Introdução a computação competitiva (C++)

Giancarlo, Guilherme e Yan

Alguns conceitos da linguagem

1. Alteração do comportamento de operadores
2. Para cada operador, é possível chamar uma função

```
template <typename T>
class vector_2D {
private:
   T x[2];

public:
   vector_2D(T x0, T x1){
       this->x[0] = x0;
       this->x[1] = x1;
   }
   T operator[](unsigned index){
       return this->x[index];
   }

   vector_2D operator+(vector_2D op2){
       return (vector_2D){
           this->x[0] + op2.x[0],
           this->x[1] + op2.x[1]
       };
   }
};
```

Entrada e saída

```
cin >>
cout <<
getline(stream, string)
stringstream ss(string);
```

FORMAS PARA RECEBER A ENTRADA

```
int casos;
cin >> casos;
while(casos--){ … }
```

```
int n;
while(cin >> n){ … }
```

STL

1. Inclui as estruturas padrão da linguagem
2. As estruturas compartilham componentes e operações
    i. Iteradores:
        Extremidades: `.begin()`, `.end()`
        Extremidades inversas: `.rbegin()`, `.rend()`
    ii. Construtor: é possível construir estruturas a partir de outras

```
queue<T> Q(V.begin(), V.end());
```

    iii. Métodos: `.size()`

STL vector<T>

```
vector<T> V(size, value);
func(?const vector<T> &V);                    //Ignora o tamanho
```

1. Dinâmico
2. Flexível

```
V.push_back(T item);                  // Insere na posição final
V.insert(iterator pos, T item);       // Insere em qualquer posição
V.erase(iterator pos);                // Apaga uma posição
```

STL set<T>

```
set<T> S;
```

1. Retira repetição de elementos
2. Não é útil para operar elementos específicos

```
S.insert(T element);        // Insere um elemento
S.lower_bound(T element);   // Primeiro ≤
S.upper_bound(T element);   // Primeiro >
```

STL queue<T>

```
queue<T> Q;
```

1. Estrutura FIFO
2. Funcionamento padrão de qualquer fila

```
Q.front();              // Acessa o primeiro elemento
Q.pop();                // Retira o primeiro elemento
Q.back();               // Acessa o último elemento
Q.empty();              // Retorna se a pilha está vazia
Q.push();               // Adiciona um elemento no final da fila
```

STL stack<T>

```
stack<T> S;
```

1. Estrutura LIFO
2. Funcionamento padrão de qualquer pilha

```
S.top();                // Acessa o elemento do topo
S.pop();                // Retira o elemento do topo
S.empty();              // Retorna se a pilha está vazia
S.push();               // Adiciona um elemento no topo da pilha
```

STL map<T1,T2>

```
map<T1,T2> M;
```

1. Estrutura que relaciona chaves e valores
2. Pode ser útil para situações de contagem de elementos e agrupamentos em geral

```
map<string, int> estoque;
estoque[“balao”] = INT_MAX;
```

STL bitset<N>

```
bitset<N> B;
```

1. Estrutura preparada para operar bits
2. Precisa ser inicializada com um tamanho fixo
3. Facilita a criação de máscaras

```
B.set(int pos, bool valor);          // Acessa o elemento do topo
B.flip(int pos, bool valor);         // Retira o elemento do topo
```

STL pair<T1,T2>

```
pair<T1,T2> P;
```

1. Pode ser interpretado como uma struct simples que relaciona dois valores com tipos possivelmente diferentes

```
pair<float,float> ponto;
ponto.first = 1;
ponto.second = 1;
```

```
// Exemplo de uso com #define
#define pair<float,float> ponto_t
#define x(p) p.first
#define y(p) p.second

ponto_t ponto;
x(ponto) = 1;
y(ponto) = 1;
```

STL Funções

```
#define all(u) u.begin(), u.end()

sort(all(TARGET));
find(all(TARGET), T item);
lower_bound(all(TARGET), T item);
upper_bound(all(TARGET), T item);
```

Complexidade

1. Os programas não devem exceder 10^9 operações
2. Orientação inicial quanto à entrada:
   - $10^3] \rightarrow O(n^3)$
   - $10^4] \rightarrow O(n^2)$
   - $10^6] \rightarrow O(n.log(n))$
   - $10^8] \rightarrow O(n)$
   - ...
3. É preciso contar com a complexidade das funções da própria linguagem

Fibonacci em Vetor URI#1176

```cpp
#include <bits/stdc++.h>
using namespace std;

#define _ ios_base::sync_with_stdio(false); \
      cin.tie(0); cout.tie(0);

#define endl '\n'
#define FOR(i,m,n) for(int i = m; i < n; i++)
#define MAXN 61

vector<long long> fib(MAXN, INT_MAX);
int n, t, c;

void calc_fib(){
   FOR(i,2,MAXN) fib[i] = fib[i-1] + fib[i-2];
}

int main(){ _
   fib[0] = 0;
   fib[1] = 1;
   calc_fib();

   cin >> t;
   while(t--) {
      cin >> c;
      cout << "Fib(" << c << ") = " << fib[c] << endl;
   }
   return 0;
}
```

Pontos de Feno URI#1261

```cpp
#include <bits/stdc++.h>
using namespace std;

map<string, unsigned> hay_point;
int m, n, v, sum;
string str_h;

int main(){ _
   cin >> m >> n;
   while(m--){
       cin >> str_h >> v;
       hay_point[str_h] = v;
   }
   while(n--){
       sum = 0;
       while(true) {
           getline(cin, str_h)
           if(str_h == ".") break;

           stringstream ss(str_h);
           while(ss >> str_h)
               sum += hay_point[str_h];
       }
       cout << sum << endl;
   }

   return 0;
}
```

Copa do mundo SPOJ#COPA1

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

using namespace std;

int main(){

int M,N;
char ta,tb;
queue<char> times;

for (char c = 'A'; c <= 'P'; c++)
      times.push(c);

   for (int i = 0; i < 15; i++){
      cin >> M >> N;
      ta = times.front();times.pop();
      tb = times.front();times.pop();
      M > N ? times.push(ta) : times.push(tb);
   }
   cout << times.front() << "\n";

   return 0;
}
```