



UNIVERSIDAD NACIONAL DE ASUNCIÓN
**FACULTAD DE
INGENIERÍA**

Vectores (Arreglos unidimensionales)

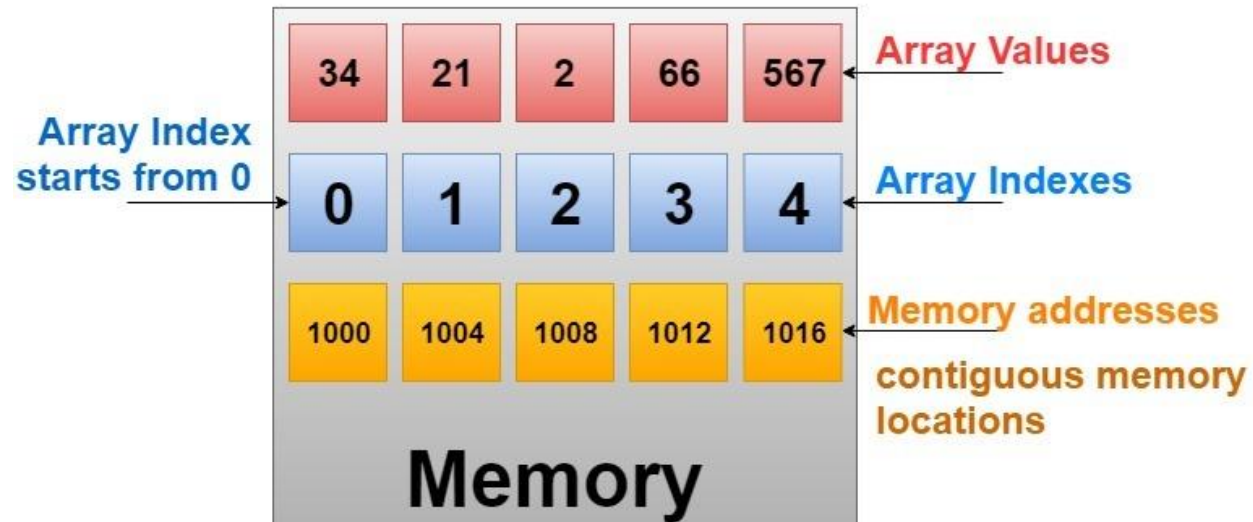
Cátedra de Computación

• **2020** •

¿Qué veremos?

- Necesidad de contar con estructuras de datos
- Vectores (arreglos unidimensionales)
 - Declaración
 - Lectura
 - Impresión
- Ejercicios

```
int x[5] = {34, 21, 2, 66, 567}
```



Necesidad de estructuras de datos

Las variables simples únicamente pueden almacenar un dato. En ejercicios anteriores, hemos trabajado con listas de números (calificaciones de alumnos de una misma clase, trabajadores de una empresa, etc.); y su manejo con variables simples puede ser un poco impráctico, además de no considerar el almacenamiento de las entradas.

```
int a = 5;  
double b = 4;  
char c = 'a';  
float d = 3.2;
```

Pregunta 1: ¿Cómo podríamos ordenar N enteros utilizando variables simples?

$\{5, 1, 8, 7, 3, 4, \dots\}$

Pregunta 2: ¿Cómo calcularíamos la cantidad de alumnos que obtuvieron una nota inferior al promedio del curso en cierta materia?

Estructuras de datos

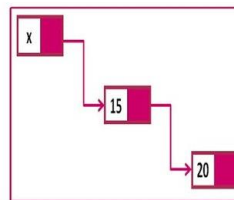
Para salvar estas situaciones, la mayoría de los lenguajes de programación incluyen características de **estructuras de datos**. Una estructura de datos es una colección de datos que pueden ser caracterizados por su organización y las operaciones que se definen en ella (*acceso, inserción, borrado*).

Las estructuras de datos se dividen en:

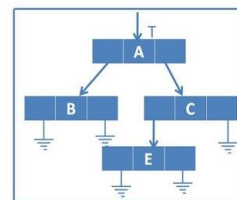
- **Estáticas** (tamaño definido de antemano): arreglos, registros.
- **Dinámicas** (no tiene limitaciones de tamaño): listas(pilas/colas), listas enlazadas, árboles, grafos.



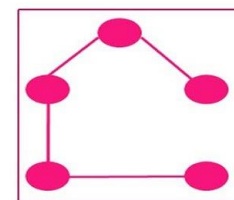
Sorting



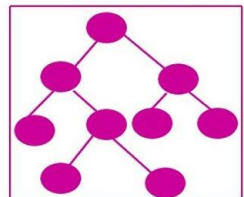
Link list



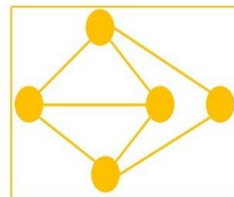
list



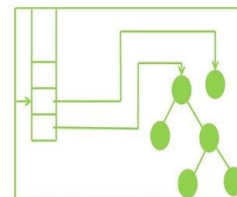
spanning tree



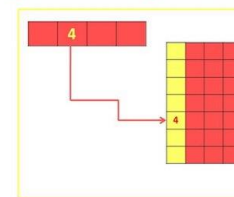
Tree



Graph

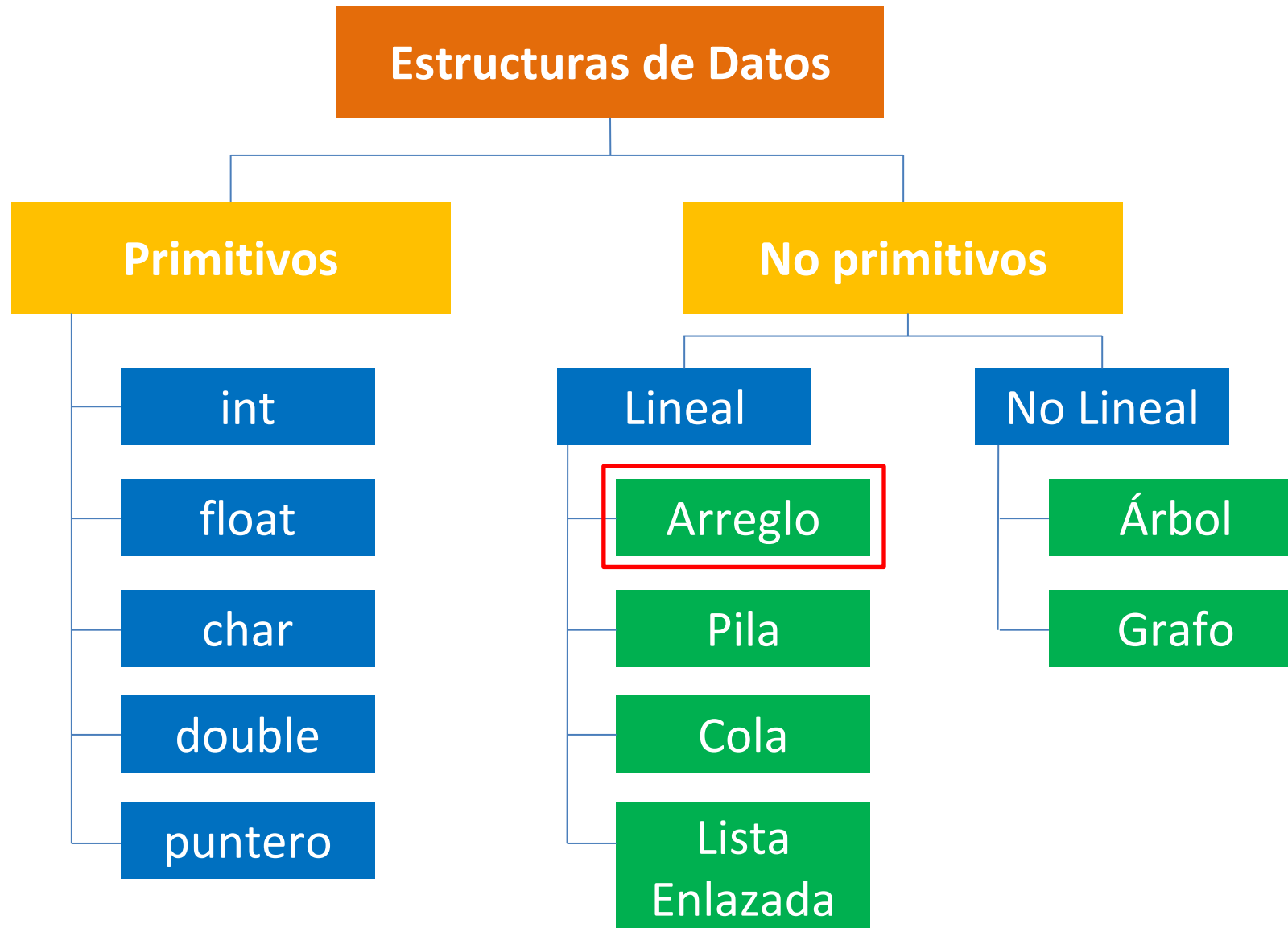


Stack



Hashing

Estructuras de datos



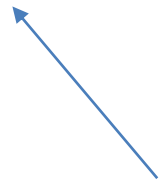
Arreglos

Las **estructuras de datos básicas** que soportan la mayoría de los lenguajes son los **arreglos** (arrays) (siendo el **vector** un arreglo de una dimensión, y la **matriz** uno de dos dimensiones).

Un **arreglo** es una secuencia de posiciones de la memoria central a las que se puede acceder directamente, que contiene **datos del mismo tipo** y pueden ser seleccionados individualmente mediante el uso de *índices*.

Ejemplo de vector: nota (notas de una clase de n alumnos)

nota 0	nota 1	nota 2	...	nota i	...	nota $n-1$
--------	--------	--------	-----	----------	-----	------------

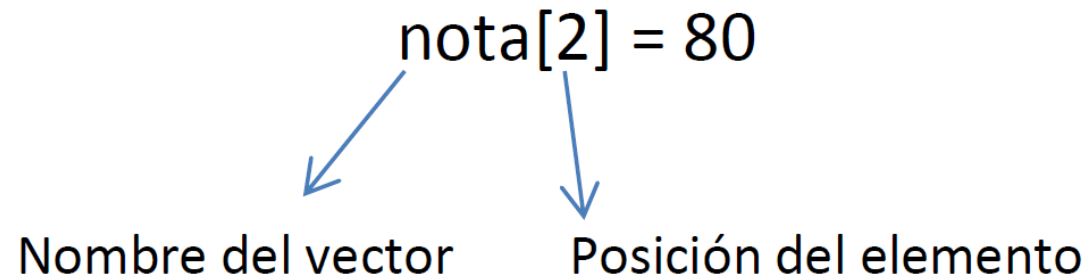


nota[2]

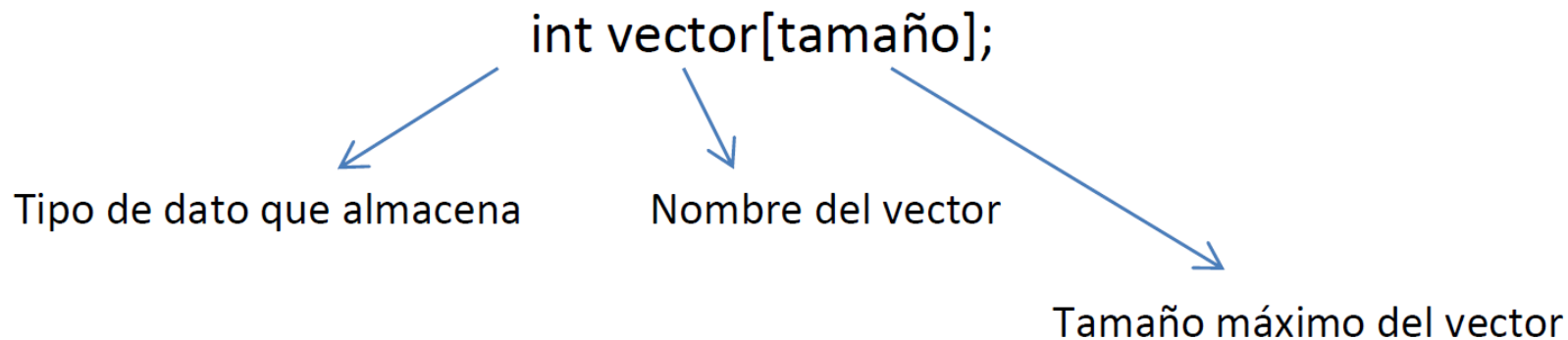
Observación importante: en C/C++, el primer elemento se representa por el índice 0. Si el tamaño del vector es n , entonces el último elemento tiene un índice $n - 1$.

Arreglos unidimensionales o vectores

El vector `nota` tiene subíndices o índices de sus elementos $(0, 1, 2, \dots, i, \dots, n-1)$ que indican la posición de un elemento particular dentro del arreglo (de tamaño n). Por ejemplo, si se desea modificar el tercer elemento de un vector de tipo entero:



La declaración de un vector en C++ se hace de la siguiente manera:



Vectores – Declaración

Declaración de vectores

```
tipo_dato nombre_vector[tam];
```

Es muy similar a la declaración de las variables simples, pero el tamaño del vector va en corchetes luego del nombre del mismo. El tamaño tam debe tener valores asignados previamente (en el caso de que sean variables), o bien ser constantes.

Si los elementos del vector están definidos, puede hacerse la siguiente declaración:

```
int a[5]={1,2,3,4,5};
```


Vectores – Lectura e impresión

```
#include <iostream>
using namespace std;
int main(){
    int n;
    cout<<"Ingrese la cantidad de elementos: ";
    cin>>n;
    int a[n]; //Se declara el vector
    int i;
    for(i=0;i<n;i++){ //Se carga el vector por teclado
        cout<<"Ingrese a["<<i<<"]: ";
        cin>>a[i];
    }
    cout<<"El vector es: "<<endl;
    for(i=0;i<n;i++) cout<<a[i]<<"\t"; //Impresión
    return 0;
}
```

Ejemplo 1

Calcular la cantidad de alumnos que obtuvieron nota inferior al promedio del curso en cierta materia. Hay N alumnos (donde N es un valor entero positivo ingresado por teclado), y todos rindieron. Las notas son números enteros que van del 0 al 100 (se asume que todas las notas son correctas).

¿Cuáles son los pasos para resolverlo?

Ejemplo 1

```
#include <iostream>
using namespace std;
int main(){
```

```
    int N;
    cout<<"Ingrese la cantidad de alumnos: ";
    cin>>N; //Se ingresa el tamaño del vector
    int notas[N]; //Se declara el vector
```

Declaración del vector de tamaño N

```
    //Se carga el vector
```

```
    int i;
    for(i=0;i<N;i++){
        cout<<"Ingrese notas["<<i<<"]: ";
        cin>>notas[i];
    }
```

Lectura de notas

```
    //Se calcula el promedio
```

```
    int suma=0;
    for(i=0;i<N;i++) suma+=notas[i];
    float prom = 1.0*suma/N;
```

Cálculo del promedio

```
    //Se cuentan las notas menores que el promedio
```

```
    int menores = 0;
    for(i=0;i<N;i++){
        if(prom>notas[i]) menores++;
    }
```

Número de notas menores al promedio

```
    cout<<"El promedio es: "<<prom<<endl;
    cout<<"La cantidad de alumnos con nota inferior al promedio es: "<<menores<<endl;
    return 0;
```

Resultados

```
}
```


Ejemplo 2

Se tienen anotadas las temperaturas (promedio) de todos los días del mes de febrero de 2020 y se deben almacenar en el vector `tempFeb`. Diseñar un programa que obtenga las temperaturas máxima, mínima (e indique los días correspondientes), y el promedio de las que se encuentran entre los días 21 y 27.

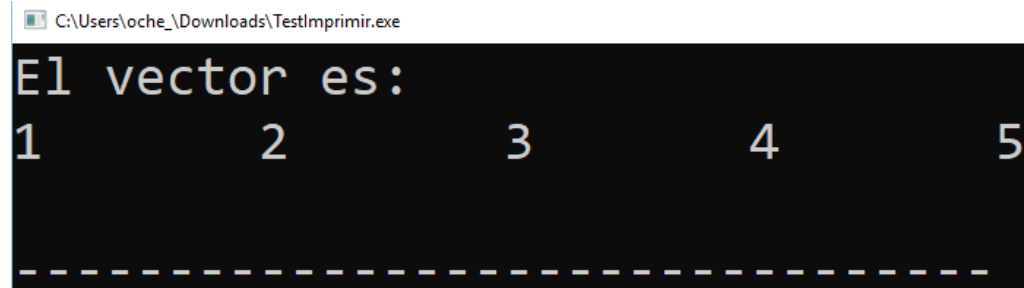
Vectores y funciones

Se puede pasar un vector como argumento de una función. Un ejemplo es el siguiente:

```
void imprimirVector(int x[], int n){  
    int i;  
    cout<<"El vector es: "<<endl;  
    for(i=0;i<n;i++) cout<<x[i]<<"\t";  
    cout<<endl;  
}
```

No colocar el tamaño aquí*

```
int main(){  
    int a[]={1,2,3,4,5};  
    imprimirVector(a,5);  
    return 0;  
}
```



```
C:\Users\oche_\Downloads\TestImprimir.exe  
El vector es:  
1      2      3      4      5  
-----
```

*Si el valor es una constante entera no dará error, pero el compilador lo ignorará. Si es una variable, dará error.

Ejemplos 3 y 4

- Diseñar un programa que calcule la magnitud de un vector A de N elementos (números reales). Este programa debe contar con una función **magnitud** que reciba un vector y su cantidad de elementos, y devuelva la magnitud del vector.
- Diseñar un programa que calcule el producto escalar de dos vectores A y B de N elementos (números reales). Este programa debe contar con una función **escalar** que reciba dos vectores (y el tamaño de ambos) y devuelva el producto escalar.

Recordar: Paso por valor

Función main

```
int main () {  
    int a,b;  
    cout<<"Ingrese a: "; cin>>a;  
    cout<<"Ingrese b: "; cin>>b;  
    int c = suma(a,b);  
    cout<<"La suma es: "<<c<<endl;  
    system("pause");  
    return 0;  
}
```

Función suma

```
int suma(int x, int y){  
    int c;  
    c = x+y;  
    return c;  
}
```

Memoria



Vectores y funciones

Cuando un vector es el argumento de una función, se pasa la **referencia*** del vector al llamar a dicha función. Por ello, si se hacen cambios dentro de la función, éstos se reflejarán en el vector original.

```
void sumar10(int x[], int n){  
    int i;  
    for(i=0;i<n;i++) x[i]+=10;  
}
```

```
int main(){  
    int a[]={1,2,3,4,5};  
    sumar10(a,5);  
    imprimirVector(a,5);  
    return 0;  
}
```

C:\Users\oche_\Downloads\TestSumar.exe

```
El vector es:  
11      12      13      14      15  
-----
```

* Se proporcionará mayores detalles en la clase sobre punteros.

Ejemplo 5

Escribir una función que reciba un vector de enteros y su tamaño, y que pase cada elemento a la posición derecha. El elemento final del vector pasa al inicio.

1	2	3	4	5	6	7	8	9
---	---	---	---	---	---	---	---	---



9	1	2	3	4	5	6	7	8
---	---	---	---	---	---	---	---	---

Tarea

Se tiene un vector de números binarios de tamaño n (siendo el mismo un múltiplo de 3). Un ejemplo es el siguiente:

1	0	1	0	0	0	1	0	0
---	---	---	---	---	---	---	---	---

Diseñar un algoritmo que cree nuevo vector a partir del vector de entrada, donde después de cada 3 elementos del vector original, se agregue un elemento que indique la cantidad de 1's en esos tres elementos. En nuestro caso, la salida sería:

1	0	1	2	0	0	0	0	1	0	0	1
---	---	---	---	---	---	---	---	---	---	---	---