LABORATÓRIO DE PROGRAMAÇÃO I

Departamento de Ciência da Computação

AULA 1 - VETORES E ORDENAÇÃO

Vetores - vector

```
#include <iostream>
#include <vector>
using namespace std; // vector -> std

int main() {
	int v1[1000]; // vetor tradicional com 1000 elementos
	vector<int> v2(1000); // vector com 1000 elementos pré-alocados
	for(int i=0; i < 1000; i++) {
		cin >> v1[i] >> v2[i];
	}
}
// por que usar vector?
```

Vetores - push_back

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

int main() {
	vector<int> v;
	for(int i=0; i < 1000; i++) {
		int j;
		cin >> j;
		v.push_back(j); // função push_back adiciona elemento no final do vetor
	}                               // e aloca mais espaço caso necessário
}
```

Vetores - size

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

int main() {
	vector<int> v;
	int x;
	while(cin >> x && x >= 0)
		v.push_back(x);
	for(int i=0; i < v.size(); i++) // função size retorna a quantidade de elementos do vetor
		cout << v[i] << endl; // o acesso ao vetor é realizado normalmente
}
```

Vetores - clear

```
...
int main() {
        vector<int> v;
        int x;
        while(cin >> x && x >= 0)
                v.push_back(x);
        for(int i=0; i < v.size(); i++)
                cout << v[i] << endl;
        v.clear(); // função clear remove todos os elementos do vetor
        while(cin >> x && x >= 0)
                v.push_back(x);
}
```

Vetores - iterator

```
#include <iterator>
...
int main() {
	vector<int> v;
	int x;
	while(cin >> x && x >= 0)
		v.push_back(x);
	for(vector<int>::iterator it=v.begin(); it != v.end(); it++) // iterator é um ponteiro para elementos
		cout << *it << endl; // não imprima o iterator, e sim o valor apontado por ele!
}
```

Vetores - erase

```
...
int main() {
        vector<int> v;
        int x;
        while(cin >> x && x >= 0)
                v.push_back(x);
        v.erase(v.begin()); // apaga o primeiro elemento
        v.erase(v.begin()+1); // apaga o segundo elemento
        v.erase(v.begin()+2, v.end()); // apaga todos os elementos exceto os dois primeiros
}
```

Vetores - vector

Saiba mais em:

http://www.cplusplus.com/reference/vector/vector/

Ordenação - sort

```
#include <iostream>
#include <vector>
#include <algorithm>
using namespace std; // sort -> std
int main() {
	vector<int> v;
	for(int i=0; i < 1000; i++) {
		int j;
		cin >> j;
		v.push_back(j);
	}
	sort (v.begin(), v.end()); // ordenação com complexidade ~ NlogN (não estável)
}
```

Ordenação - stable_sort

```
#include <iostream>
#include <vector>
#include <algorithm>
using namespace std; // sort -> std
int main() {
	vector<int> v;
	for(int i=0; i < 1000; i++) {
		int j;
		cin >> j;
		v.push_back(j);
	}
	stable_sort (v.begin(), v.end()); // ordenação com complexidade = NlogN
}
```

Ordenação - sort & struct

```
…
struct pessoa {
        int id;
        string nome;
        ...
};
bool cmp(pessoa i, pessoa j) { return (i.id < j.id || i.id == j.id && i.nome < j.nome); }
int main() {
        vector<pessoa> v;
        ...
        stable_sort (v.begin(), v.end(), cmp);
}
```

Vetores - sort

Saiba mais em:

http://www.cplusplus.com/reference/algorithm/sort/