Programação

list

list

- Uma lista é uma estrutura de dados sequencial, que permite inserções e remoções em qualquer posição em **tempo constante**
  - Além de iterações em ambos os sentidos.

list

- Vantagens
  - Quando comparada com *vector* e *deque*, a lista geralmente tem melhor desempenho para inserção, remoção e movimentação de elementos em qualquer posição no contêiner;
  - Consequentemente, algoritmos como os ordenação têm melhor desempenho.
- Desvantagem
  - Ainda comparada a *vector* e *deque*, as listas não possuem acesso direto aos elementos, de acordo com sua posição ou índice;
  - É necessário iterar de uma posição conhecida até o elemento desejado, em **tempo linear**.

list

```
#include <iostream>
#include <list>
using namespace std;
int main ()
{
  list<int> first; // lista de inteiros vazia
  first.push_front( 1 ); //insere na frente
  first.push_front( 2 );
```

list

```
  first.push_back( 4 ); //insere no final
  first.push_back( 1 );
  first.remove( 4 ); // remove todos os 4s
  first.unique(); // remove elementos duplicados
  first.pop_front(); // remove elemento da parte da frente
  first.pop_back(); // remove elemento da parte de trás
  first.sort(); // ordena values
  cout << "O conteúdo é: ";
  for (list<int>::iterator it = first.begin(); it != first.end(); it++)
     cout << *it << " ";
  return 0;
}
```

list

- Neste exemplo, temos os seguintes métodos da classe *list*:
  - *sort*: ordena a lista em ordem crescente;
  - *unique*: remove elementos duplicados;
  - *remove*: apaga todas as ocorrências de um determinado valor da lista.
- Existem outros como:
  - *reverse*: inverte a lista;
  - *merge*: intercala listas;
  - remove_if: remove elementos que atendam um critério.

deque

deque

- Um *deque* (***d****ouble-****e****nded* ***que****ue*) é uma fila de ponta dupla;
- Em outras palavras, é uma estrutura de dados sequencial e de tamanho dinâmico, podendo ser expandida ou contraída em ambas extremidades
  - No início (*frente*) ou no final.

deque

- Vantagens
  - O *deque* é muito parecido com *vector*, porém, é mais eficiente para inserção e remoção de elementos também no início;
  - Para armazenar muitos dados, é melhor que *vector*, pois realoca memória mais facilmente.
- Desvantagens
  - Ao contrário do *vector*, não é uma estrutura contígua;
  - Quando comparado com lista, o *deque* tem pior desempenho em operações que envolvem inserções e remoções frequentes em posições que não sejam o início ou o final.

deque

```
#include <iostream>
#include <deque>
using namespace std;
int main ()
{
  unsigned int i;
  deque<int> first;          // deque vazio do tipo int
```

deque

```
  first.push_front( 2 );
  first.push_front( 3 );
  first.push_back( 1 );
  // utiliza o operador de subscrito para modificar elemento na localização 1
  first[ 1 ] = 5;
  first.pop_front(); // remove o primeiro elemento
  first.pop_back(); // remove o primeiro elemento
  cout << "O conteúdo é:";
  for (i=0; i < first.size(); i++)
    cout << " " << first[i];
  return 0;
}
```

deque

- O método *push_front* está disponível apenas para *list* e *deque*;
- O operador **[]** permite acesso direto aos elementos do *deque*
  - Também pode ser utilizado em um *vector*.
- Em geral, um *deque* possui um desempenho levemente inferior em relação a um *vector*
  - No entanto, é mais eficiente para fazer inserções e remoções no início.

Problemas Selecionados

Problemas Selecionados

- http://www.urionlinejudge.com.br/judge/en/problems/view/1430
- http://www.urionlinejudge.com.br/judge/en/problems/view/1025

Perguntas?