

◀Manipulação de Strings▶

Kaio Christaldo
Fabricio Matsunaga

◀Conteúdos▶


- **Bibliotecas**
 - ◀string▶
 - ◀cctype▶
 - ◀sstream▶
 - ◀iomanip▶
- **Problemas Motivacionais**
- **Lista de Exercícios**

Apresentação Problema Motivador

beecrowd | 1024



Criptografia

Por Neilor Tonin, URI  Brasil

Timelimit: 1

Solicitaram para que você construísse um programa simples de criptografia. Este programa deve possibilitar enviar mensagens codificadas sem que alguém consiga lê-las. O processo é muito simples. São feitas três passadas em todo o texto.

Na primeira passada, somente caracteres que sejam letras minúsculas e maiúsculas devem ser deslocadas 3 posições para a direita, segundo a tabela ASCII: letra 'a' deve virar letra 'd', letra 'y' deve virar caractere 'l' e assim sucessivamente. Na segunda passada, a linha deverá ser invertida. Na terceira e última passada, todo e qualquer caractere a partir da metade em diante (truncada) devem ser deslocados uma posição para a esquerda na tabela ASCII. Neste caso, 'b' vira 'a' e 'a' vira ' '.

Por exemplo, se a entrada for “Texto #3”, o primeiro processamento sobre esta entrada deverá produzir “Wh{wr #3”. O resultado do segundo processamento inverte os caracteres e produz “3# rw{hW”. Por último, com o deslocamento dos caracteres da metade em diante, o resultado final deve ser “3# rvzgV”.

Entrada

A entrada contém vários casos de teste. A primeira linha de cada caso de teste contém um inteiro **N** ($1 \leq N \leq 1 \cdot 10^4$), indicando a quantidade de linhas que o problema deve tratar. As **N** linhas contém cada uma delas **M** ($1 \leq M \leq 1 \cdot 10^3$) caracteres.

Saída

Para cada entrada, deve-se apresentar a mensagem criptografada.

Exemplo de Entrada	Exemplo de Saída
4 Texto #3 abcABCl vxpdylY .ph vv.xwfxo.fd	3# rvzgV lFECedc ks. \n{frzx gi.r{hyz-xx



◀Strings▶

«Strings»

No c++, strings são sequências de caracteres que são usados para armazenar palavras e textos. Elas também são usadas para armazenar dados, como números e outros tipos de informações na forma de texto.

Strings são providas pelo arquivo de cabeçalho «string» na forma de uma classe string.

«Strings» – C style string

Como o C++ é uma extensão do C, ele herda sua forma de criar strings. Em C, strings são um array de caracteres terminados em um caracter NULL `'\0'`.

«Strings» – C style string



```
1 char str[50];
2
3 // Using character list
4 char str[] = {'G', 'e', 'e', 'k', 's', '\0'};
5
6 // Using string literal
7 char str[] = "Geeks";
```



```
1 char str[] = "Geeks";
2
3 // Access first character
4 // of string
5 printf("%c", str[0]);
6
7
8 char str[] = "Geeks";
9
10 // Update the first
11 // character of string
12 str[0] = 'R';
13 printf("%c", str[0]);
14
```



```
1 char greetings[] = "Hello World!";
2 printf("%s", greetings);
```



```
1 char str[5];
2
3 // Read string
4 // from the user
5 scanf("%s", str);
6
7 // Print the string
8 printf("%s", str);
```

«Strings» – String Class

Diferente das strings em C, que por serem arrays, tinham as seguintes limitações:

- **Tamanho Fixo: uma vez declarado, seu tamanho não mudava.**
- **Falta de operações de String fáceis: nenhuma operação de alto nível como concatenação ou extração de substring. Moreover, updating was also complex.**

«Strings» – String Class



```
1  string str_name;  
2  
3  string str = "Some Text here";
```



```
1  // Creating a string  
2  string greeting = "Welcome to GfG!";  
3  
4  // Accessing string  
5  cout << greeting;
```



```
1  string str = "Sonu";  
2  
3  // Accessing 3rd character  
4  cout << str[2] << endl;  
5  
6  // Accessing first character  
7  cout << str[0];
```



```
1  string str = "Tara";  
2  cout << str << endl;  
3  
4  // Updating string  
5  str = "Singh";  
6  cout << str;
```



```
1  #include <iostream>  
2  using namespace std;  
3  
4  cin >> str;
```

«Strings» – String Class – Funções

find() – Encontra uma substring em uma string.

rfind() – Encontra a ultima ocorrencia de uma substring em uma string.

append() – Realiza a concatenação de string.

insert() – Insere uma string em uma string.

erase() – Apaga caracteres de uma string.

replace() – Substitui porções de uma string.

compare() – Compara duas string.

«Strings» – String Class – Procurando substrings



```
1  string str = "Hello world, wonderful world!";
2  cout << "String: " << str << endl;
3
4  // find the first occurrence
5  size_t first_occurrence = str.find("world");
6
7  // find the last occurrence
8  size_t last_occurrence = str.rfind("world");
9
10 if (first_occurrence != string::npos) {
11     cout << "First occurrence: 'world' found at position: " << first_occurrence << endl;
12     cout << "Last occurrence: 'world' found at position: " << last_occurrence << endl;
13 }
14 else {
15     cout << "'world' not found" << endl;
16 }
```

```
String: Hello world, wonderful world!
First occurrence: 'world' found at position: 6
Last occurrence: 'world' found at position: 23
```

«Strings» – String Class – Acrescentando strings



```
1  #include <iostream>
2  #include <string>
3  using namespace std;
4
5  int main() {
6
7      string str = "Hello world,";
8      cout << "Before: " << str << endl;
9
10     //append the string
11     str.append(" Have a good day!");
12
13     cout << "After: " << str << endl;
14
15     return 0;
16 }
```

Before: Hello world,
After: Hello world, Have a good day!



```
1  #include <iostream>
2  #include <string>
3  using namespace std;
4
5  int main() {
6
7      string str = "Hello world, world!";
8      cout << "Before: " << str << endl;
9
10     // insert "beautiful" at the 13th index
11     str.insert(13, " beautiful");
12
13     cout << "After: " << str << endl;
14
15     return 0;
16 }
```

Before: Hello world, world!
After: Hello world, beautifulworld!

«Strings» – String Class – Apagando N caracteres de uma certa posição.



```
1  #include <iostream>
2  #include <string>
3  using namespace std;
4
5  int main() {
6
7      string str = "Hello world, beautiful world!";
8      cout << "Before: " << str << endl;
9
10     // erase five characters starting from the seventh index
11     str.erase(7, 5);
12
13     cout << "After: " << str << endl;
14
15     return 0;
16 }
17
```

Before: Hello world, beautiful world!
After: Hello w beautiful world!

«Strings» – String Class – Substituindo N caracteres dentro de uma String



```
1  #include <iostream>
2  #include <string>
3  using namespace std;
4
5  int main() {
6
7      string str = "Hello world, beautiful world!";
8      cout << "Before: " << str << endl;
9
10     // replace two characters with "Earth"
11     // starting from the seventh index
12     str.replace(6, 2, "Earth");
13
14     cout << "After: " << str << endl;
15
16     return 0;
17 }
```

Before: Hello world, beautiful world!
After: Hello Earthrld, beautiful world!

«Strings» – String Class – Comparar Strings Alfabeticamente



```
1  #include <iostream>
2  #include <string>
3  using namespace std;
4
5  int main() {
6
7      string str1 = "Hello world";
8      string str2 = "Hello world";
9      string str3 = "Hello";
10     string str4 = "Hello world, What's Up?";
11
12     cout << "String 1: " << str1 << endl;
13     cout << "String 2: " << str2 << endl;
14     cout << "String 3: " << str3 << endl;
15     cout << "String 4: " << str4 << endl;
16
17     // compare the strings
18     cout << "Comparing String 1 and String 2: " << str1.compare(str2) << " (Equal)" << endl;
19     cout << "Comparing String 1 and String 3: " << str1.compare(str3) << " (String 1 is Longer)" << endl;
20     cout << "Comparing String 1 and String 4: " << str1.compare(str4) << " (String 1 is Smaller)" << endl;
21
22     return 0;
23 }
```

```
String 1: Hello world
String 2: Hello world
String 3: Hello
String 4: Hello world, What's Up?
Comparing String 1 and String 2: 0 (Equal)
Comparing String 1 and String 3: 6 (String 1 is Longer)
Comparing String 1 and String 4: -12 (String 1 is Smaller)
```

◀cctype▶

◀cctype▶ – Classificação de caracteres

Para realizar classificação de caracteres em C++ é possível através das funções da biblioteca ◀cctype▶

isalnum() – Verifica se um caracter é alfanumerico

isalpha() – Verifica se um caracter é alfabetico

islower() – Verifica se um caracter é minuscuro

isupper() – Verifica se um caracter é maiusculo

isdigit() – Verifica se um caracter é um digito

isxdigit() – Verifica se um caracter é hexadecimal

«ctype» – Classificação de caracteres

- isctrl()** – Verifica se um caracter é um caracter de controle
- isgraph()** – Verifica se um caracter é um caracter gráfico
- isspace()** – Verifica se um caracter é um espaço
- isblank()** – Verifica se um caracter é um caracter vazio
- isprint()** – Verifica se um caracter é imprimível
- ispunct()** – Verifica se um caracter é um simbolo de pontuação
- tolower()** – Converte um caractere para caixa baixa
- toupper()** – Converte um caractere para caixa alta

⟨cctype⟩ – Exemplos



```
1 // C++ program to demonstrate isalpha()
2 #include <cctype>
3 #include <iostream>
4 using namespace std;
5
6 // Driver Code
7 int main()
8 {
9     // initializing character array
10    char ch[5] = "g1";
11
12    // checking for isalpha() function
13    for (int i = 0; i < 2; i++) {
14        if (isalpha(ch[i]))
15            cout << ch[i] << " is alphabet" << endl;
16        else
17            cout << ch[i] << " is not alphabet" << endl;
18    }
19 }
```



```
1 // C++ program to demonstrate isupper()
2 #include <cctype>
3 #include <iostream>
4 using namespace std;
5
6 // Driver Code
7 int main()
8 {
9     // initializing character array
10    char ch[5] = "Gg";
11
12    // checking for isupper() function
13    for (int i = 0; i < 2; i++) {
14        if (isupper(ch[i]))
15            cout << ch[i] << " is an upper alphabet" << endl;
16        else
17            cout << ch[i] << " is not an upper alphabet" << endl;
18    }
19 }
```

◀cctype▶ – Exemplos



```
1 // Driver Code
2 int main()
3 {
4     // initializing character array
5     char ch[4] = " \n\t";
6
7     // checking for isblank() function
8     for (int i = 0; i < 3; i++) {
9         if (isblank(ch[i]))
10             cout << " Character is blank" << endl;
11         else
12             cout << " Character is not blank" << endl;
13     }
14 }
```



```
1 // C program to demonstrate
2 // example of toupper() function.
3 #include <cctype.h>
4 #include <stdio.h>
5
6 int main()
7 {
8     char ch;
9
10    ch = 'g';
11    printf("%c in uppercase is represented as %c",
12           ch, toupper(ch));
13
14    return 0;
15 }
```

Resolução do Problema Motivador

1024 – Criptografia

A resolução estará disponível no Drive. Tente resolver por conta própria e, se precisar, compare com a solução! 😊

Apresentação Problema Motivador

beecrowd | 2588



Jogo dos Palíndromos

Por Ricardo Martins, IFSULDEMINAS 🇧🇷 Brazil

Timelimit: 1

Rener era um garoto que adorava palíndromos. Tanto que inventou um jogo com estes. Dada uma sequência de letras, quantas mais teriam que ser adicionadas, pelo menos, de modo que alguma permutação desta sequência formasse um palíndromo. Por exemplo, batata precisa adicionar um b no final, para virar o palíndromo batatab. Em outro exemplo, aabb, não precisa adicionar nenhuma letra, pois se faz o palíndromo abba ou baab. Em mais um exemplo, abc precisa de duas letras a mais, para formar um palíndromo, que pode ser abcba, acbca, bacab, bcacb, cabac ou cbabc. Escreva um programa que, dada uma sequência de letras, informe qual é o mínimo de letras que precisam ser adicionadas à sequência, para que haja, pelo menos, um anagrama que forme um palíndromo.

Entrada

Haverá diversos casos de teste. Em cada caso, é mostrada uma sequência de, no máximo, 1000 letras. Os casos de teste terminam com fim de arquivo.

Saída


Para cada caso de teste, imprima um valor inteiro, correspondente à quantidade mínima de letras a serem acrescentadas para que a sequência se torne um palíndromo, em uma de suas permutações.

2588 – Jogo dos
Palíndromos

◀**sstream**▶

Biblioteca <sstream>

- O sstream é uma biblioteca (ou mais precisamente, um cabeçalho da STL da linguagem C++).
 - Ela fornece classes baseadas em templates para manipulação de streams de string (fluxos de entrada/saída usando strings em vez de arquivos ou console). Ou seja:
 - Ele contém classes baseadas em templates, como:
 - istreamstream (entrada de string)
 - ostreamstream (saída para string)
 - stringstream (entrada e saída)



```
1  #include <iostream>
2  #include <sstream>
3  using namespace std;
4
```


Biblioteca **<sstream>**

Operações

Principais Operações Usadas:

- **istringstream s(str)** – Cria stream de leitura a partir da string str – **$O(1)$**
- **s >> var** – Extrai valor (como cin) da string – **$O(1)$** por leitura
- **ostringstream s** – Cria stream de escrita (para construir uma string formatada) – **$O(1)$**
- **s << var** – Insere valor formatado na string – **$O(1)$** por inserção
- **stringstream s(str)** – Stream de leitura e escrita – **$O(1)$**
- **s.str()** – Retorna string atual da stream – **$O(n)$**

Biblioteca `<sstream>`

Operações

Principais Operações Usadas:

- **`s.str(str2)`** – Redefine a string da stream – **$O(1)$**
- **`s.clear()`** – Limpa erros de leitura (EOF/fail) para reutilizar stream – **$O(1)$**
- **`getline(s, var)`** – Lê até '`\n`' da stream – **$O(k)$** , k = tamanho da linha
- **`stoi(s)` / `stod(s)`** – Conversão direta string \rightarrow int / double (útil para parsing manual) – **$O(\log x)$**

Biblioteca `<sstream>`

- `istringstream s(str)` e `s >> var`



```
1  string linha = "10 20 30";  
2  istringstream s(linha);  
3  
4  int a, b, c;  
5  s >> a >> b >> c; // 0(1) por leitura  
6  
7  cout << a << " " << b << " " << c << "\n"; // Saída: 10 20 30  
8
```

Biblioteca `<sstream>`

- `ostringstream s` e `s << var`



```
1  ostringstream s;  
2  int x = 42, y = 10;  
3  s << "Resposta: " << x << "\n"; // 0(1) por inserção  
4  s << "Resposta: " << y << "\n"; // 0(1) por inserção  
5  
6  cout << s.str(); // Saída: Resposta: 42  
7                      // Responde: 10
```

Biblioteca `<sstream>`

- `stringstream s(str)` e `s.str()`



```
1  stringstream s("123 abc");
2  int n; string palavra;
3  s >> n >> palavra;
4
5  s.str("");           // limpa conteúdo
6  s.clear();           // limpa flags de leitura
7
8  s << "resultado: " << palavra << " " << n << "\n";
9  cout << s.str();
```

Biblioteca <sstream>


- `s.str(str2)`



```
1  stringstream s("antigo");
2  cout << "Antes: " << s.str() << endl;
3
4  s.str("novo conteudo"); // redefine o conteúdo da stream
5  cout << "Depois: " << s.str() << endl;
```

Biblioteca `<sstream>`

- `s.clear()`



```
1  stringstream s("1 2");
2  int a, b, c;
3  s >> a >> b >> c; // falha ao ler o terceiro
4
5  if (s.fail()) {
6      cout << "Falha detectada!\n";
7      s.clear(); // limpa os erros (necessário antes de reutilizar)
8  }
9
10 s.str("3 4 5");
11 s >> a >> b >> c;
12 cout << a << " " << b << " " << c << endl;
```

Biblioteca <sstream>

- `getline(s, linha)`



```
1
2  stringstream s("linha 1\nlinha 2\nlinha 3");
3  string linha;
4
5  while (getline(s, linha)) {
6      cout << "[" << linha << "]" << endl;
7  }
8
```


Biblioteca <sstream>

- `stoi(s1) / stod(s2)`



```
1  string s1 = "12345";  
2  string s2 = "3.14159";  
3  
4  int x = stoi(s1);      // string para int  
5  double pi = stod(s2);  // string para double  
6  
7  cout << x << " " << pi << endl;
```

Biblioteca <sstream>

Ideias:

- Função Split com getline()


```
1 vector<string> split(const string& s, char delim) {  
2     vector<string> resultado;  
3     resultado.erase(remove(resultado.begin(), resultado.end(), ' '), resultado.end());  
4  
5     istringstream ss(s);  
6     string item;  
7     while (getline(ss, item, delim)) {  
8         resultado.push_back(item);  
9     }  
10    return resultado;  
11 }  
12
```

```
1 string linha = "banana, maçã, laranja, uva";  
2 vector<string> frutas = split(linha);  
3  
4 for (string f : frutas)  
5     cout << "[" << f << "]\n";  
6
```

◀**iom manip**▶

Biblioteca **<iomanip>**

- **<iomanip>** — Biblioteca de formatação de entrada/saída
 - O cabeçalho **<iomanip>** (input/output manipulators) faz parte da STL e fornece manipuladores de fluxo para controlar a formatação da entrada/saída em C++.
 - Ele funciona junto com **cin**, **cout**, **stringstream**, etc.



```
1  #include <iostream>
2  #include <sstream>
3  using namespace std;
4
```

Biblioteca <iomanip>

Manipuladores

Principais Manipuladores Usadas:

- **setprecision(n)** – Define número de dígitos significativos ou decimais
- **fixed** – Usa notação decimal fixa (p/ float/double)
- **setw(n)** – Define largura mínima de campo na saída
- **setfill(c)** – Preenche espaços com caractere c
- **left, right, internal** – Alinhamento à esquerda, direita ou interno
- **showpos / noshowpos** – Mostra ou oculta o sinal + em números positivos
- **hex, dec, oct** – Define base numérica da saída
- **showbase** – Mostra prefixo da base (0x, 0, etc.)
- **uppercase** – Letras maiúsculas na notação científica ou hexadecimal

Biblioteca `<sstream>`

- `setprecision(n) + fixed`



```
1 double pi = 3.14159265;  
2 cout << fixed << setprecision(3) << pi << endl; // 3.142
```

Biblioteca <sstream>

- `setw(n) + setfill(c) + left / right`



```
1 cout << setfill('.') << setw(8) << 42 << endl;           // .....42 (right default)
2 cout << left << setw(8) << 42 << endl;                     // 42.....
3 cout << right << setw(8) << 42 << endl;                    // .....42
```

Biblioteca **<sstream>**

- **showpos / noshowpos**



```
1  
2  cout << showpos << 5 << " " << -3 << endl;  // +5 -3  
3
```


Biblioteca <sstream>

- hex, oct, dec + showbase + uppercase



```
1 int n = 255;
2 cout << showbase << hex << n << endl;           // 0xff
3 cout << showbase << uppercase << hex << n << endl; // 0XFF
4 cout << dec << n << endl;                         // 255
5 cout << oct << n << endl;                         // 0377
```



```
1 int x;
2 cin >> hex >> x; // Entrada: 1a → x = 26
3 cout << "Decimal: " << x << endl;
```

Resolução do Problema Motivador

2588 – Jogo dos Palíndromos

Dicas:

- Contar a frequência dos caracteres

A resolução estará disponível no Drive. Tente resolver por conta própria e, se precisar, compare com a solução! 😊

Lista de Exercícios

2588 – Jogo dos Palíndromos

1024 – Criptografia

2242 – Huaauhahhuahau

1257 – Array Hash

1168 – LED

1255 – Frequência de Letras

1871 – Zero vale Zero



Se tiver alguma dúvida ou dificuldade na resolução de algum exercício, sinta-se à vontade para perguntar! 😊

Referências

[1] CPLUSPLUS.COM. C++ algorithm. Disponível em: <https://cplusplus.com/reference/algorithm/>. Acesso em: 15 mai. 2025.

[2] CPLUSPLUS.COM. C++ iterator. Disponível em: <https://cplusplus.com/reference/iterator/>. Acesso em: 15 mai. 2025.

[3] CPLUSPLUS.COM. C++ tuple. Disponível em: <https://cplusplus.com/reference/tuple/>. Acesso em: 15 mai. 2025.

GEEKSFORGEEKS. Heap em C++ STL. GeeksforGeeks, [S. l.], [s. d.]. Disponível em: https://www-geeksforgeeks-org.translate.goog/cpp-stl-heap/?_x_tr_sl=en&_x_tr_tl=pt&_x_tr_hl=pt&_x_tr_pto=tc. Acesso em: 8 maio 2025.