

Algoritmos y Estructuras de Datos

Clase de Práctica 12 - Arreglos unidimensionales

Comisión A y F

Docentes de práctica:

Auxiliar Tomás Assenza

Auxiliar Facundo Sanchez

Universidad Tecnológica Nacional - Facultad Regional Santa Fe

Definición de arreglo

"Un arreglo es un grupo de posiciones de memorias contiguas que tienen el mismo nombre y tipo"

Tipos de arreglos

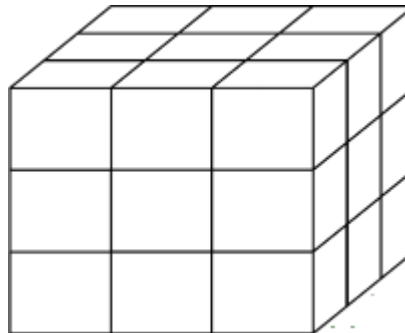
- Unidimensionales: arreglos de una dimensión, llamados vectores o listas.

a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]
------	------	------	------	------	------	------

- Bidimensionales: arreglos de dos dimensiones, conocidos como matrices.

a[0][0]	a[0][1]	a[0][2]	a[0][3]	a[0][4]	a[0][5]	a[0][6]
a[1][0]	a[1][1]	a[1][2]	a[1][3]	a[1][4]	a[1][5]	a[1][6]
a[2][0]	a[2][1]	a[2][2]	a[2][3]	a[2][4]	a[2][5]	a[2][6]

- Multidimensional: los demás tipos son llamados multidimensionales.



Tipos de tamaño




- Tamaño físico: es el tamaño máximo de elementos que el arreglo puede almacenar



Tamaño físico: 7




Declaración y acceso

- Declaración

<pre>int main() { int tam=12; int c[tam]; ... }</pre> 	<pre>int main() { int c[12]; ... }</pre> 	<pre>#define TAM 12 int main() { int c[TAM]; ... }</pre> 
---	---	--

Declaración y acceso

- Declaración

<pre>int main() { int tam=12; int c[tam]; ... }</pre> 	<pre>int main() { int c[12]; ... }</pre> 	<pre>#define TAM 12 int main() { int c[TAM]; ... }</pre> 
---	--	--

- Acceso

a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]
13	45	12	1	5		

- Para referirse a un elemento hay que especificar:
 - El nombre del arreglo
 - El número de posición.
- El primer elemento está en la posición 0: a[0] -> 13
- El segundo elemento está en la posición 1: a[1] -> 45
- El tercer elemento está en la posición 2: a[2] -> 12

Ejercicios

Array fill I - Beecrowd 1173

Descripción

Leer un número y hacer un programa que lo ponga en la primera posición de un arreglo $N[10]$. En cada posición subsecuente, se debe colocar el doble de lo que tiene la posición anterior. Por ejemplo, si el ingreso es 1, el arreglo tendrá los números 1,2,4,8, y los que correspondan cómo siguientes.

Entrada:

La entrada contiene un número entero V ($V < 50$)

Salida:

Imprimir el número guardado en cada posición del arreglo en la forma " $N[i] = X$ ", donde i es la posición del arreglo y x es el número guardado en la posición i . El primer número X es V .

Ejercicio en Beecrowd: <https://judge.beecrowd.com/es/problems/view/1173>

Solucion

```
#include <iostream>
#define TAM 10
using namespace std;

void inicializar_teclado(int [], int);
void imprimir_vector(int [], int);

int main(int argc, char *argv[]) {
    int vector[TAM];
    inicializar_teclado(vector,TAM);
    imprimir_vector(vector,TAM);
    return 0;
}

void inicializar_teclado(int V[], int tam){
    cin>> V[0];
    for(int i=1; i<tam;i++)
        V[i]= 2*V[i-1];
}

void imprimir_vector(int V[], int tam){
    for(int i=0; i<tam;i++)
        cout<<"N["<<i<<"]"<<" = "<<V[i]<<endl;
}
```


Ejercicios

Frecuencia de Los Números I - Beecrowd 1171

Descripción

En este problema su trabajo será leer algunos números enteros positivos e imprimir cuántas veces aparece cada número en la entrada. Deberá escribir cada valor diferente que aparezca en la entrada, ordenados ascendentemente.

Entrada:

La entrada consiste de un único caso de prueba. La primera línea contiene un entero N , que indica la cantidad de números enteros X ($1 \leq X \leq 2000$) que se deberán leer. Cada número no aparecerá más de 20 veces en la entrada.

Salida:

Imprima la salida como se muestra en el caso de ejemplo, indicando cuántas veces aparece cada número en la entrada, ascendentemente.

Ejercicio en Beecrowd: <https://judge.beecrowd.com/es/problems/view/1171>

```
#include <iostream>
#define TAMFRECUENCIAS 20000
using namespace std;
void ingresaElementos(int [], int);
void imprimeFrecuencias(int []);

int main(int argc, char *argv[]) {
    int frecuencias[TAMFRECUENCIAS] {};
    int n;

    cin >> n;

    ingresoElementos(frecuencias, n);
    imprimeFrecuencias(frecuencias);

    return 0;
}
void ingresaElementos(int frecuencias[], int n) {
    for (int i = 0; i < n; i++) {
        int ingreso;
        cin >> ingreso;
        frecuencias[ingreso]++;
    }
}
void imprimeFrecuencias(int frecuencias[]) {
    for (int i = 0; i < TAMFRECUENCIAS; i++) {
        if (frecuencias[i] != 0)
            cout << i << " aparece " << frecuencias[i] << " vez(es)" << endl;
    }
}
```

Tipos de tamaño

- Tamaño físico: es el tamaño máximo de elementos que el arreglo puede almacenar.



Tamaño físico: 7

- Tamaño lógico: es la cantidad de elementos que tiene almacenado el arreglo.



Tamaño físico: 7

Tamaño lógico: 5

Ejercicios

1) Declarar un arreglo de **N** componentes numéricos **enteros**. Luego, implementar variantes para la **inicialización** de sus componentes:

- Asignando **valores** iniciales por extensión en la declaración.
- Definiendo la **función** `void inicializar_teclado(int V[], int tam)` que carga los valores a través de la entrada estándar (teclado).
- Definiendo la **función** `void inicializar_aleatorio(int V[], int tam)` que carga los valores a través de funciones de generación de números aleatorios haciendo uso de las funciones `srand(time(NULL))` y `rand()`

Ejercicios

1) Declarar un arreglo de N componentes numéricos **enteros**. Luego, implementar variantes para la **inicialización** de sus componentes:

- Asignando **valores** iniciales por extensión en la declaración.

```
#include <iostream>  
using namespace std;  
#define N 1000
```

Definimos N como un tamaño muy grande, ya que será el tamaño físico del arreglo, y no sabemos cuántos componentes contendrá.

Ejercicios

1) Declarar un arreglo de **N** componentes numéricos **enteros**. Luego, implementar variantes para la **inicialización** de sus componentes:

- Asignando **valores** iniciales por extensión en la declaración.

```
#include <iostream>
using namespace std;
#define N 1000

int main() {
    int vector[N] {12, 13, 14, 15};
}
```

- *¿Cual será el tamaño físico de este arreglo? ¿Y el lógico?*
- *¿Que pasa con los elementos a la derecha de los que declaramos (vector[5], vector[6], ...)? ¿Que pasa si inicializamos el vector como vector[N] {}, vector[N] = {} o vector[N] = {0}?*

1) Declarar un arreglo de **N** componentes numéricos **enteros**. Luego, implementar variantes para la **inicialización** de sus componentes:

- Definiendo la **función** `void inicializar_teclado(int V[], int tam)` que carga los valores a través de la entrada estándar (teclado).

```
#include <iostream>
using namespace std;
#define N 1000

void inicializar_teclado(int V[], int tam);

int main(int argc, char *argv[]) {
    int vector[1000];
    int tamañoLogico;
    cout << "Ingrese la cantidad de elementos que tiene su arreglo" << endl;
    cin >> tamañoLogico;

    inicializar_teclado(vector, tamañoLogico);
    return 0;
}

void inicializar_teclado(int V[], int tam) {
    for (int i = 0; i < tam; i++) {
        int ingreso;
        cout << "Ingrese el valor " << i + 1 << endl;
        cin >> ingreso;
        V[i] = ingreso;
    }
}
```

1) Declarar un arreglo de **N** componentes numéricos **enteros**. Luego, implementar variantes para la **inicialización** de sus componentes:

- Definiendo la **función** `void inicializar_aleatorio(int V[], int tam)` que...

```
#include <iostream>
#include <ctime>
using namespace std;
void inicializar_aleatorio(int V[], int tam);
int main(int argc, char *argv[]) {
    int vector[1000];
    int tamañoLogico;
    cout << "Ingrese la cantidad de elementos que tiene su arreglo" << endl;
    cin >> tamañoLogico;

    inicializar_aleatorio(vector, tamañoLogico);
    return 0;
}
void inicializar_aleatorio(int V[], int tam) {
    srand(time(NULL));
    for (int i = 0; i < tam; i++) {
        int ingreso = rand();
        V[i] = ingreso;
    }
}
```

¿Como podríamos modificar la función para que tenga un tope inferior y superior?

Ejercicios

10) Se leen 10 valores enteros menores que 30. Luego se leen valores enteros positivos hasta que la suma de los dígitos de uno de los valores leídos, sea igual a alguno de los 10 valores inicialmente leídos. Informar el número que cumplió esta condición.

Ejemplo:

- Valores iniciales: 1 6 25 18 23 2 6 19 14 13
- i. 9245
- ii. 340
- iii. 694 Imprimir: 694

10) Se leen 10 valores enteros menores que 30. Luego se leen valores enteros positivos hasta que la suma de los dígitos de uno de los valores leídos, sea igual a alguno de los 10 valores inicialmente leídos. Informar el número que cumplió esta condición.

```
#include <iostream>
#define TAMANIO 10
using namespace std;

void inicializaArreglo(int [], int);

bool existeNumero(int [], int, int);

int sumaDigitos(int);
```

```
int main(int argc, char *argv[]) {  
    int numerosIniciales[TAMANIO], ingreso;  
    inicializaArreglo(numerosIniciales, TAMANIO);  
  
    do {  
        cout << "Ingrese un numero" << endl;  
        cin >> ingreso;  
    } while(!existeNumero(numerosIniciales, TAMANIO, sumaDigitos(ingreso)));  
  
    cout << "El numero que cumple la condicion es: " << ingreso << endl;  
    return 0;  
}  
  
void inicializaArreglo(int V[], int tam) {  
    for (int i = 0; i < tam; i++) {  
        int ingreso;  
        cout << "Ingrese el valor " << i + 1 << endl;  
        cin >> ingreso;  
        V[i] = ingreso;  
    }  
}
```

```
bool existeNumero(int V[], int tam, int numero) {  
    bool encontrado = false;  
    int i = 0;  
    while (i < tam and !encontrado) {  
        if (V[i] == numero) encontrado = true;  
        i++;  
    }  
    return encontrado;  
}
```

```
int sumaDigitos(int numero) {  
    if (numero < 10) return numero;  
    return numero % 10 + sumaDigitos(numero/10);  
}
```

Actividad grupal propuesta

Ejercicio 13 de la guía -> Enviar a Tomás Assenza

Ejercicio 11 de la guía -> Enviar a Facundo Sanchez

Equipo de Algoritmos y Estructuras de Datos

Comisión A y F