Vetores dinâmicos.
Declaração (ex): **vector <int> vt;**
Iterador: **vector <int> :: iterator it;**

```
for (it = vt.begin(); it != vt.end(); it++)
    cout << *it << " ";
```

//mostra todos os valores do vetor

Comandos gerais:

.clear() // Limpa o vetor
.size() // Retorna o tamanho do vetor
sort(vt**.begin(),** vt**.end());** // Ordena o vetor em ordem crescente
reverse(vt**.begin(),** vt**.end());** // Inverte a ordem do vetor

- **Vector**

Pode ser acessado por [ ] ou iterador.
Ex:

```
for (int i = 0; i < vt.size(); i++)
    cout << vt[i] << " ";
```

Comandos importantes:

.push_back(x **)** // Insere o valor x no fim do vetor (#define pb)
.pop_back() // Apaga o último elemento inserido

- **Set**

Vetor ordenado automaticamente, apagando elementos repetidos — para trabalhar com elementos repetidos, usa-se "Multiset".
Não pode ser acessado com [ ], apenas iterador.
Para inserir valores, usa-se **.insert(** x **);**

- **Pair**

Trabalha com pares de valores, podendo ser usado com todas as estruturas.
Declaração (ex): **vector <pair <int, int> > vt;**
Comandos importantes:

.make_pair(x**,** y **);** // Cria um par com os valores x e y (#define mk)
Ex: **vt.pb(mk(2, 20));** //Insere o par (2, 20) no vetor
.first; //Retorna o primeiro elemento do par (no caso, x);
.second; //Retorna o segundo elemento do par (no caso, y);

Obs.: Ao acessar o par a partir do iterador, deve-se usar **it->first**.

Exemplo:

```
for(it = vt.begin(); it != vt.end(); it++)
    cout << it->first << " " << it->second << endl;
```

- **Map**

Permite a criação de índices de outros tipos além de inteiros, como strings.
Ex: **map <string, double> mp;** (um map de índice string e conteúdo double)
Necessita do iterador para acessar seus valores.
Ordena os índices de maneira crescente, excluindo repetidos (como o set).

```
for(it = mp.begin(); it != mp.end(); it++)
    cout << it->first << " " << it->second <<endl;
        //mostra o índice e seu conteúdo
```

Para inserir um valor no map (índice "Bolo" e conteúdo 3.50):
— **mp.insert("Bolo", 3.50);** ou
— **mp["Bolo"] = 3.50;**

- **Deque**

Idêntico ao vector, mas possibilita a inserção de valores também no início.
Pode ser acessado com [].
Comandos importantes:
.push_front(x **);** //insere o valor x no início
.pop_front(); //remove o primeiro elemento

- **Queue (fila)**

O primeiro a entrar é o primeiro a sair, literalmente uma fila.
Não é possível acessar elementos além do primeiro e do último.
Não é possível usar o comando **.clear()**, sendo necessário eliminar os elementos um por um.
Comandos importantes:
.push(x **);** //insere o elemento x no final da fila
.pop(); //remove o elemento do início da fila
.empty(); //verifica se a fila está vazia
.front(); //acessa o elemento do início da fila
.back(); //acessa o elemento do fim da fila
Para limpar a fila:

```
while (!q.empty())
    q.pop();
```

- **Priority_Queue (fila de prioridade)**

Adaptador da queue, no qual o primeiro elemento será sempre o maior entre todos.
Possui as mesmas operações da fila normal.

- **Pilha (Stack)**

O último elemento a entrar é o primeiro a sair, literalmente uma pilha.
Só é possível acessar o elemento do topo.
Não é possível usar o comando **.clear()**, sendo necessário eliminar os elementos um por um.
Comandos importantes:

.push(x **);** //insere um elemento no topo da pilha
.pop(); //remove o elemento do topo
.empty(); //verifica se a pilha está vazia
.top(); //acessa o elemento do topo