

# Ayudantía 1

# ¿Que es programación competitiva?



La Programación Competitiva es un deporte mental, en el que los participantes se sumergen en la emoción de resolver problemas desafiantes con especificaciones concretas en un tiempo limitado.

# ¿De qué me sirve la programación competitiva?

La programación competitiva te ayuda a profundizar en temas avanzados de matemáticas y algoritmos, te brinda la habilidad de implementar soluciones complejas y entrena tu pensamiento lógico.



# ¿De qué me sirve en el futuro profesional?

Muchas grandes empresas requieren superar pruebas técnicas, que a menudo incluyen la resolución e implementación de algoritmos.

Además, las empresas buscan personas capaces de crear e implementar soluciones a problemas complejos, por lo que esta habilidad es fundamental.

# ¿Dónde puedo aprender más sobre programación

La página más activa y con una amplia cantidad de problemas, recursos y tutoriales es Codeforces. En esta plataforma se realizan competencias con frecuencia, y la comunidad crea tutoriales para aprender temas avanzados.

CODEFORCES

Sponsored by TON

HOME

TOP

CATALOG

CONTESTS

GYM

PROBLEMSET

GROUPS

RATING

EDU

API

CALENDAR

HELP

CharlesLakes

Logout

Current or upcoming contests

Name	Writers	Start	Length	
Codeforces Round (Div. 2)		Aug/25/2024 10:35UTC+4	02:00	Before start 5 days <a href="#">Register now</a> <a href="#">x6859</a> *has extra registration
Codeforces Round (Div. 2)		Sep/14/2024 10:35UTC+4	02:00	Before start 4 weeks Before registration 3 weeks

Past contests

Name	Writers	Start	Length	
Codeforces Round 967 (Div. 2)	Misaki	Aug/20/2024 10:35UTC+4	02:00	<a href="#">Final standings</a> <a href="#">x3868</a>
Educational Codeforces Round 169 (Rated for Div. 2)	BledDest Neon adedalic awoo	Aug/15/2024 10:35UTC+4	02:00	<a href="#">Final standings</a> <a href="#">x34832</a>
Codeforces Round 966 (Div. 3)	Gornak40 Vladimirov myav senjougahara	Aug/13/2024 10:40UTC+4	02:15	<a href="#">Final standings</a> <a href="#">x46389</a> Solved: 6 out of 8
EPIC Institute of Technology Round August 2024 (Div. 1 + Div. 2)	Flamie ledn xoyte	Aug/11/2024 10:35UTC+4	03:00	<a href="#">Final standings</a> <a href="#">x29359</a>
Codeforces Round 965 (Div. 2)	cry sageev343 sum	Aug/10/2024 10:35UTC+4	02:00	<a href="#">Final standings</a> <a href="#">x40031</a> Solved: 3 out of 6
Codeforces Round 964 (Div. 4)				

Pay attention

Before contest  
[Codeforces Round \(Div. 2\)](#)  
5 days  
[Register now](#)  
\*has extra registration

Past contests filter

Contest type:  
Any

Rated:  
Doesn't matter

Tried:  
Doesn't matter

Substring:  
In contest title and writers

Filter

Recent virt. contests

Contest	Time
Codeforces Round 967 (Div. 2)	Apr/11/2024

Algorithms

General

General discussions that cannot be attributed to a specific topic

The Ultimate Topic List (with Resources, Problems and Templates) - YouKnowWho - 3 years ago

General ideas - adamant - 8 years ago

A bit more of general ideas - adamant - 2 years ago

[Tutorial] Collection of little techniques - Is this it - 2 years ago

Binary and Ternary Searches

EDU: Binary Search - pashka - 4 years ago

[Tutorial] Binary search and other "halving" methods - nor - 3 years ago

Bin search and relative error - Um\_nik - 8 years ago

Multi-dimensional ternary search - adamant - 3 years ago

Parallel Binary Search [tutorial] - himanshujaju - 8 years ago

An alternative and very interesting approach on binary search - Pankin - 4 years ago

Binary Lifting, No Memory Wasted - Urbanowicz - 4 years ago  
Binary lifting with O(n) memory

Constructive Algorithms

Tips on Constructive Algorithms - RealNero - 4 years ago

6 weeks ago User Tom66 inserted blog post Montgomery/Barret reduction and NTT (Performance optimization) by alexvim: inserted field blogEntryId 129600, inserted field catalogFolder FFT and Similar Transformations, inserted field createTime Jul 12, 2024, 12:44:47 PM, inserted field deleted false, inserted field hidden false, inserted field position 1300.

6 weeks ago User Tom66 inserted blog post A tool for hacking rolling hashes with fixed modulus and bases by Sugar\_Ian: inserted field blogEntryId 129538, inserted field catalogFolder Hashing, inserted field createTime Jul 10, 2024, 3:26:39 PM, inserted field deleted false, inserted field hidden false, inserted field position 800.

6 weeks ago User Tom66 updated folder Testing and Polygon: updated field englishName Testing and Polygon.

6 weeks ago User Tom66 updated blog post [Tutorial] Boruvka's Algorithm by RockyB: updated field position 4300.

6 weeks ago User Tom66 updated blog post Algorithm Gym :: Graph Algorithms by PrinceOfPersia: updated field position 8400.

6 weeks ago User Tom66 updated blog post Codeforces Updates (April-May, 2015) by kuviman: undated field

Ayudantía 1 Algoritmos y Complejidad

5 / 27



# ¿Que competencias existen?



# ¿Que lenguajes de programación se usan?

## Top

1. C++
2. Python
3. Java

# ¿Por qué C++?

- Es un lenguaje rápido en comparación con Python u otros de alto nivel.
- La biblioteca estándar de C++ (STL) ofrece una amplia gama de funcionalidades y estructuras de datos.
- Permite programar soluciones eficientes y versátiles gracias a la STL.
- En mi experiencia con programación competitiva, nunca he necesitado usar punteros.



# Estructura básica de un programa

```
// Esta directiva incluye todas las bibliotecas estándar del lenguaje.  
// Aunque no es óptima para desarrollar software complejo,  
// en programación competitiva nos ahorra mucho tiempo.  
#include <bits/stdc++.h>  
  
using namespace std;  
  
int main() {  
    cout << "Hola mundo" << endl;  
  
    return 0;  
}
```

## ¿Cómo usar arreglos dinámicos sin punteros?

Podemos utilizar el contenedor `vector`, implementado en la STL de C++.

Se define de la siguiente manera:

```
vector<tipo_de_dato> nombre_de_variable;
```

# Estructuras de Datos - Vector

```
vector<int> arr1; // Arreglo vacío de ints
vector<int> arr2(10); // Arreglo inicializado con 10 elementos
vector<int> arr3(8, -1); // Arreglo inicializado con 8 elementos, asignando -1 a todas las posiciones
vector<vector<string>> arr5; // Vector que almacena vectores de string
```

```
arr1.push_back(10); // Inserta 10 al final en O(1)
arr2.pop_back(); // Elimina el último elemento en O(1)
arr2[3] = 15; // Asigna 15 en la posición 3
arr2.clear(); // Elimina todos los elementos del vector
cout << arr3.size(); // Devuelve el tamaño del vector
```

# Estructuras de Datos - Vector

```
// Recorrer el vector usando un bucle for tradicional
for(int i = 0; i < arr1.size(); i++){
    cout << arr1[i] << endl;
}
// Recorrer el vector usando un foreach
for(int numero : arr1){
    cout << numero << endl;
}
```

# Estructuras de Datos - Queue

Una `queue` (cola) es un contenedor que sigue el principio FIFO (First In, First Out).

Se define de la siguiente manera:

```
queue<tipo_de_dato> nombre_de_variable;
```



# Estructuras de Datos - Queue

```
queue<int> cola;
```

```
cola.push(20); // Inserta el valor 20 al final de la cola  
cout << cola.front() << endl; // Imprime el valor en el frente de la cola  
cola.pop(); // Elimina el elemento al frente de la cola  
cout << cola.size() << endl; // Devuelve la cantidad de elementos en la cola  
cout << (cola.empty() ? "Cola vacía" : "Cola no vacía") << endl; // Verifica si la cola está vacía
```

# Estructuras de Datos - Stack

Una `stack` (pila) es un contenedor que sigue el principio LIFO (Last In, First Out).

Se define de la siguiente manera:

```
stack<tipo_de_dato> nombre_de_variable;
```

# Estructuras de Datos - Stack

```
stack<int> pila;
```

```
pila.push(30); // Inserta el valor 30 en la pila
cout << pila.top() << endl; // Imprime el valor en el tope de la pila
pila.pop(); // Elimina el elemento en el tope de la pila

cout << pila.size() << endl; // Devuelve la cantidad de elementos en la pila
cout << (pila.empty() ? "Pila vacía" : "Pila no vacía") << endl; // Verifica si la pila está vacía
```

# Estructuras de Datos - Set

Un `set` es un contenedor que almacena elementos únicos en orden específico.

Se define de la siguiente manera:

```
set<tipo_de_dato> nombre_de_variable;
```

# Estructuras de Datos - Set

```
set<int> conjunto; // Orden de menor a mayor
set<int,greater<int>> alreves; // Orden de menor a mayor
```

```
conjunto.insert(15); // Inserta el valor 15 en el set
conjunto.insert(20); // Inserta el valor 20 en el set
conjunto.insert(10); // Inserta el valor 10 en el set
cout << conjunto.size() << endl; // Devuelve la cantidad de elementos en el set
cout << (conjunto.empty() ? "Set vacío" : "Set no vacío") << endl; // Verifica si el set está vacío
// Imprimir todos los elementos del set
for (int elemento : conjunto) {
    cout << elemento << " "; // Imprime cada elemento
}
cout << endl;
// Eliminar un elemento
conjunto.erase(20); // Elimina el elemento 20 del set
```



# Estructuras de Datos - Map

Un `map` es un contenedor que almacena pares clave-valor ordenados por las claves.

Se define de la siguiente manera:

```
map<tipo_de_clave, tipo_de_valor> nombre_de_variable;
```

# Estructuras de Datos - Map

```
map<string, int> mapa;
```

```
mapa["manzanas"] = 10; // Asigna el valor 10 a la clave "manzanas"
mapa["naranjas"] = 5; // Asigna el valor 5 a la clave "naranjas"

cout << mapa["manzanas"] << endl; // Imprime el valor asociado a la clave "manzanas"
mapa.erase("naranjas"); // Elimina el par clave-valor con la clave "naranjas"

cout << mapa.size() << endl; // Devuelve la cantidad de elementos en el map
cout << (mapa.empty() ? "Mapa vacío" : "Mapa no vacío") << endl; // Verifica si el map está vacío

// Iteración sobre el map usando un foreach
for (auto par : mapa) {
    cout << "Clave: " << par.first << ", Valor: " << par.second << endl; // Imprime cada clave y su valor asociado
}
```

# Otros Tipos de Datos - Pair

Un `pair` es una estructura que puede contener dos valores de diferentes tipos. Se usa comúnmente para almacenar pares de datos relacionados, como coordenadas o claves y valores.

Definición y uso:

```
pair<tipo_de_dato1, tipo_de_dato2> nombre_de_variable;  
pair<int, string> par;  
  
// Inicializar un par  
par = make_pair(1, "uno");  
  
// Acceso a elementos  
cout << par.first << endl; // Imprime el primer elemento (1)  
cout << par.second << endl; // Imprime el segundo elemento ("uno")
```

# Otros Tipos de Datos - Tuple

Una `tuple` es similar a un `pair`, pero puede contener múltiples valores de diferentes tipos. Esto es útil cuando necesitas agrupar más de dos valores.

Definición y uso:

```
#include <tuple>
tuple<tipo_de_dato1, tipo_de_dato2, tipo_de_dato3> nombre_de_variable;
tuple<int, string, double> tupla;

// Inicializar una tupla
tupla = make_tuple(1, "uno", 3.14);

// Acceso a elementos
cout << get<0>(tupla) << endl; // Imprime el primer elemento (1)
cout << get<1>(tupla) << endl; // Imprime el segundo elemento ("uno")
cout << get<2>(tupla) << endl; // Imprime el tercer elemento (3.14)
```

# Algoritmos - Sort

La función `sort` es utilizada para ordenar elementos en un contenedor. Esta función es parte de la librería `<algorithm>` y proporciona ordenamientos rápidos y eficientes  $O(n \log(n))$ .



# Algoritmos - Sort

## Ordenar en Orden Ascendente

```
vector<int> numeros = {5, 2, 9, 1, 5, 6};

// Ordenar el vector en orden ascendente
sort(numeros.begin(), numeros.end());

// Imprimir el vector ordenado
cout << "Vector ordenado: ";
for (int num : numeros) {
    cout << num << " ";
}
cout << endl;
```

# Algoritmos - Sort

## Ordenar en Orden Descendente

Para ordenar en orden descendente, puedes utilizar la función `greater<int>()` como tercer argumento:

```
sort(numeros.begin(), numeros.end(), greater<int>());

cout << "Vector ordenado en orden descendente: ";
for (int num : numeros) {
    cout << num << " ";
}
cout << endl;
```

## Ordenar con una Función de Comparación Personalizada

Puedes definir una función de comparación personalizada para ordenar elementos de acuerdo con un criterio específico. A continuación, se muestra un ejemplo de cómo ordenar un vector de pares de enteros según el segundo elemento de cada par:

# Algoritmos - Sort

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

bool compararPorSegundo(pair<int, int> a, pair<int, int> b) {
    return a.second < b.second;
}

int main() {
    vector<pair<int, int>> pares = {{1, 4}, {2, 2}, {3, 3}};
    sort(pares.begin(), pares.end(), compararPorSegundo);
    cout << "Vector de pares ordenado por el segundo elemento: ";
    for (pair<int, int> par : pares) {
        cout << "(" << par.first << ", " << par.second << ") ";
    }
    cout << endl;
    return 0;
}
```