Presentación de la materia
Algoritmos y Estructuras de Datos I - AED I

AED I

2020

UBP UNIVERSIDAD BLAS PASCAL

docentes

Gustavo Echeverria
gustavo.echeverria@live.com

horarios

miércoles
módulos 3, 4, 5 y 6
18.45 a 20:15 20:30 a 22:00
lugar: LA5

jueves
módulos 3, 4, 5 y 6
18.45 a 20:15 20:30 a 22:00
lugar:6

régimen de regularidad

Exámenes

- 2 parciales
- 2 recuperatorios

Trabajos Prácticos

- 3 trabajos Prácticos

calendario de evaluaciones

1° Parcial: 30/04/2020

2° Parcial: 11/06/2020

Recuperatorio 1° Parcial: 18/06/2020

Recuperatorio 2° Parcial: 18/06/2020

17/06 feriado

objetivo de la materia

- Seleccionar un algoritmo o estructura de datos de búsqueda u ordenamiento bajo diferentes criterios.
- Aplicar un algoritmo o estructura de datos de búsqueda u ordenamiento en la construcción de un programa.
- Codificar y testear una estructuctura de datos dinámica
- Codificar y testear un algoritmo recursivo.
- Comprender los tipos de datos abstractos.
- Construir un programa nuevo con módulos, o descomponer uno existente en módulos.
- Testear un módulo.
- Corregir una descomposición modular.

contenido

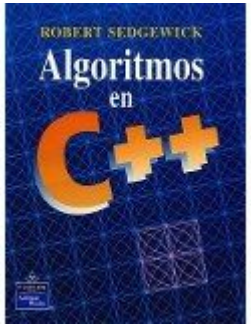
1. Programación Orientada a Objetos
2. Estructuras de Datos Básicas
3. Búsqueda
4. Programación Dinámica
5. Interfaces Gráficas de Usuario
6. Laboratorio

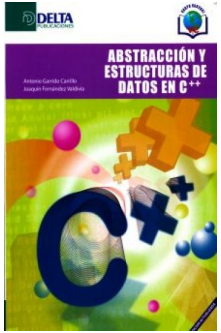
metodología

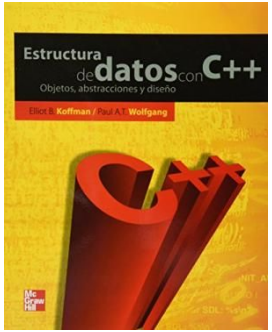
La totalidad de las clases son teórico prácticas en laboratorio

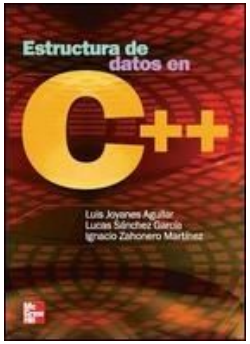
Las evaluaciones se llevarán a cabo en el laboratorio y utilizando las máquinas del laboratorio

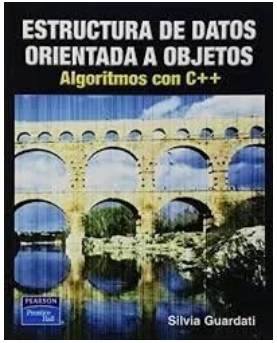
bibliografía en español











- **Algoritmos en C++**, Robert Sedgewick. Addison Wesley
- **Abstracción y Estructuras de Datos en C++**, Garrido Carrillo Antonio y otros. Delta Publicaciones
- **Estructura de Datos en C++**, Koffman Elliot y otros. Mc Graw Hill
- **Estructuras de datos en C++**, Luis Joyanes Aguilar y otros. Mc Graw Hill
- **Estructura de Datos Orientada a Objetos. Algoritmos con C++**, Silvia Guardatti. Pearson

Repaso

AED I

2020

UBP UNIVERSIDAD BLAS PASCAL

Ejercicio 0.01.

Bisiesto

Escribí un programa que solicite un año como entrada y diga si ese año es o no un año bisiesto. Teniendo en cuenta que:

- Un año divisible por 400 es bisiesto
- Un año divisible por 4 es bisiesto siempre que no sea divisible por 100

Ejercicio 0.01.

```
int main(){
    int anio;
    std::cout<<"Ingrese el año: ";
    std::cin>>anio;

    if((anio % 400 == 0) || (anio % 100 != 0 && anio % 4 == 0))
        std::cout << "El año: " << anio << " es bisiesto" << std::endl;
    else
        std::cout << "El año: "<< anio << " no es bisiesto" << std::endl;

    return 0;
}
```

Ejercicio 0.02.

Vocales

Escribí un programa que determine si el caracter ingresado por consola es o no una vocal.

Preferentemente utilizá una estructura `switch` en la solución

Ejercicio 0.02.

```
int main(){
    char caracter;
    std::cout << "Ingrese un caracter: ";
    std::cin >> caracter;

    if ((caracter >= 'A' && caracter <= 'Z') || (caracter >= 'a' && caracter <= 'z')) {
        switch (caracter){
            case 'a':
            ...
            case 'u':
                std::cout << caracter << " es una vocal minúscula." << std::endl;
                break;
            case 'A':
            ...
            case 'U':
                std::cout << caracter << " es una vocal mayúscula." << std::endl;
                break;
            default:
                std::cout << caracter << " es consonante." << std::endl;
                break;
        }
    }
    else
        std::cout << caracter << " no es una letra." << std::endl;
    return 0;
}
```

Ejercicio 0.03.

Cantidad de caracteres

Escribí un programa que informe la cantidad de cada uno de los caracteres que posee la cadena en minúsculas que se ingresa por consola

```
Ingrese una cadena en minúsculas:
esta es una cadena de prueba
La cadena tiene:
5 a
1 b
1 c
2 d
5 e
2 n
1 p
1 r
2 s
1 t
2 u
```

Ejercicio 0.03.

```
#define N 50
#define TAM_ALFABETO 26
int main(){
    char cadena[N];
    int contadores[TAM_ALFABETO];
    for (int i = 0; i < TAM_ALFABETO; i++)
        contadores[i] = 0;
    std::cout << "Ingrese una cadena en minúsculas: " << std::endl;
    std::cin.getline(cadena,N);
    for (int i = 0; i < N; i++){
        if (cadena[i] >= 97 && cadena[i] <= 122)
            contadores[cadena[i] - 97]++;
    }
    std::cout << "La cadena tiene:" << std::endl;
    for (int i = 0; i < TAM_ALFABETO; i++)
        if (contadores[i])
            std::cout << contadores[i] << " " << (char)(i+97) << std::endl;
}
```

Ejercicio 0.04.

Valores Repetidos

Escribí un programa que permita ingresar un array de 10 elementos e informe cuáles son los valores repetidos en el array y en qué posiciones se encuentran esas ocurrencias

```
Ingrese valor[1]: 12
Ingrese valor[2]: 34
Ingrese valor[3]: 30
Ingrese valor[4]: 21
Ingrese valor[5]: 16
Ingrese valor[6]: 34
Ingrese valor[7]: 10
Ingrese valor[8]: 40
Ingrese valor[9]: 30
Ingrese valor[10]: 11
El valor 34 está repetido en las posiciones 2 y 6
El valor 30 está repetido en las posiciones 3 y 9
```

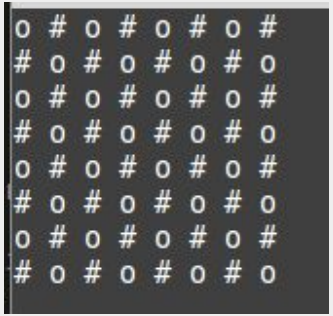
Ejercicio 0.04.

```
#define N 10
int main(){
    int valores[N];
    for (int i = 0; i < N; i++){
        std::cout << "Ingrese valor[" << i+1 << "]: ";
        std::cin >> valores[i];
    }
    for (int i = 0; i < 10; i++){
        for (int j = i+1; j < 10; j++){
            if (valores[i] == valores[j]){
                std::cout << "El valor " << valores[i] << " está repetido en las posiciones "
                << i+1 << " y " << j+1 << std::endl;
            }
        }
    }
}
```

Ejercicio 0.05.

Patrones

Escribí un programa que muestre en la consola un patrón similar al de un tablero de ajedrez



Ejercicio 0.05.

```
#define N 8

int main(){

    for (int i = 0; i < N; i++){
        for (int j = 0; j < N; j++){
            std::cout << (((i + j) %2) ? "# " : "o ");
        }
        std::cout << std::endl;
    }
    return 0;
}
```

Ejercicio 0.05. (alternativa)

```
#define N 8

int main(){

    for (int i = 0; i < N; i++){
        for (int j = 0; j < N; j++){
            std::cout << ((i==0||i==N-1||j==0||j==N-1) ? "#" : " ");
        }
        std::cout << std::endl;
    }
    return 0;
}
```

Ejercicio 0.06.

Factorial

Escriba un programa que calcule el factorial de un número

*El **factorial** de un entero positivo n, el **factorial** de **n** o **n factorial** se define en principio como el producto de todos los números enteros positivos desde 1 (es decir, los números naturales) hasta **n**. El **factorial** de 0 es 1*

$$0! = 1$$

$$n! = n \times (n-1)!$$

Ejercicio 0.06.

```
int factorial(int n){
    if (n==0) return 1;
    else return n * factorial(n-1);
}

int main(){
    for (int i = 0; i < 10; i++){
        std::cout<<"Factorial de " << i << " = " << factorial(i) << std::endl;
    }
}
```