiplicativaExpandida(const string& mensagem, int a) {
    string resultado = "";

    if (mdc(a, n) != 1) {
        return "Erro: o valor de 'a' deve ser primo com " + to_string(n) + " (mdc(a, n) = 1).";
    }

    for (char letra : mensagem) {
        letra = toupper(letra);
        size_t pos = ALFABETO.find(letra);

        if (pos != string::npos) {
            int c = (a * pos) % n;
            resultado += ALFABETO[c];
        }
        else {
            resultado += letra; // mantém acentos, emojis, etc.
        }
    }

    return resultado;
}

int main() {
    string mensagemOriginal;
    int a;

    cout << "=== CIFRA MULTIPLICATIVA COM ALFABETO EXPANDIDO ===" << endl;
    cout << "Alfabeto: " << ALFABETO << endl;
    cout << "Digite a mensagem: ";
    getline(cin, mensagemOriginal);

    cout << "Digite a chave 'a' (deve ser primo com " << n << "): ";
    cin >> a;

    string resultado = cifraMultiplicativaExpandida(mensagemOriginal, a);

    cout << "\nMensagem criptografada: " << resultado << endl;

    return 0;
}
```

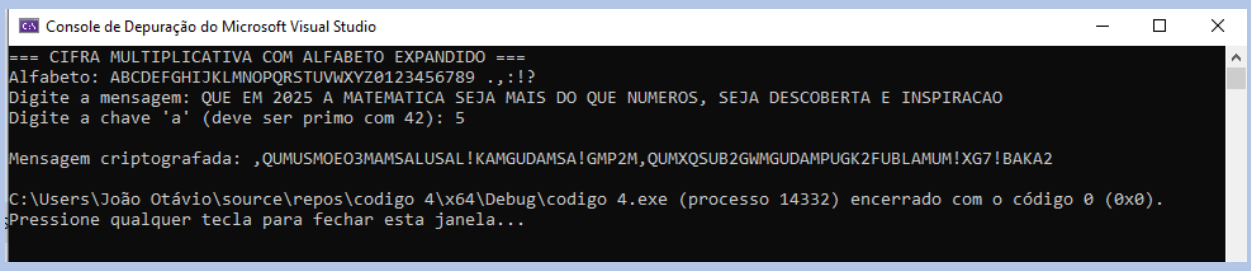
- Solicite que os alunos compilem o código com Ctrl + Alt + B.
- Caso ocorram erros, utilize-os como oportunidade para discutir diagnóstico e correção de sintaxe.
- Após correção, executem o programa com Ctrl + F5.

Etapa 5 – Testes e Exploração

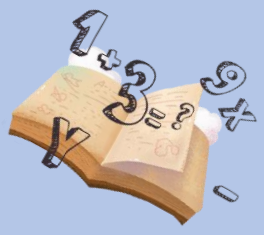
- Oriente os alunos a usar os dados do exemplo:

- Mensagem original: **QUE EM 2025 A MATEMATICA SEJA MAIS DO QUE NUMEROS, SEJA DESCOBERTA E INSPIRACAO**
- Chave utilizada: **a = 5**

- Mensagem criptografada: ,QUMUSMOEO3MAMSALUSAL!KAMGUDAM
- SA!GMP2M,QUMXQSUB2GWMGUDAMPUGK2FUBLAMUM!XG7!BAKA2



- Incentive os alunos a testar outros valores de a e observar os efeitos nas mensagens. Reforce a verificação da condição mdc(a, 44) = 1 antes da execução.



🔎 Habilidades Desenvolvidas

A cifra multiplicativa com alfabeto expandido permite aplicar, de forma prática, conceitos como inverso modular, coprimalidade e congruência, evidenciando a estrutura dos grupos multiplicativos. Ao incluir letras, números e pontuação, aproxima o conteúdo matemático das aplicações reais em segurança digital.

Além disso, a atividade estimula a integração entre matemática e programação, desenvolvendo o raciocínio algorítmico e a análise crítica dos parâmetros. Ao experimentar diferentes valores de chave, os estudantes compreendem o impacto direto dessas variações na decodificação, fortalecendo sua intuição e compreensão dos sistemas criptográficos.

Atividade 6 – Palavras como Chave na Cifra de Vigenère

🎯 **Objetivo da Atividade:** Apresentar a cifra de Vigenère como exemplo clássico de criptografia polialfabética, reforçando os conceitos de modularidade, deslocamentos múltiplos e repetição cíclica. A atividade permite que os alunos compreendam como uma palavra-chave alfabética modifica sistematicamente diferentes partes de uma mensagem, desenvolvendo habilidades em matemática, programação e raciocínio lógico.

👣 **Passo a Passo para o Professor**

Etapa 1 – Introdução ao Conceito

- Explique que a cifra de Vigenère é uma evolução da cifra de César.
- Mostre que, em vez de um único valor de deslocamento, ela utiliza uma palavra-chave, em que cada letra define um deslocamento diferente.

Apresente a fórmula utilizada:

$$ci \equiv mi + ki \pmod{26}$$

onde:

- ci é o caractere cifrado na posição i,
- mi é o valor da letra original na posição i,
- ki é o valor da letra correspondente da chave,
- a operação é feita módulo 26 (alfabeto com 26 letras).

Etapa 2 – Preparação no Visual Studio

- Oriente os alunos a criar um novo projeto em C++ no Visual Studio.
- Sugira que nomeiem o projeto como CifraVigenere.

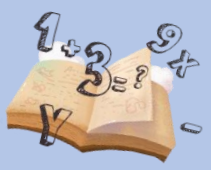
Etapa 3 – Digitação do Código

- Peça que copiem e compreendam o seguinte código:

```
#include <iostream>
#include <string>
using namespace std;
string cifraVigenere(const string& mensagem, const string& chave) {
    string resultado = "";
    int chaveIndex = 0;
    int tamanhoChave = chave.length();
    for (char letra : mensagem) {
        if (isalpha(letra)) {
            char base = isupper(letra) ? 'A' : 'a';
            char k = toupper(chave[chaveIndex % tamanhoChave]) - 'A';
            int m = letra - base;
            int c = (m + k) % 26;
            resultado += static_cast<char>(base + c);
            chaveIndex++;
        }
        else {
            resultado += letra; // mantém espaços e pontuação
        }
    }
    return resultado;
}
int main() {
    string mensagemOriginal, chave;
    cout << "=== Cifra de Vigenère ===" << endl;
    cout << "Digite a mensagem: ";
    getline(cin, mensagemOriginal);
    cout << "Digite a palavra-chave: ";
    cin >> chave;
    string criptografada = cifraVigenere(mensagemOriginal, chave);
    cout << "\nMensagem criptografada: " << criptografada << endl;
    return 0;
}
```

Etapa 4 – Visualização, Compilação e Execução

- A estrutura do código também estimula a leitura lógica dos comandos e o desenvolvimento de habilidades básicas de programação



codigo5.cpp

```cpp
#include <iostream>
#include <string>
using namespace std;

string cifraVigenere(const string& mensagem, const string& chave) {
    string resultado = "";
    int chaveIndex = 0;
    int tamanhoChave = chave.length();

    for (char letra : mensagem) {
        if (isalpha(letra)) {
            char base = isupper(letra) ? 'A' : 'a';
            char k = toupper(chave[chaveIndex % tamanhoChave]) - 'A';
            int m = letra - base;
            int c = (m + k) % 26;
            resultado += static_cast<char>(base + c);
            chaveIndex++;
        }
        else {
            resultado += letra; // mantém espaços e pontuação
        }
    }

    return resultado;
}

int main() {
    string mensagemOriginal, chave;

    cout << "=== Cifra de Vigenère ===" << endl;
    cout << "Digite a mensagem: ";
    getline(cin, mensagemOriginal);

    cout << "Digite a palavra-chave: ";
    cin >> chave;

    string criptografada = cifraVigenere(mensagemOriginal, chave);

    cout << "\nMensagem criptografada: " << criptografada << endl;

    return 0;
}
```

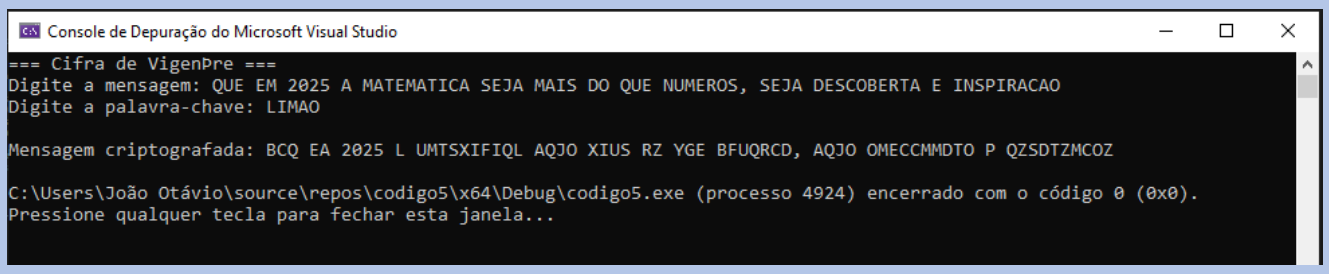
- Oriente a compilação do código com o atalho Ctrl + Alt + B.
- Execute com Ctrl + F5 e garanta que todos os testes sejam realizados com letras maiúsculas e sem acento.

Etapa 5 – Teste e Análise

- A cifra foi aplicada com a palavra-chave "LIMAO", resultando na seguinte transformação criptográfica:

- Mensagem original: QUE EM 2025 A MATEMATICA SEJA MAIS DO QUE NUMEROS, SEJA DESCOBERTA E INSPIRACAO
- Palavra-chave utilizada: LIMAO

- Mensagem criptografada: **BCQ EA 2025 L UMTSXIFIQL AQJO XIUS RZ YGE BFUQRCD, AQJO OMECCMMDTO P QZSDTZMCOZ**



- Proponha que os alunos testem com outras palavras-chave e observem as diferenças na codificação.

🔑 Habilidades Desenvolvidas

A cifra de Vigenère introduz o conceito de múltiplos deslocamentos controlados por uma chave cíclica, ampliando a compreensão dos alunos sobre sistemas criptográficos. Sua aplicação envolve operações modulares e manipulação de strings, contribuindo para o desenvolvimento do pensamento computacional. A atividade também favorece a interdisciplinaridade ao dialogar com conteúdos de Língua Portuguesa, História e Informática. Ao lidar com entradas livres e criptografia polialfabética, os estudantes enfrentam desafios reais da segurança digital, refletindo sobre estratégias de proteção e exposição de informações.

Considerações Finais

A elaboração deste manual didático teve como propósito aproximar a **criptografia** da realidade escolar, apresentando-a não apenas como técnica de **segurança digital**, mas também como recurso pedagógico capaz de despertar a **curiosidade** e promover a

aprendizagem significativa em **Matemática**. A contextualização histórica, desde a **scytale espartana** e a **Cifra de César** até a **Cifra de Vigenère** e os algoritmos modernos como **RSA** e **AES**, evidenciou que a criptografia sempre caminhou junto ao avanço matemático, constituindo-se em um campo fértil para a exploração didática.

As **propostas** reunidas foram pensadas para **atender** diferentes **níveis de ensino** e podem ser **adaptadas** conforme a maturidade e os conhecimentos prévios dos estudantes. Ao introduzir conceitos básicos de **criptografia**, desenvolver operações de **aritmética modular**, estimular a **resolução de problemas** e a **análise de padrões**, busca-se proporcionar experiências em que os alunos percebam a **Matemática** como disciplina viva, útil e conectada a temas contemporâneos. Além disso, o trabalho **colaborativo** e a **interdisciplinaridade** reforçam o protagonismo dos estudantes, tornando-os agentes ativos na construção do conhecimento.

As atividades apresentadas combinaram momentos **lúdicos** e **práticos**, desde **exercícios em papel** e **rodas de codificação físicas** até a implementação computacional em **C++** no ambiente **Visual Studio**. Essa **diversidade** de **recursos** permitiu tanto a **valorização** da **manipulação** concreta, essencial para os anos finais do **Ensino Fundamental**, quanto a exploração de **ambientes digitais**, mais apropriada para o **Ensino Médio**. Cada proposta foi estruturada com **objetivos pedagógicos claros**, instruções detalhadas, exemplos e trechos de código comentados, de modo a favorecer o **raciocínio lógico**, a **criatividade** e o desenvolvimento do **pensamento computacional**.

Conclui-se que a **criptografia**, além de sua relevância tecnológica, configura-se como uma **estratégia didática inovadora** e **instigante** para o ensino da **Matemática**. Ao desvendar códigos, criar mensagens cifradas e compreender os cálculos que sustentam esses processos, os estudantes vivenciam uma **Matemática** mais concreta, **interdisciplinar** e **motivadora**, que contribui para sua **formação crítica** e para sua preparação diante dos **desafios da sociedade digital**.

REFERÊNCIAS

BARBOSA, V. C. Fundamentos da Matemática Discreta. 2. ed. Rio de Janeiro: LTC, 2017.

BORBA, M. C.; PENTEADO, M. G. Educação Matemática e Tecnologias Digitais: interfaces com a prática docente. Belo Horizonte: Autêntica, 2016.

BRASIL. Ministério da Educação. Base Nacional Comum Curricular. Brasília: MEC, 2018. Disponível em: <http://basenacionalcomum.mec.gov.br>. Acesso em: [coloque a data de acesso].

CARNEIRO, A. História da criptografia: das cifras antigas à segurança da informação. São Paulo: Érica, 2017.

CLAYBOURNE, Anna. 91 truques matemáticos legais. São Paulo: Pé da Letra, 2021.

COUTINHO, S. C. A matemática e os números secretos: uma introdução à criptografia. 2. ed. Rio de Janeiro: Zahar, 2009.

GAUSS, C. F. Disquisitiones Arithmeticae. Leipzig: Gerh. Fleischer, 1801.

HEFEZ, A. Teoria dos Números: uma introdução. Rio de Janeiro: SBM, 2022.
HERÓDOTO. História. Livro V. Tradução de Mário da Gama Kury. Brasília: Editora da UnB, [s.d.].

OLIVEIRA, J. P. Introdução à Teoria dos Números. São Paulo: Livraria da Física, 2007.
PAPERT, S. *Mindstorms: children, computers, and powerful ideas.* New York: Basic Books, 1980.

SINGH, S. O livro dos códigos: a história da criptografia, da Antiguidade à era da informática. Tradução de Alyda Faber. Rio de Janeiro: Record, 2007.

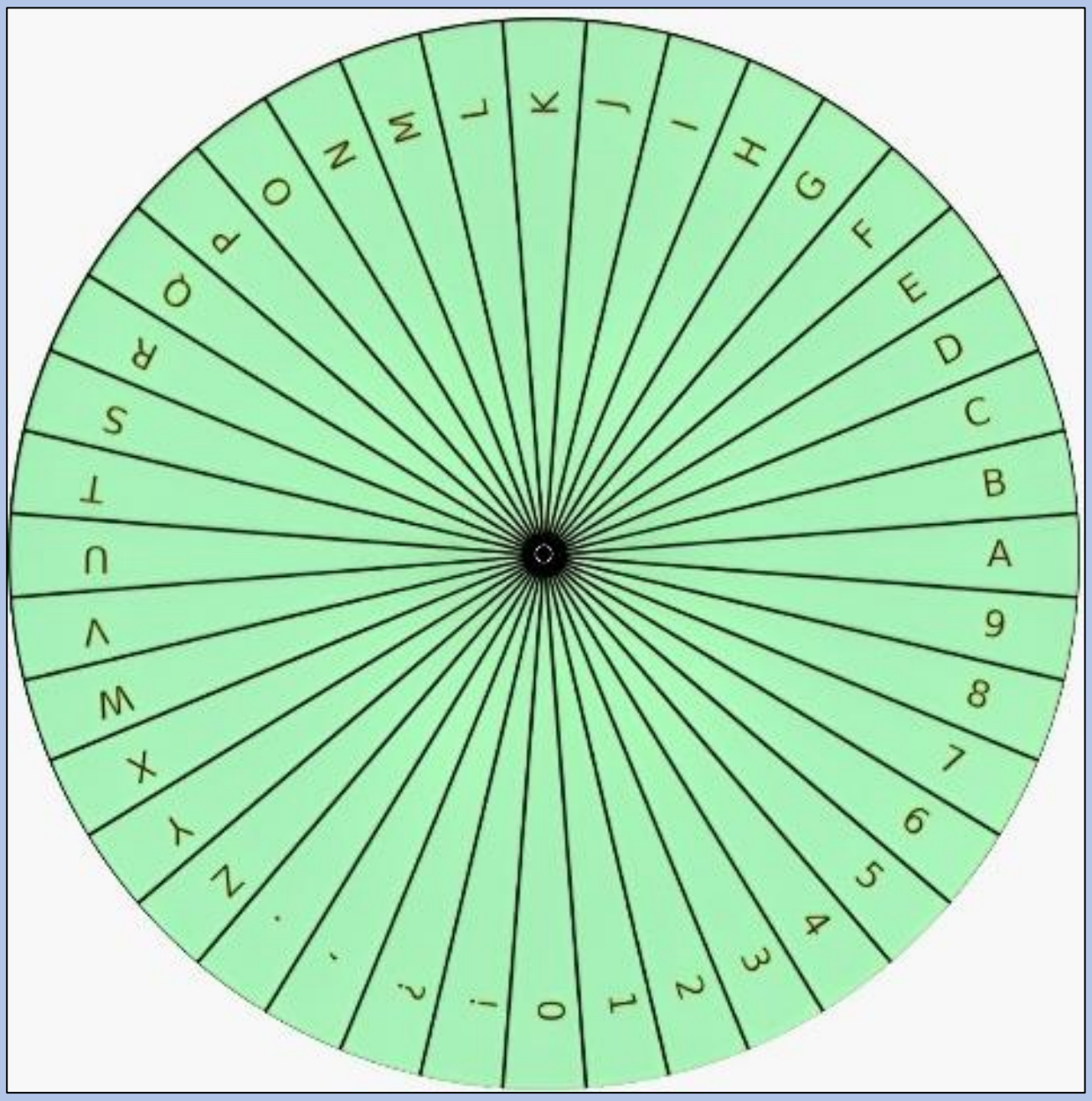
STALLINGS, W. Criptografia e segurança de redes: princípios e práticas. 4. ed. São Paulo: Pearson, 2006.

STEWART, I. Os números da natureza: os segredos da matemática que governa o universo. Rio de Janeiro: Zahar, 2015.

TANENBAUM, A. S.; WETHERALL, D. J. Redes de computadores. 5. ed. São Paulo: Pearson, 2011.

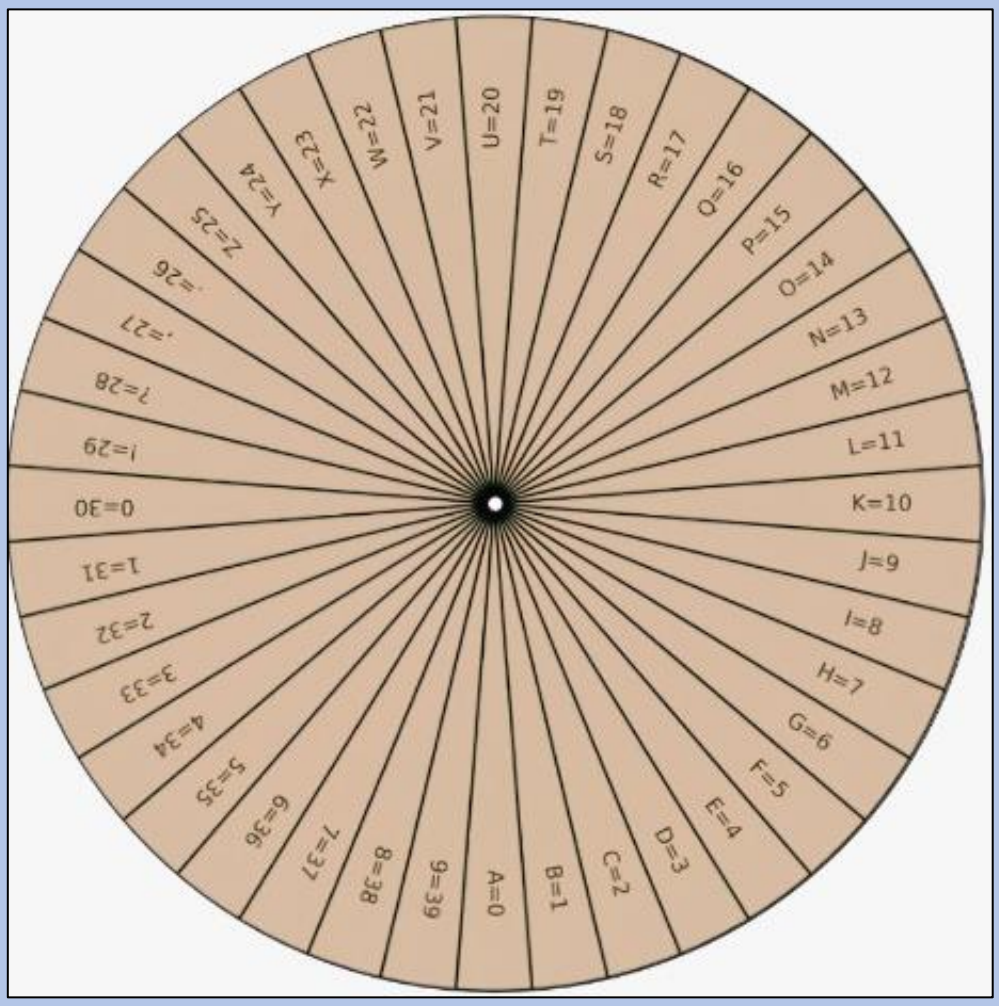
Anexo 1

Molde para o Criptodisco – Parte Inferior



Anexo 2

Molde para o Criptodisco – Parte Superior



Anexo 3

Tradução e explicação dos principais comandos C++ utilizados na implementação dos algoritmos de criptografia

Comando ou Função	Tradução / Explicação
#include <iostream>	Importa a biblioteca de entrada e saída (ex.: cin, cout).
#include <string>	Permite uso de strings (textos) no programa.
using namespace std;	Evita repetição de std:: antes de comandos padrão como cout, cin, etc.
int main()	Função principal do programa. Onde tudo começa a ser executado.
getline(cin, mensagem)	Lê uma linha completa de texto digitada pelo usuário.
cin >> chave	Lê um valor (geralmente numérico) digitado pelo usuário.
isalpha(letra)	Verifica se o caractere é uma letra (A–Z ou a–z).
isupper(letra)	Verifica se a letra é maiúscula.
toupper(letra)	Converte letra minúscula para maiúscula.
letra − base	Transforma a letra em um número (ex: A → 0, B → 1...).
base + c	Converte um número de volta em letra.
%	Operador de módulo: retorna o resto da divisão. Usado na aritmética modular.
if, else, while, for	Estruturas de controle condicional ou repetição.
return valor;	Encerra a função e devolve um valor.
string resultado = \"\";	Inicializa uma string vazia onde será armazenado o texto cifrado.
size_t pos = ALFABETO.find(letra)	Busca a posição do caractere na string do alfabeto.
expMod(base, expoente, n)	Função para cálculo de potência modular (usada no RSA).
inversoModular(e, phi)	Função para calcular o inverso de e mod phi usando o algoritmo de Euclides.
mdc(a, b)	Calcula o máximo divisor comum entre dois números.
static_cast<char>(...)	Converte um número inteiro para caractere com segurança.
cout << ...	Imprime textos ou resultados na tela.