

Estructuras de Datos y Algoritmos

Programación en Python y C++

Rafael De Luna Loredo

Facultad de Ciencias, UASLP



Indice

① Introducción

② Operaciones

③ Condicionales

④ Ciclos

⑤ Arreglos y
Matrices

⑥ Funciones

⑦ Apuntadores

⑧ Estructuras de
Datos

⑨ Objetos

⑩ Ordenamiento

⑪ Búsqueda

⑫ Grafos

⑬ Cadenas

⑭ Referencias

1 Introducción

Conceptos

Básicos

Tipos de Datos

2 Operaciones

3 Condicionales

4 Ciclos

5 Arreglos y Matrices

6 Funciones

7 Apuntadores

8 Estructuras de Datos

9 Objetos

10 Ordenamiento

11 Búsqueda

12 Grafos

13 Cadenas

14 Referencias

¿Qué es un algoritmo?

Es una serie de pasos a seguir para solucionar un problema. Pero no solo eso, incluso en nuestra vida diaria usamos algoritmos sin saberlo, por ejemplo al ponernos los zapatos o vestirnos seguimos una serie de pasos para llegar a un resultado final

¿Como programar algoritmos?

Antes de empezar a tirar código hay que hacernos unas cuántas preguntas

- ¿sigue una serie de pasos?
 - ¿son consecutivos?
 - ¿que resultados puedo esperar?

¿Que lenguajes de programación usar?

Podemos hacer uso de cualquier lenguaje, todo dependerá de que tan cómodos nos sintamos con el lenguaje que vayamos a usar o estemos utilizando.

En este curso veremos los ejemplos en Python y C++, C++ por ser el más utilizado en programación competitiva y Python por su sintaxis sencilla y su amplia utilización en la industria.

1 Introducción Conceptos Básicos

Bibliotecas,
cabeceras,
espacio de
nombres

Tipos de Datos

2 Operaciones

3 Condicionales

4 Ciclos

5 Arreglos y Matrices

6 Funciones

7 Apuntadores

8 Estructuras de Datos

9 Objetos

10 Ordenamiento

11 Búsqueda

12 Grafos

13 Cadenas

14 Referencias

1 Introducción

Conceptos Básicos

Bibliotecas,
cabeceras,
espacio de
nombres

Tipos de Datos

2 Operaciones

3 Condicionales

4 Ciclos

5 Arreglos y Matrices

6 Funciones

7 Apuntadores

8 Estructuras de Datos

9 Objetos

10 Ordenamiento

11 Búsqueda

12 Grafos

13 Cadenas

14 Referencias

Cabeceras

Son archivos que contienen las declaraciones de funciones y/o clases por lo que suelen tener código

Bibliotecas

Una biblioteca o librería podríamos considerarla una colección de cabeceras, además de cabeceras incluye archivos de enlazado dinámico o estático.

Espacio de nombres

Es un contenedor donde existen una o más identificadores para clases, funciones ó métodos contenidos en las cabeceras.

1 Introducción

Conceptos
Básicos

Tipos de Datos
Númericos
Caracteres
Lógicos
Contenedores y
colecciones
Dinámicos

2 Operaciones

3 Condicionales

4 Ciclos

5 Arreglos y Matrices

6 Funciones

7 Apuntadores

8 Estructuras de Datos

9 Objetos

10 Ordenamiento

11 Búsqueda

12 Grafos

13 Cadenas

14 Referencias

Tipos de Datos

- Nos ayudan a representar datos o valores del mundo real
- Pueden tener números, palabras o simbolos
- Algunos pueden tener múltiples elementos
- En Python son clases

1 Introducción

Conceptos
Básicos

Tipos de Datos
Númericos
Caracteres
Lógicos
Contenedores y
colecciones
Dinámicos

2 Operaciones

3 Condicionales

4 Ciclos

5 Arreglos y Matrices

6 Funciones

7 Apuntadores

8 Estructuras de Datos

9 Objetos

10 Ordenamiento

11 Búsqueda

12 Grafos

13 Cadenas

14 Referencias

Enteros

- No aceptan decimales
- Pueden ser negativos o positivos
- En C++ ocupan 4 bytes de memoria y un valor máximo de ± 2147483647
- En Python varía el tamaño en memoria pero puede ser desde 28 bytes hasta 408 bytes o más

Reales

- Aceptan decimales
- Pueden ser negativos o positivos
- En C++ ocupan 4 bytes de memoria y valores entre $1.17549e-38$ y $3.40282e+38$
- En Python varía el tamaño en memoria pero puede ser desde 24 bytes hasta 408 bytes o más

Reales de doble precisión

- Aceptan decimales
- Pueden ser negativos o positivos
- En C++ ocupan 8 bytes de memoria y valores entre $2.22507e-308$ y $1.79769e+308$
- En Python el tipo float hace una implementación a bajo nivel del tipo **double** de C.

1 Introducción

Conceptos
Básicos

Tipos de Datos
Númericos
Caracteres
Lógicos
Contenedores y
colecciones
Dinámicos

2 Operaciones

3 Condicionales

4 Ciclos

5 Arreglos y Matrices

6 Funciones

7 Apuntadores

8 Estructuras de Datos

9 Objetos

10 Ordenamiento

11 Búsqueda

12 Grafos

13 Cadenas

14 Referencias

Caracter

- Representan letras, simbolos o caracteres
- En C++ ocupan 1 byte de memoria y valores de ± 127
- En Python como tal no existe.

Cadena de caracteres

- Representan letras, simbolos, caracteres y/o palabras
- Tanto en C++ como en Python el tamaño varía según el tamaño de la cadena
- En el caso de C++ existen dos tipos, un arreglo de tipo **char** y el tipo **str** a través de la biblioteca **cstring**

1 Introducción

Conceptos
Básicos

Tipos de Datos
Númericos
Caracteres
Lógicos
Contenedores y
colecciones
Dinámicos

2 Operaciones

3 Condicionales

9 Objetos

4 Ciclos

10 Ordenamiento

5 Arreglos y Matrices

11 Búsqueda

6 Funciones

12 Grafos

7 Apuntadores

13 Cadenas

8 Estructuras de Datos

14 Referencias

Booleanos

- Representan solamente **True** o **False**, **1** o **0**
- Ocupan 1 byte de almacenamiento, por lo que se desperdicia mucho espacio de memoria

1 Introducción

Conceptos
Básicos

Tipos de Datos

Númericos
Caracteres
Lógicos
Contenedores y
colecciones
Dinámicos

2 Operaciones

3 Condicionales

4 Ciclos

5 Arreglos y Matrices

6 Funciones

7 Apuntadores

8 Estructuras de Datos

9 Objetos

10 Ordenamiento

11 Búsqueda

12 Grafos

13 Cadenas

14 Referencias

Diccionarios

- Se componen de una llave o clave y un valor
- Son elementos ordenados
- Pueden existir múltiples llaves
- Las llaves no se pueden repetir
- En C++ se llaman mapas, para usarlos hay que importar la cabecera *map*

Listas

- Son una secuencia de elementos
- En Python pueden contener múltiples tipos de datos
- Para usarlos en C++ hay que importar la cabecera *list*

Tuplas

- Son una secuencia de elementos
- Pueden contener múltiples tipos de datos
- En Python son inmutables, es decir no se pueden modificar o eliminar elementos
- Para usarlos en C++ hay que importar la cabecera *tuple*

Conjuntos

- Son una secuencia de elementos no repetidos
- Hacen alusión a la definición matemática de conjuntos
- En C++ existen dos tipos, ordenados y no ordenados
- Para usarlos en C++ hay que importar las cabeceras *set* y *unordered_set* respectivamente

1 Introducción

Conceptos
Básicos

Tipos de Datos
Númericos
Caracteres
Lógicos
Contenedores y
colecciones
Dinámicos

2 Operaciones

3 Condicionales

4 Ciclos

5 Arreglos y Matrices

6 Funciones

7 Apuntadores

8 Estructuras de Datos

9 Objetos

10 Ordenamiento

11 Búsqueda

12 Grafos

13 Cadenas

14 Referencias

Tipo vacío o sin tipo

- Están mas orientados al uso con apuntadores
- Pueden convertirse en cualquier tipo de dato
- No aplica en Python

1 Introducción

2 Operaciones

3 Condicionales

4 Ciclos

5 Arreglos y Matrices

6 Funciones

7 Apuntadores

8 Estructuras de Datos

9 Objetos

10 Ordenamiento

11 Búsqueda

12 Grafos

13 Cadenas

14 Referencias

Operaciones Matemáticas

- Tienen orden de precedencia, siendo el mismo que conocemos en matemáticas
- Van de izquierda a derecha
- Se realizan en pares
- Se pueden realizar entre diferentes tipos, todo dependerá de si se guardan o no en una variable
- Son las mismas que en matemáticas, suma, resta, multiplicación, división y módulo o residuo
- Para uso de funciones más avanzadas habrá que hacer uso de cabeceras o bibliotecas creadas para dicho propósito

Operaciones Lógicas

- Devuelven un valor booleano
- Van de izquierda a derecha
- Se realizan en pares
- Son algunas que ya conocemos en matemáticas, mayor que, menor que, igual a, diferente de, menor o igual, mayor o igual, conjunción y disyunción
- La conjunción y disyunción dependen de las otras

1 Introducción

2 Operaciones

3 Condicionales

if else elif

Switch

Operador

ternario

4 Ciclos

5 Arreglos y Matrices

6 Funciones

7 Apuntadores

8 Estructuras de Datos

9 Objetos

10 Ordenamiento

11 Búsqueda

12 Grafos

13 Cadenas

14 Referencias

Introducción

- Son un tipo de estructura
- Nos ayudan a tener control según valores o resultados esperados
- Evalúan operaciones lógicas
- Pueden contener o concatenar múltiples estructuras

1 Introducción

2 Operaciones

3 Condicionales

if else elif

Switch

Operador
ternario

4 Ciclos

5 Arreglos y Matrices

6 Funciones

7 Apuntadores

8 Estructuras de Datos

9 Objetos

10 Ordenamiento

11 Búsqueda

12 Grafos

13 Cadenas

14 Referencias

else

Es la contraparte de **if**, en caso de que la condición de **if** no se cumpla, ejecuta el código que contiene

elif

Es una combinación de `else` con `if`, hace una segunda evaluación de otro posible resultado esperado

1 Introducción

2 Operaciones

3 Condicionales

if else elif

Switch

Operador
ternario

4 Ciclos

5 Arreglos y Matrices

6 Funciones

7 Apuntadores

8 Estructuras de Datos

9 Objetos

10 Ordenamiento

11 Búsqueda

12 Grafos

13 Cadenas

14 Referencias

Switch

- En Python no existe pero se puede hacer con diccionarios o encadenando múltiples `if else`
- Es una estructura que evalúa múltiples casos posibles en base al posible valor de una variable
- También puede contener otras estructuras

1 Introducción

2 Operaciones

3 Condicionales

if else elif

Switch

Operador
ternario

4 Ciclos

5 Arreglos y Matrices

6 Funciones

7 Apuntadores

8 Estructuras de Datos

9 Objetos

10 Ordenamiento

11 Búsqueda

12 Grafos

13 Cadenas

14 Referencias

Operador ternario

- El funcionamiento es como el de un **if**, en Python de hecho es un **if**
- Sirve para comparaciones simples, donde no necesitamos ejecutar mucho código

① Introducción

② Operaciones

③ Condicionales

④ Ciclos

⑤ Arreglos y Matrices

⑥ Funciones

⑦ Apuntadores

⑧ Estructuras de Datos

⑨ Objetos

⑩ Ordenamiento

⑪ Búsqueda

⑫ Grafos

⑬ Cadenas

⑭ Referencias

Introducción

- Son un tipo de estructura
- Nos ayudan a ejecutar un código múltiples veces
- Ayudan a reducir código

for

- Se define un contador
- El contador cambia sin necesidad de declarar dicho cambio
- Hay que definir un tope o límite
- Hay que definir como avanza el contador

while

- Se ejecuta mientras dada una condición, dicha condición exista o se cumpla
- No se define un límite, mientras la condición exista, se seguirá ejecutando
- Necesitamos crear una manera de salir, por si la condición sigue existiendo
- Podría no ejecutarse si la condición no existe antes de hacer la evaluación

do while

- Es muy parecido a **while** con la diferencia de que primero ejecuta y después comprueba si dicha condición existe
- No se define un límite, mientras la condición exista, se seguirá ejecutando
- Necesitamos crear una manera de salir, por si la condición sigue existiendo
- Se ejecuta al menos una vez
- En Python no existe

1 Introducción

2 Operaciones

3 Condicionales

4 Ciclos

5 Arreglos y Matrices

Arreglos Matrices

6 Funciones

7 Apuntadores

8 Estructuras de Datos

9 Objetos

10 Ordenamiento

11 Búsqueda

12 Grafos

13 Cadenas

14 Referencias

1 Introducción

2 Operaciones

3 Condicionales

4 Ciclos

5 Arreglos y
Matrices

Arreglos
Métodos
Matrices

6 Funciones

7 Apuntadores

8 Estructuras de
Datos

9 Objetos

10 Ordenamiento

11 Búsqueda

12 Grafos

13 Cadenas

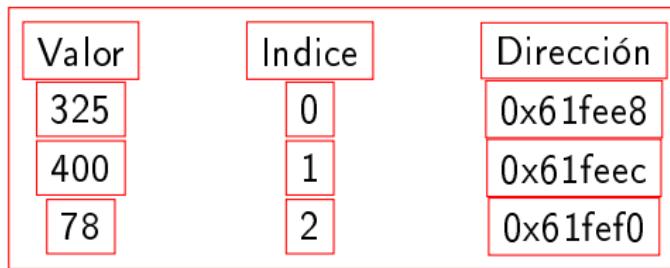
14 Referencias

Introducción

- Son de un solo tipo
- Puede contener múltiples valores
- Es unidimensional
- Su espacio en memoria varía según el tipo de dato y los valores que pueda tener
- En Python como tal no existen, se usan las listas en su lugar
- El índice siempre empieza en cero, puede variar en algunos lenguajes, pero suele ser una regla general

Almacenamiento en memoria

- El valor nos indica el contenido del arreglo en determinada posición
- El índice nos indica la posición en el arreglo
- La dirección de memoria donde se encuentra almacenado dicho valor



1 Introducción**2** Operaciones**3** Condicionales**4** Ciclos**5** Arreglos y
Matrices**Arreglos**
Métodos

Matrices

6 Funciones**7** Apuntadores**8** Estructuras de
Datos**9** Objetos**10** Ordenamiento**11** Búsqueda**12** Grafos**13** Cadenas**14** Referencias

Insertar en C++

- Hay que conocer el índice donde queremos el valor
- No es necesario insertarlos en orden
- No se puede tener un índice mayor al tamaño del arreglo

Insertar en Python

- Las listas en Python tienen un método para agregar valores
- No es necesario conocer el índice
- No es necesario conocer el tamaño

Modificar

- Hay que conocer el índice
- No se puede tener un índice mayor al tamaño del arreglo

Eliminar

- En C++ no se puede eliminar un elemento de un arreglo, solo se ignora el índice
- En Python existen 3 métodos ¹
 - Por el último índice
 - Por índice
 - Por valor

¹Si se elimina un elemento, el tamaño y los índices cambian

1 Introducción

2 Operaciones

3 Condicionales

4 Ciclos

5 Arreglos y Matrices

6 Funciones

7 Apuntadores

8 Estructuras de Datos

Arreglos Matrices Métodos

9 Objetos

10 Ordenamiento

11 Búsqueda

12 Grafos

13 Cadenas

14 Referencias

Introducción

- Son de un solo tipo
- Puede contener múltiples valores
- Son bidimensionales
- Su espacio en memoria varía según el tipo de dato y los valores que pueda tener
- En Python como tal no existen, se usan listas que contienen listas
- El índice siempre empieza en cero, puede variar en algunos lenguajes, pero suele ser una regla general
- Tienen doble índice, uno para las filas y otro para las columnas
- Las filas y columnas no tienen que ser del mismo tamaño

1 Introducción

2 Operaciones

3 Condicionales

4 Ciclos

5 Arreglos y
Matrices

Arreglos
Matrices
Métodos

6 Funciones

7 Apuntadores

8 Estructuras de
Datos

9 Objetos

10 Ordenamiento

11 Búsqueda

12 Grafos

13 Cadenas

14 Referencias

Insertar en C++

- Hay que conocer ambos indices donde queremos agregar el valor
- No es necesario insertarlos en orden
- No se pueden tener indices mayores a las filas o columnas

Insertar en Python

- Hay que conocer los indices donde se ubican las listas
- No es necesario conocer el índice
- No es necesario conocer el tamaño

Modificar

- Hay que conocer el índice
- No se puede tener un índice mayor al tamaño del arreglo

Eliminar

- En C++ no se puede eliminar un elemento de una matriz, solo se ignora el índice
- En Python
 - Por el último índice
 - Por índice
 - Por valor

1 Introducción**2** Operaciones**3** Condicionales**4** Ciclos**5** Arreglos y
Matrices**6** Funciones

Introducción

Tipos de
ParámetrosFunciones
anónimasRecursividad
Macros**7** Apuntadores**8** Estructuras de
Datos**9** Objetos**10** Ordenamiento**11** Búsqueda**12** Grafos**13** Cadenas**14** Referencias

1 Introducción**2** Operaciones**3** Condicionales**4** Ciclos**5** Arreglos y
Matrices**6** Funciones

Introducción

Tipos de
ParámetrosFunciones
anónimasRecursividad
Macros**7** Apuntadores**8** Estructuras de
Datos**9** Objetos**10** Ordenamiento**11** Búsqueda**12** Grafos**13** Cadenas**14** Referencias

Introducción

- Es código que se encuentra fuera del código principal
- Nos ayudan a crear código mas legible, organizado y fácil de mantener
- Pueden ser de distintos tipos, correspondientes al tipo de datos
- Pueden retornar o no, un valor
- Ocupan o no Parámetros
- Se pueden usar con apuntadores

Sintaxis

Una función se compone de 4 partes

- El tipo de dato de la función ²
- El nombre de la función
- Los parámetros, si es que lleva
- El código que ejecuta

²En Python no aplica

1 Introducción**2** Operaciones**3** Condicionales**4** Ciclos**5** Arreglos y
Matrices**6** Funciones

Introducción

**Tipos de
Parámetros**Funciones
anónimasRecursividad
Macros**7** Apuntadores**8** Estructuras de
Datos**9** Objetos**10** Ordenamiento**11** Búsqueda**12** Grafos**13** Cadenas**14** Referencias

Por Referencia

A la función se le pasa la dirección de memoria del parámetro que solicita, es decir donde se encuentra alojado el valor de dicho parámetro por lo que podemos modificar dicho valor dentro de la función por lo que hay que tener cuidado cuando se use este tipo de parámetro ya que podríamos terminar con resultados inesperados

Por valor

A la función se le pasa solo el valor del parámetro que solicita, es decir se le pasa una copia de la variable que se le está pasando, aquí el valor del parámetro aunque lo modifiquemos en la función, en el código principal no va a cambiar

1 Introducción**2** Operaciones**3** Condicionales**4** Ciclos**5** Arreglos y
Matrices**6** Funciones

Introducción

Tipos de
ParámetrosFunciones
anónimasRecursividad
Macros**7** Apuntadores**8** Estructuras de
Datos**9** Objetos**10** Ordenamiento**11** Búsqueda**12** Grafos**13** Cadenas**14** Referencias

Funciones anónimas

- Son funciones sin nombre
- Se declaran in situ
- Pueden tener como parámetros todo el scope

1 Introducción**2** Operaciones**3** Condicionales**4** Ciclos**5** Arreglos y
Matrices**6** Funciones

Introducción

Tipos de
ParámetrosFunciones
anónimasRecursividad
Macros**7** Apuntadores**8** Estructuras de
Datos**9** Objetos**10** Ordenamiento**11** Búsqueda**12** Grafos**13** Cadenas**14** Referencias

Recursividad

- Es más un método
- Aplica para problemas que están definidos sobre si mismos (factorial y fibonacci)
- Las funciones se llaman a si mismas
- Tienen la desventaja de llegar a consumir un espacio considerable de memoria
- Pueden ser difíciles de implementar

1 Introducción**2** Operaciones**3** Condicionales**4** Ciclos**5** Arreglos y
Matrices**6** Funciones

Introducción

Tipos de
ParámetrosFunciones
anónimas

Recursividad

Macros

7 Apuntadores**8** Estructuras de
Datos**9** Objetos**10** Ordenamiento**11** Búsqueda**12** Grafos**13** Cadenas**14** Referencias

Macros

- Son directivas tratadas por el preprocesador
- No son variables ni datos
- Sustituyen lo que tengan en donde se haga la llamada a la macro
- Disminuyen los tiempos de ejecución

1 Introducción

2 Operaciones

3 Condicionales

4 Ciclos

5 Arreglos y
Matrices

6 Funciones

7 Apuntadores

Introducción
Operadores
Aritmética de
apuntadores
Arreglos y
apuntadores
Apuntadores a
funciones

8 Estructuras de Datos

9 Objetos

10 Ordenamiento

11 Búsqueda

12 Grafos

13 Cadenas

14 Referencias

① Introducción

② Operaciones

③ Condicionales

④ Ciclos

⑤ Arreglos y
Matrices

⑥ Funciones

7 Apuntadores

Introducción

Operadores

Aritmética de
apuntadores

Arreglos y
apuntadores

Apuntadores a
funciones

8 Estructuras de Datos

9 Objetos

⑩ Ordenamiento

⑪ Búsqueda

⑫ Grafos

⑬ Cadenas

⑭ Referencias

Introducción

- Hacen referencia a direcciones de memoria
- Pueden apuntar a direcciones de memorias de variables y funciones
- Su tipo de dato puede cambiar ³
- También pueden apuntar a arreglos o matrices

³solo cuando se declaran de tipo void

Ventajas

Debajo de muchos de los lenguajes actuales se hace uso de los apuntadores, sin embargo no permiten hacer un uso mas profundo de ellos por lo que podríamos terminar con código mas pesado o ineficiente, los apuntadores nos permiten:

- Administrar el uso de memoria
- Crear código mas eficiente y rápido
- Permiten evitar o disminuir el overhead ⁴
- Iterar eficientemente sobre estructuras de datos

⁴Exceso de tiempo de ejecución y memoria

Desventajas

Aunque los apuntadores son una poderosa herramienta también implican ciertos riesgos o desventajas

- Mala administración de la memoria
- Posible acceso a partes críticas del sistema
- Requiere una previa planeación sobre su implementación el código

1 Introducción

2 Operaciones

3 Condicionales

4 Ciclos

5 Arreglos y
Matrices

6 Funciones

7 Apuntadores

Introducción

Operadores

Aritmética de
apuntadores

Arreglos y
apuntadores

Apuntadores a
funciones

8 Estructuras de Datos

9 Objetos

10 Ordenamiento

11 Búsqueda

12 Grafos

13 Cadenas

14 Referencias

Operador de dirección &

Nos permite obtener la dirección de memoria de cualquier variable, sea un apuntador o no, hay que tener cuidado al usarlos, ya que podríamos terminar mostrando direcciones importantes del sistema

Operador de indirección *

Nos permite obtener el contenido de la dirección a la que el apuntador hace referencia, es decir nos muestra un valor o dato, aquí también hay que tener cierto cuidado o podríamos terminar mostrando valores importantes del sistema.

① Introducción

② Operaciones

③ Condicionales

④ Ciclos

⑤ Arreglos y
Matrices

⑥ Funciones

7 Apuntadores

Introducción

Operadores

Aritmética de
apuntadores

Arreglos y
apuntadores

Apuntadores a
funciones

8 Estructuras de Datos

9 Objetos

⑩ Ordenamiento

⑪ Búsqueda

⑫ Grafos

⑬ Cadenas

⑭ Referencias

Suma y resta

Podemos hacer sumas o restas en un apuntador como en cualquier otra variable sin embargo debemos considerar que en realidad lo que estariamos haciendo es cambiar la dirección de memoria a la que se apunta. En el caso de la suma estariamos avanzando hacia una dirección de memoria que se encuentre mas adelante, según el tipo de dato al que apunte se este apuntando. Para la resta estariamos retrociendo hacia una dirección de memoria que se encuentre mas atrás

① Introducción

② Operaciones

③ Condicionales

④ Ciclos

⑤ Arreglos y
Matrices

⑥ Funciones

7 Apuntadores

Introducción

Operadores

Aritmética de
apuntadores

Arreglos y
apuntadores

Apuntadores a
funciones

8 Estructuras de Datos

9 Objetos

⑩ Ordenamiento

⑪ Búsqueda

⑫ Grafos

⑬ Cadenas

⑭ Referencias

Relación entre arreglos y apuntadores

Dado que un arreglo es una variable en la que se reserva un n tamaño de memoria, mediante apuntadores podemos acceder a la memoria de los diferentes índices del arreglo, también mediante apuntadores podemos crear arreglos, la diferencia es que con apuntadores el espacio de memoria se reserva en tiempo de ejecución (memoria dinámica)

Matrices con apuntadores

En realidad una matriz con apuntadores, es un arreglo de apuntadores, donde cada apuntador hace referencia a una dirección de memoria, tal vez suene más complicado pero la creación y recorrido es de la misma manera que en una matriz, solo que aplicando aritmética de apuntadores.

Ventajas respecto a arreglos o matrices

- No es necesario conocer el tamaño
- Se reserva el tamaño exacto
- Se puede liberar la memoria en caso de ya no utilizarla
- Se puede redimensionar el tamaño

Desventajas respecto a arreglos o matrices

- A veces es complicado de implementar
- Para redimensionar el tamaño es necesario crear una copia
- No se pueden eliminar elementos intermedios
- Cuando se libera memoria se borra todo el arreglo o matriz
- Se tiene que volver a crear el espacio de memoria en caso de querer redimensionar

① Introducción

② Operaciones

③ Condicionales

④ Ciclos

⑤ Arreglos y
Matrices

⑥ Funciones

7 Apuntadores

Introducción

Operadores

Aritmética de
apuntadores

Arreglos y
apuntadores

Apuntadores a
funciones

8 Estructuras de Datos

9 Objetos

⑩ Ordenamiento

⑪ Búsqueda

⑫ Grafos

⑬ Cadenas

⑭ Referencias

Apuntadores a funciones

- Hay que saber el tipo de dato que retorna la función
- Hay que saber los parámetros de la función
- Pueden hacer referencia a cualquier función que cumpla con los primeros puntos
- Podemos crear código más limpio y legible
- En principio puede ser difícil de implementar

1 Introducción

2 Operaciones

3 Condicionales

4 Ciclos

5 Arreglos y
Matrices

6 Funciones

7 Apuntadores

8 Estructuras de
Datos

Introducción

Pilas

Colas

Listas simples

Listas dobles

9 Objetos

10 Ordenamiento

11 Búsqueda

12 Grafos

13 Cadenas

14 Referencias

1 Introducción

2 Operaciones

3 Condicionales

4 Ciclos

5 Arreglos y
Matrices

6 Funciones

7 Apuntadores

8 Estructuras de
Datos

Introducción

Pilas

Colas

Listas simples

Listas dobles

9 Objetos

10 Ordenamiento

11 Búsqueda

12 Grafos

13 Cadenas

14 Referencias

Introducción

- Son un grupo o colección de datos que contienen operaciones para poder manipularlos
- Pueden ser estáticos o dinámicos
- Nos ayudan a representar datos reales en información
- Pueden ser lineales o no lineales

1 Introducción

2 Operaciones

3 Condicionales

4 Ciclos

5 Arreglos y
Matrices

6 Funciones

7 Apuntadores

8 Estructuras de
Datos

Introducción

Pilas

Métodos

Colas

Listas simples

Listas dobles

9 Objetos

10 Ordenamiento

11 Búsqueda

12 Grafos

13 Cadenas

14 Referencias

Pilas

- Son una estructura de tipo LIFO (Last In First Out)
- Puede ser estática o dinámica
- Cuenta con 2 operaciones básicas
- Tiene un tope
- Si es estática hay que definir un tamaño máximo

1 Introducción

2 Operaciones

3 Condicionales

4 Ciclos

5 Arreglos y
Matrices

6 Funciones

7 Apuntadores

8 Estructuras de
Datos

Introducción

Pilas

Métodos

Colas

Listas simples

Listas dobles

9 Objetos

10 Ordenamiento

11 Búsqueda

12 Grafos

13 Cadenas

14 Referencias

Inserción

- Hay que comprobar que no este llena
- El tope es el nuevo elemento insertado

Eliminación

- Hay que comprobar que no este vacía
- El nuevo tope será el elemento debajo del que eliminamos

1 Introducción

2 Operaciones

3 Condicionales

4 Ciclos

5 Arreglos y
Matrices

6 Funciones

7 Apuntadores

8 Estructuras de
Datos

Introducción
Pilas

Colas

Métodos

Listas simples
Listas dobles

9 Objetos

10 Ordenamiento

11 Búsqueda

12 Grafos

13 Cadenas

14 Referencias

Colas

- Son una estructura de tipo FIFO (First In First Out)
- Puede ser estática o dinámica
- Cuenta con 2 operaciones básicas
- El final es el último elemento
- Tiene un inicio
- Si es estática hay que definir un tamaño máximo

1 Introducción

2 Operaciones

3 Condicionales

4 Ciclos

5 Arreglos y
Matrices

6 Funciones

7 Apuntadores

8 Estructuras de
Datos

Introducción
Pilas

Colas
Métodos

Listas simples
Listas dobles

9 Objetos

10 Ordenamiento

11 Búsqueda

12 Grafos

13 Cadenas

14 Referencias

Inserción

- Hay que comprobar que no este llena
- El tope es el nuevo elemento insertado

Eliminación

- Hay que comprobar que no este vacía
- El inicio es el elemento siguiente al que eliminamos

1 Introducción

2 Operaciones

3 Condicionales

4 Ciclos

5 Arreglos y
Matrices

6 Funciones

7 Apuntadores

8 Estructuras de
Datos

Introducción

Pilas

Colas

Listas simples

Métodos

Listas dobles

9 Objetos

10 Ordenamiento

11 Búsqueda

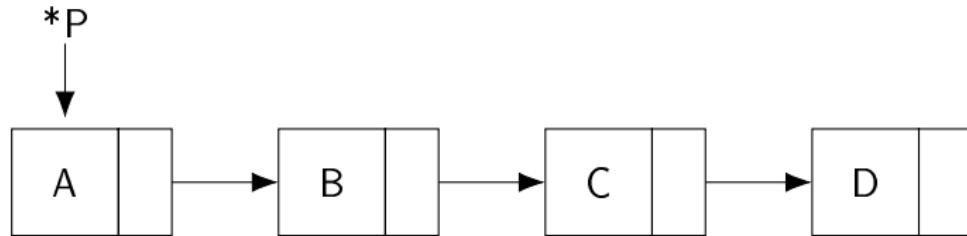
12 Grafos

13 Cadenas

14 Referencias

Listas simples

- Son una estructura lineal
- Es dinámica
- Son una colección de nodos
- Cada nodo es un puntero
- Cada nodo apunta al que le sigue



1 Introducción

2 Operaciones

3 Condicionales

4 Ciclos

5 Arreglos y
Matrices

6 Funciones

7 Apuntadores

8 Estructuras de
Datos

Introducción

Pilas

Colas

Listas simples

Métodos

Listas dobles

9 Objetos

10 Ordenamiento

11 Búsqueda

12 Grafos

13 Cadenas

14 Referencias

Inserción al inicio

- Si está vacía el nodo será nulo
- Hay que localizar el primer nodo
- Creamos un nuevo nodo y apuntara al primer nodo
- El nuevo nodo será el primer nodo

Inserción al final

- Hay que localizar el último nodo
- Creamos un nodo y el último nodo apuntara a este nuevo nodo
- El nuevo nodo apuntara a un valor nulo

Inserción entre nodos

- Hay que localizar los nodos entre los cuáles queremos hacer la inserción
- El nodo anterior apunta al nuevo nodo
- El nuevo nodo apunta al nodo siguiente

Eliminación al inicio

- Hacemos que el inicio de la lista apunte al siguiente elemento
- El inicio será el elemento siguiente al que eliminamos
- Si la lista queda vacía el nodo será nulo

Eliminación al final

- Localizamos el último
- Hacemos que el nodo anterior al último sea nulo
- Eliminamos el último nodo

Eliminación entre nodos

- Localizamos el nodo a eliminar
- Hacemos que el nodo anterior apunte al nodo siguiente del que queremos eliminar
- Eliminamos el nodo

1 Introducción

2 Operaciones

3 Condicionales

4 Ciclos

5 Arreglos y
Matrices

6 Funciones

7 Apuntadores

8 Estructuras de
Datos

Introducción

Pilas

Colas

Listas simples

Listas dobles

Métodos

9 Objetos

10 Ordenamiento

11 Búsqueda

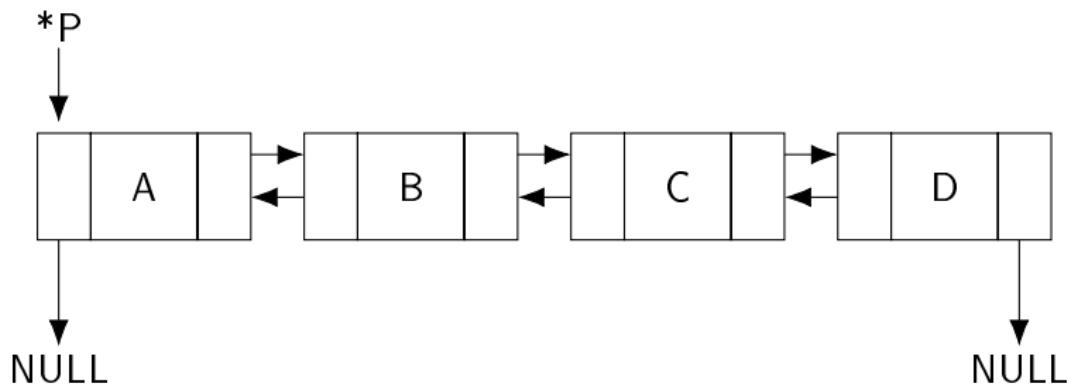
12 Grafos

13 Cadenas

14 Referencias

Listas dobles

- Tienen dos nodos
- Un nodo apunta al siguiente nodo
- El otro nodo apunta al nodo anterior



1 Introducción

2 Operaciones

3 Condicionales

4 Ciclos

5 Arreglos y
Matrices

6 Funciones

7 Apuntadores

8 Estructuras de
Datos

Introducción

Pilas

Colas

Listas simples

Listas dobles

Métodos

9 Objetos

10 Ordenamiento

11 Búsqueda

12 Grafos

13 Cadenas

14 Referencias

Inserción al inicio

- Si esta vacía, ambos nodos serán anulos
- Localizamos ini ⁵
- El nodo anterior de ini apuntara al nodo siguiente de ins⁶
- El nodo siguiente de ins apuntara al nodo anterior de ini
- EL nodo anterior de ins será nulo

⁵elemento inicial

⁶elemento a insertar

Inserción al final

- Localizamos fin⁷
- El nodo siguiente de fin apuntara al nodo anterior de ins⁸
- El nodo anterior de ins apuntara al nodo siguiente de fin
- EL nodo siguiente de ins será nulo

⁷elemento final

⁸elemento a insertar

Inserción entre nodos

- Hay que localizar los nodos entre los cuáles queremos hacer la inserción ⁹
- El nodo siguiente de izq apuntara a nodo anterior de ins ¹⁰
- El nodo anterior de ins apuntara a nodo siguiente de izq
- Nodo siguiente de ins apuntara a nodo anterior de der
- Nodo anterior de der apuntara a nodo siguiente de ins

⁹izq y der

¹⁰elemento a insertar

Eliminación al inicio

- Si quedará vacía la lista apuntará a nulo
- Nodo anterior de nwini¹¹ será nulo
- Eliminamos nodo siguiente de elim¹²

¹¹segundo elemento

¹²elemento a eliminar

Eliminación al final

- Localizamos fin
- Hacemos nodo siguiente de nwfin¹³ nulo
- Hacemos nulo al nodo anterior de fin

¹³elemento anterior a fin

Eliminación entre nodos

- Localizamos elim
- Nodo siguiente de der apunta a nodo anterior de izq
- Nodo anterior de izq apunte a izq

① Introducción

② Operaciones

③ Condicionales

④ Ciclos

⑤ Arreglos y
Matrices

⑥ Funciones

⑦ Apuntadores

⑧ Estructuras de
Datos

⑨ Objetos

Introducción

⑩ Ordenamiento

⑪ Búsqueda

⑫ Grafos

⑬ Cadenas

⑭ Referencias

1 Introducción

2 Operaciones

3 Condicionales

4 Ciclos

5 Arreglos y
Matrices

6 Funciones

7 Apuntadores

8 Estructuras de
Datos

9 Objetos
Introducción

10 Ordenamiento

11 Búsqueda

12 Grafos

13 Cadenas

14 Referencias

1 Introducción

2 Operaciones

3 Condicionales

4 Ciclos

5 Arreglos y Matrices

6 Funciones

7 Apuntadores

8 Estructuras de Datos

9 Objetos

10 Ordenamiento

Introducción

11 Búsqueda

12 Grafos

13 Cadenas

14 Referencias

① Introducción

② Operaciones

③ Condicionales

④ Ciclos

⑤ Arreglos y Matrices

⑥ Funciones

⑦ Apuntadores

⑧ Estructuras de Datos

⑨ Objetos

⑩ Ordenamiento Introducción

⑪ Búsqueda

⑫ Grafos

⑬ Cadenas

⑭ Referencias

1 Introducción

2 Operaciones

3 Condicionales

4 Ciclos

5 Arreglos y Matrices

6 Funciones

7 Apuntadores

8 Estructuras de Datos

9 Objetos

10 Ordenamiento

11 Búsqueda Introducción

12 Grafos

13 Cadenas

14 Referencias

1 Introducción**2** Operaciones**3** Condicionales**4** Ciclos**5** Arreglos y
Matrices**6** Funciones**7** Apuntadores**8** Estructuras de
Datos**9** Objetos**10** Ordenamiento**11** Búsqueda
Introducción**12** Grafos**13** Cadenas**14** Referencias

1 Introducción

2 Operaciones

3 Condicionales

4 Ciclos

5 Arreglos y Matrices

6 Funciones

7 Apuntadores

8 Estructuras de Datos

9 Objetos

10 Ordenamiento

11 Búsqueda

12 Grafos

Introducción

13 Cadenas

14 Referencias

1 Introducción**2** Operaciones**3** Condicionales**4** Ciclos**5** Arreglos y
Matrices**6** Funciones**7** Apuntadores**8** Estructuras de
Datos**9** Objetos**10** Ordenamiento**11** Búsqueda**12** Grafos
Introducción**13** Cadenas**14** Referencias

① Introducción

② Operaciones

③ Condicionales

④ Ciclos

⑤ Arreglos y Matrices

⑥ Funciones

⑦ Apuntadores

⑧ Estructuras de Datos

⑨ Objetos

⑩ Ordenamiento

⑪ Búsqueda

⑫ Grafos

⑬ Cadenas Introducción

⑭ Referencias

1 Introducción**2** Operaciones**3** Condicionales**4** Ciclos**5** Arreglos y Matrices**6** Funciones**7** Apuntadores**8** Estructuras de Datos**9** Objetos**10** Ordenamiento**11** Búsqueda**12** Grafos**13** Cadenas
Introducción**14** Referencias

1 Introducción

2 Operaciones

3 Condicionales

4 Ciclos

5 Arreglos y Matrices

6 Funciones

7 Apuntadores

8 Estructuras de Datos

9 Objetos

10 Ordenamiento

11 Búsqueda

12 Grafos

13 Cadenas

14 Referencias

Referencias |

- [1] Bhasin, H.
Python Basics, 3 ed.
David Pallai, Mercury Learning and Information, 2019.
- [2] Brassard, G., and Bratley, P.
Algorithmics: Theory and Practice, 31 ed.
Prentice Hall, 1988.
- [3] CAIRÓ, O., and Guardati, S.
Algorithms, 3 ed.
McGraw-Hill Interamericana, 2006.
- [4] Deitel, P., and Deitel, H.
C++ Como Programar, 9 ed.
Pearson Educación de México, 2014.

Referencias II

- [5] Downey, B., A.
Think Python, 2 ed.
O'Reilly, 2016.
- [6] Jaworski, M., and Ziadé, T.
Expert Python Programming, 3 ed.
Packt Publishing, 2019.
- [7] Laakmann, M., G.
Cracking the Coding Interview, 6 ed.
CareerCup, 2016.
- [8] Matthes, E.
Python Crash Course, 2 ed.
No Star Press, 2019.

Referencias III

- [9] Ramalho, L.
Fluent Python, 1 ed.
O'Reilly, 2014.
- [10] Reek, K.
Pointers On C.
Addison-Wesley Longman, 1997.
- [11] Sedgewick, R., and Wayne, K.
Algorithms, 4 ed.
Pearson Education, 2011.
- [12] Stroustrup, B.
The C++ Programming Language, 4 ed.
Pearson Education, 2013.

Referencias IV

- [13] Stroustrup, B.
Principles and Practice Using C++, 2 ed.
Pearson Education, 2014.
- [14] unknown.
Cplusplus.

Danke!