aula1

Básico de C++

Template

Para soluções em C++, normalmente, vamos utilizar o seguinte template:

```cpp
#include <bits/stdc++.h>

using namespace std;

using ll = long long;

#define int ll

int32_t main() {
    cin.tie(0);
    ios_base::sync_with_stdio(false);

    // Solução

    return 0;
}
```

A solução do problema vai ser colocada no lugar do comentário "Solução". Em algums sites (e.g. codeforces), os problemas normalmente contém "casos de teste", que são várias instâncias do problema sem nenhuma relação entre si.

Nesses casos, pode ser útil usar o seguinte template:

```cpp
#include <bits/stdc++.h>

using namespace std;

using ll = long long;
```

```cpp
#define int ll

void solve() {
    // Solução
}

int32_t main() {
    cin.tie(0);
    ios_base::sync_with_stdio(false);

    int t;
    cin >> t;

    for (int i = 0; i < t; i++) {
        solve();
    }

    return 0;
}
```

Nós usamos uma função "solve" para poder lidar com testes de caso isoladamente, já que eles são simplesmente uma instância do problema. Além disso, ainda podemos usar `return` para terminar um teste de caso de forma conveninente.

Tipos de dados muito comuns e úteis da STL

Vector

Vectors são arrays dinâmicos, eles podem ser entendidos como a forma moderna de um array (exceto pelo fato de que os buffers são alocados na heap).

Operações comuns de arrays também são válidas em vectors:

```cpp
vector<int> vec(5);

// Acessar um elemento do vector
cout << vec[0] << "\n";

// Alterar um elemento do vector
vec[1] = 3;

// Ler um número e colocar no vector
cin >> vec[2];
```

Coisas úteis sobre vectors:

```cpp
// Cria um vector vazio (0 elementos)
vector<int> vec;

// Colocar o número 5 no final do vector (isso aumenta o tamanho do vector)
vec.push_back(5);

// Conseguir o número de elementos no vector
vec.size();

// Deletar o último elemento do vector
vec.pop_back();

// Criar um vector com 10 elementos, todos eles
// inicializados com o valor padrão do tipo do vector
vector<int> vec(10);

// Criar um vector com 10 elementos, todos eles com valor 3
vector<int> vec(10, 3);

// Ordenar os elementos de um vector
sort(vec.begin(),vec.end());
```

Exercícios:

1. Escreva uma função que retorna o elemento do meio de um vector.
2. Escreva uma função que inverte a ordem dos elementos de um vector.
3. Escreva uma função que recebe um `int` e retorna os dígitos desse `int` na forma de um vector.

Strings

Assim como o `vector` é como uma versão moderna de um array, uma string é como uma versão moderna de um `char[]`.
É possível fazer as mesmas coisas de `char[]` em uma string, mas ela inclui várias coisas mais convenientes.

```
// Criar uma string vazia
string s;

// Ler uma string sem espaços
cin >> s;

// Conseguir tamanho da string
s.length();

// Conseguir um caractere na string
cout << s[3] << "\n";

// Modificar um caractere na string
s[4] = 'b';

// Adicionar um caractere ao final da string
s += 'c';

// Adicionar uma string ao final da string
s += "abcde";

// Comparar duas strings
if (s == "ABC") {
```

```
    // ...
}
```

Exercícios:

1. Escreva uma função que troca todos os a's de uma string por b's e todos os b's por a's.
2. Escreva uma função que conta quantas vogais existem em uma string.

ASCII

[Tabela ASCII]

Fatos úteis: ```cpp // Lowercase ⟷ Uppercase char c = 'a'; c ^= ' '; // c == 'A' c ^= ' '; // c == 'a'

```
// To uppercase
char c = 'a';
c = c & '_'; // c == 'A'
// To lowercase
c = c | ' '; // c == 'a'
```


Exercícios:
1. Escreva uma função que converte todos os valores de uma string para uppercase.
2. Escreva uma função que troca todos os valores de uma string entre uppercase e lowercase.
3. Escreva uma função que remove todos as letras maiúsculas de uma string, e retorna uma nova string.
4. **Desafio**: Escreva uma função que remove todos os caracteres maiúsculos de uma string, sem criar uma nova string.

```
<br>
```