Tabla con Tipos de Datos Básicos y Funciones útiles de C++

Tipos de Datos Básicos

Nombre del tipo	Otros nombres	Intervalo de valores
`int`	`signed`	**De -2.147.483.648 a 2.147.483.647**
`long long`	**Ninguno (equivalente a `__int64`)**	**De -9.223.372.036.854.775.808 a 9.223.372.036.854.775.807**
`float`	**None**	**3.4E +/- 38 (aprox. siete dígitos)**
`double`	**None**	**1.7E +/- 308 (aprox. 15 dígitos)**
`char`	**None**	**De 0 a 255 según corresponda**
`bool`	**None**	`false` **o** `true`

Algunas Funciones y Comandos Útiles

Nombre	Función	Ejemplo
Potencia	pow()	X^2 = pow(x, 2)
Valor Absoluto	abs()	|x| = abs(x)
Logaritmo natural	log(x)	ln(x) = log(x)
Logaritmo base 10	log10(x)	$\log_{10}$ x = log10(x)
Ingreso de String con espacios (**Cuando la entrada tiene espacios)**	getline()	string frase; getline(cin, frase)
Raíz cuadrada	sqrt()	$\sqrt{x}$ = sqrt(x)
Raíz cúbica	cbrt()	$\sqrt[3]{x}$ = cbrt(x)
Ordenar Vector	sort()	sort(nombreVector.begin(), nombreVector.end())

Datos útiles

Datos	Valor
π	3.141592
e	2.7182818

Comando y métodos útiles C++

Uso de setprecision:

Se utiliza para determinar la cantidad de decimales de una variable que permita ese tipo de datos.

Ejemplo para precisión con tres decimales:

cout << fixed << setprecision(3) << variable << endl;

Fórmula para la Suma de números consecutivos naturales:

Suma = (n*(n+1)) / 2

Vectores Dinámicos

1. Declaración e inicialización

Ejemplos:

Con tamaño n	Sin tamaño Los ingresos de datos se deben realizar con push_back()
int n; cin >> n; vector<int> v(n);	vector<string> v;

2. Métodos más usados

La siguiente tabla resume los métodos y operaciones más frecuentes al trabajar con vectores.

Método	Descripción	Ejemplo
`size()`	Cantidad de elementos	`v.size()`
`empty()`	true si está vacío	`v.empty()`

Método	Descripción	Ejemplo
`push_back(x)`	Agrega x al final	`v.push_back(x);`
`pop_back()`	Elimina el último elemento	`v.pop_back();`
`front()`	Referencia al primer elemento	`v.front()`
`back()`	Referencia al último elemento	`v.back()`
`v[i]`	Acceso por índice (sin chequeo)	`v[0] = 5;`
`clear()`	Elimina todos los elementos	`v.clear();`
`resize(n)`	Cambia el tamaño a n	`v.resize(10);`
`insert(it, x)`	Inserta x en la posición del iterador	`v.insert(v.begin()+2, 99);`
`erase(it)`	Elimina el elemento en esa posición	`v.erase(v.begin()+1);`
`begin() / end()`	Iteradores al inicio / fin	`for (auto it = v.begin();...)`
`capacity()`	Capacidad reservada (≥ size)	`v.capacity()`

3. Recorridos típicos

3.1 Por índice

Cuando se trabaja con un tamaño **n:**

```
for (int i = 0; i < n; i++) {
    cout << v[i] << " ";
}
```

Cuando se trabaja con el límite del vector:

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

3.2 Por rango (recomendado si no se necesita el índice)

```
for (int x : v) {
    cout << x << " ";
}
```

3.3 Por referencia (para modificar los elementos)

```
for (int &x : v) {
    x *= 2;
}
```

4. Ejemplo integrador

Programa que lee N números, los guarda en un vector y muestra el mayor:

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

int main() {
    int n;
    cin >> n;

    vector<int> v;
    for (int i = 0; i < n; i++) {
        int x;
        cin >> x;
        v.push_back(x);
    }

    int mayor = v[0];
    for (int x : v) {
        if (x > mayor) mayor = x;
    }

    cout << mayor << endl;
    return 0;
}
```