Lab 12:

STL – Parte 2
Standard Template Library

ECOP13A - Programação Orientada a Objetos
Prof. André Bernardi
andrebernardi@unifei.edu.br

Universidade Federal de Itajubá

02 Jul. 2025

Estruturas não-lineares

Para alguns problemas, existem maneiras melhores de se representar os dados do que uma simples sequência. Com as implementações da STL das estruturas não-lineares que iremos discutir a seguir, podemos aumentar nossa eficiência na aplicação de algoritmos sempre que o problema em questão apresentar condições favoráveis para sua utilização.

De que estruturas estamos falando?

- Balanced Binary Search Tree (BST): <map> e <set>
- Heap: priority_queue <queue>

Árvores Binárias de Busca (BST)

STL <map> e <set> são implementações de um tipo de árvore binária de busca balanceada chamada **Red-Black Tree**, ou Árvore Rubro-Negra. Portanto, nelas, todas as operações são realizadas em $O(\log n)$.

Qual a diferença entre as duas?

<map> armazena **pares** (chave, dado)

<set> armazena apenas a **chave**.

Exercício 1.

Utilizando a STL, escreva um programa em C++ para a demonstração do funcionamento de uma ***Heap*** ou ***Fila de Prioridade***, incluída através do cabeçalho <queue>. Faça um programa que deve mostrar repetidamente um menu com as opções que podem ser escolhidas pelo usuário. Ele deve funcionar de maneira semelhante ao exemplo a seguir:

```
--------------------------
Programa de Heap STL
--------------------------


1.Insira um elemento na heap
2.Remova um elemento da heap
3.Tamanho da heap
4.Primeiro elemento da heap
5.Sair
Escolha (1-5):  1 (cin)
Entre com o valor a ser inserido:  87 (cin)

1.Insira um elemento na heap
2.Remova um elemento da heap
3.Tamanho da heap
4.Primeiro elemento da heap
5.Sair
Escolha (1-5):  1 (cin)
Entre com o valor a ser inserido:  92 (cin)
```

```
--------------------------
1.Insira um elemento na heap
2.Remova um elemento da heap
3.Tamanho da heap
4.Primeiro elemento da heap
5.Sair
Escolha (1-5):  1 (cin)
Entre com o valor a ser inserido:  35 (cin)


1.Insira um elemento na heap
2.Remova um elemento da heap
3.Tamanho da heap
4.Primeiro elemento da heap
5.Sair
Escolha (1-5):  2 (cin)
Elemento 92 removido do topo da heap
```

```
1.Insira um elemento na heap
2.Remova um elemento da heap
3.Tamanho da heap
4.Primeiro elemento da heap
5.Sair

Escolha (1-5):  3 (cin)
Tamanho da heap: 2
1.Insira um elemento na heap
2.Remova um elemento da heap
3.Tamanho da heap
4.Primeiro elemento da heap
5.Sair
Escolha (1-5):  4 (cin)
Primeiro elemento da heap (topo): 87
```

```
1.Insira um elemento na heap
2.Remova um elemento da heap
3.Tamanho da heap
4.Primeiro elemento da heap
5.Sair
Escolha (1-5):  5(cin)


Programa finalizado!
```

Exemplo exo1.cpp

```
#include <queue>
#include <iostream>
using namespace std;

priority_queue<int> filaP;

int menu () {
	cout << "\n1.Insira um elemento na heap\n";
	cout << "2.Remova um elemento da heap\n";
	cout << "3.Tamanho da heap\n";
	cout << "4.Primeiro elemento da heap\n";
	cout << "5.Sair\n";

	int aux = -1;

	while (!(1 <= aux && aux <= 5)) {
		cout << "Escolha (1-5): ";
		cin >> aux;
	}
	return aux;
}
```

```
int main () {

	cout << "-----------------------------\n";
	cout << "   Programa de Heap STL   \n";
	cout << "-----------------------------\n";

	int op = -1;

	while (op != 5) {
		op = menu();

		switch (op) {
			case 1:
				cout << "Entre com o valor a ser inserido: ";

				int val;
				cin >> val;

				filaP.push(val);
				break;
```

```
    case 2:
        if (!filaP.empty()) {
            cout << "Elemento " << filaP.top() << " removido\n";
            filaP.pop();
        } else {
            cout << "Heap vazia\n";
        }
        break;
    case 3:
        cout << "Tamanho da heap: " << filaP.size() << "\n";
        break;
    case 4:
        if (!filaP.empty()) {
            cout << "Primeiro elemento da heap: " << filaP.top() << "\n";
        } else {
            cout << "Heap vazia\n";
        }
    }
  cout << "\n";
 }
 cout << "Programa finalizado!\n";
 return 0;
}
```

Exercício 2.

(exo2.cpp) Utilizando a STL, escreva um programa em C++ para a demonstração do funcionamento de uma *árvore binária de busca balanceada* ou ***set***, incluída através do cabeçalho <set>. Faça um programa que deve mostrar repetidamente um menu com as opções que podem ser escolhidas pelo usuário. A árvore poderá conter valores repetidos, portanto escolha a classe adequada (multiset). Ele deve funcionar de maneira semelhante ao exemplo a seguir:

```
========================================
 Implementação de Árvore (set) no STL
========================================
1. Inserir elemento
2. Remover elementos com determinado valor
3. Exibir elementos em ordem
4. Exibir quantidade de elementos
5. Remover todos os elementos
6. Consultar quantidade de elementos com determinado valor
7. Sair

Escolha uma opção: (Usuário entra com cin)
```

Exemplo exo2.cpp

```
#include <set>
#include <iostream>
using namespace std;

multiset<int> conjunto;

void imprime () {
	for (auto x : conjunto)
		cout << x << " ";
	cout << endl;
}
int menu () {
	cout << "1. Inserir elemento \n";
	cout << "2. Remover elementos com determinado valor \n";
	cout << "3. Exibir elementos em ordem\n";
	cout << "4. Exibir quantidade de elementos\n";
	cout << "5. Remover todos os elementos\n";
	cout << "6. Consultar quantidade de elementos com determinado valor\n";
	cout << "7. Sair\n";
	int aux = -1;
	while (!(1 <= aux && aux <= 7)) {
		cout << "Escolha uma opcao: ";
		cin >> aux;
	}
	return aux;
}
```

```
int main () {
	cout << "=======================================\n";
	cout << " Implementação de Árvore (set) no STL \n";
	cout << "=======================================\n";

	int op = -1, val = -1;

	while (op != 7) {
		op = menu();

		switch (op) {
			case 1:
				cout << "Elemento a ser inserido: ";
				cin >> val;
				conjunto.insert(val);
				break;

			case 2:
				cout << "Elemento a ser removido: ";
				cin >> val;
				conjunto.erase(val);
				break;

			case 3:
				imprime();
				break;
```

```
                case 4:
                        cout << "Quantidade de Elementos: " << conjunto.size() << "\n";
                        break;

                case 5:
                        cout << "Removendo todos elementos: \n";
                        conjunto.clear();
                        break;

                case 6:
                        cout << "Elemento a ser consultado: ";
                        cin >> val;
                        cout << val << " aparece: "<< conjunto.count(val) << " vezes.\n";
                        break;
        }

        cout << "\n";
    }

    cout << "Programa finalizado!\n";

    return 0;
}
```

Referências

- https://cplusplus.com/reference/queue/priority_queue/
- https://cplusplus.com/reference/set/
  - https://cplusplus.com/reference/set/set/
  - https://cplusplus.com/reference/set/multiset/