BÀI TẬP THỰC HÀNH

BỘ MÔN HỆ THỐNG THÔNG TIN

CẤU TRÚC DỮ LIỆU VÀ GIẢI THUẬT

THÁNG 4, 2024

TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP TP. HỒ CHÍ MINH

MỤC LỤC

BÀI TẬP TUẦN 1: ÔN TẬP PYTHON

Mục tiêu :

1. Ôn tập các khái niệm và dữ liệu cơ bản, các toán tử trong Python. Lệnh nhập xuất, các cấu trúc điều kiện, hàm, xử lý chuỗi, xử lý danh sách, xử lý tập tin và các thư viện của Python
2. Lập trình hướng đối tượng bằng Python

1) TOÁN TỬ - CÁC LỆNH CƠ BẢN – HÀM – TẬP TIN

1) Viết các lệnh xuất các biến sau:

```
# Kiểu dữ liệu số
x = 10 #Python hiểu đây là số nguyên
y = 10.5 #Python hiểu đây là số thực
z = 5 #Python hiểu đây là số nguyên
t = x + z #Python hiểu t là số nguyên
t = x + y #Python hiểu t là số thực
# Kết luận
# số nguyên (+,-,*,/) số nguyên = số nguyên
# số nguyên (+,-,*,/) số thực = số thực
# số thực (+,-,*,/) số thực = số thực

# Kiểu thực
a = 1.4
a = 31.4e-10
# Kiểu số phức: a +(-)ib
z = 2 + 7j
# Kiểu số dạng bát phân (8), thập lục phân (16) hoặc nhị phân (2)
x = 0b100100
y = 0xf400
# True và False
u = True # u -> 1
v = False # v -> 0
# Kiểu dữ liệu chuỗi
a = "Hello" # Python hiểu a là kiểu chuỗi
print(a)
b = 'World' # Python hiểu b là kiểu chuỗi
print(b)
# Sử dụng \n để xuống dòng
print("Python programming \n for Data Analysis")
# Sử dụng f để hiện thị giá trị
```

```
print(f'Giá trị biến a là: {a}')
print("Giá trị biến a là:",a)
```

2) Các kiểu dữ liệu List, Tuple

```
# Kiểu dữ liệu danh sách: LIST
# Đây kiểu dữ liệu có khả năng chứa nhiều phần tử khác nhau
# kể cả khác kiểu dữ liệu
listA = ['red',100,'blue',5.5,"yellow",2+4j]
# Có đánh chỉ chỉ số dương lẫn âm
# listA đánh chỉ số dương: 0,1,2,3,4,5
# listA đánh chỉ số âm: -6,-5,-4,-3,-2,-1
print(listA[2])
print(listA[-2])
# Kết quả trả về của việc lấy theo một dãy các
# phần tử theo chỉ sô
# trong danh sách cũng là danh sách
# --> Cái này cũng được gọi là kỹ thuật SLICE
subListA = listA[2:5]
print(subListA)
print(subListA[1])
print(listA[1:-2])
# Đếm số lượng phần tử của danh sách
soluongpt = len(listA)
print(soluongpt)
print(len(listA))
# Lấy phần tử cuối cùng trong list
print(listA[-1]) # Lấy theo chỉ số âm
print(listA[soluongpt-1]) # Lấy theo chỉ số dương
```

3) Các cấu trúc điều khiển, hàm, danh sách, tập tin

1. Viết chương trình in ra "Hello, World!"
2. Viết chương trình nhập vào 2 số và tính tổng của chúng
3. Viết chương trình nhập vào bán kính của một hình tròn và tính diện tích của nó.
4. Viết chương trình tìm số lớn nhất trong một list.
5. Viết chương trình tìm số nhỏ nhất trong một list.
6. Viết chương trình tìm tổng các số trong một list.
7. Viết chương trình đảo ngược một chuỗi.
8. Viết chương trình kiểm tra xem một số có phải số nguyên tố hay không
9. Viết chương trình đổi độ C sang độ F
10. Viết chương trình đổi độ F sang độ C.
11. Viết chương trình tìm số lớn thứ hai trong một list.

12. Viết chương trình tìm giá trị trung bình của các phần tử trong một list.
13. Viết chương trình kiểm tra xem một chuỗi có phải là palindrome hay không.
14. Viết chương trình tìm các số chia hết cho 3 hoặc 5 trong một list.
15. Viết chương trình tìm số Fibonacci thứ n.
16. Viết chương trình sắp xếp một list theo thứ tự tăng dần.
17. Viết chương trình tính số lần xuất hiện của một phần tử trong một list.
18. Viết chương trình đổi một số nhị phân sang hệ thập phân.
19. Viết chương trình kiểm tra xem một chuỗi có phải là chuỗi hợp lệ để biểu diễn một số nguyên hay không
20. Viết chương trình tìm số lớn nhất trong một list không sử dụng hàm max().
21. Viết chương trình giải phương trình bậc 1 ax + b = 0, trong đó, a, b là các số thưc được nhập từ bàn phím
22. Tính tổng s = 0 + 1 + ... + n, trong đó n là số tự nhiên nhập từ bàn phím. Yêu cầu viết bằng for…range, while.
23. Dùng for…in:

a) Liệt kê danh sách các ký tự có trong 1 từ, với từ được nhập vào từ bàn phím
b) Liệt kê danh sách các giá trị có trong 1 danh sách, với danh sách được nhập vào từ bàn phím.
c) In ra màn hình các số chẵn của 1 mảng được nhập vào từ bàn phím.
d) Nhập vào một chuỗi ký tự, nhập ký tự cần tìm. In ra ký tự đó xuất hiện bao nhiêu lần tại vị trí nào và cho biết chiều dài chuỗi ký tự nhập vào.

24. Viết chương trình tính tiền điện hàng tháng của một hộ gia đình. Người sử dụng sẽ nhập số kWh từ bàn phím (là một số nguyên). Tiền điện được tính như sau: Nếu số KhW nhỏ hơn bằng 100 thì đơn giá là 2000 VNĐ. Nếu số KWh vượt 100, thì đơn giá cho mỗi KWh sẽ được cộng dồn thêm 100 VNĐ

Ví dụ:

Số Kwh = 90 thì tổng tiền = 180000 = 90 * 2000

Số KWh = 101 thì tổng tiền = 202100 = 100 * 2000 + (2000 + 100)

Số Kwh = 102 thì tổng tiền = 204300 = 100*2000 + (2000+100) + (2000+100+100)

25. Viết hàm tính giá trị bình phương của một số Sau đó, gọi hàm.
26. Viết hàm tính các phép tính số học (+,-,*,/) của hai số được nhập vào từ bàn phím.
27. Hãy nhập một số tự nhiên từ bàn phím sau đó, xuất màn hình thông báo số đó có phải là số nguyên tố hay không
28. Hãy nhập từ bàn phím số tự nhiên n và xuất ra màn hình: Các số nguyên tố nhỏ hơn n, Xuất ra tổng các số nguyên tố nhỏ hơn n
29. Hãy viết chương trình đọc file in.txt và hiển thị ra màn hình nội dung từng dòng trong file đó

30. Hãy viết chương trình xuất ra file out.txt các số chẵn nhỏ số n được nhập từ bàn phím. Biết rằng mỗi dòng thì lưu 1 số
31. Đọc file dữ liệu num_input.csv Hãy lưu trữ các số trong file vào chương trình. Sau đó, xuất ra màn hình tổng của các số trong file.
32. Tạo ra một list mới mang tên myUpper chứa các chuỗi trong myCharacter nhưng được viết hoa
33. Hãy tạo danh sách myEvenSquare là các bình phương các số chẵn trong myNumber
34. Hãy lấy ra danh sách myChanChuc các số là số chẵn và chia hết cho 5

myChanChuc = list(filter(lambda x: (x%2 == 0 and x%5 == 0), myNumber))

print(myChanChuc)

(Chúng ta dùng lambda trong filter khi điều kiện lọc là hàm phức tạp hoặc trên tập dự liệu có thể lặp (iterator) còn nếu hàm đơn giản và dùng trên list thì chúng ta có thể dùng list comprhension)

Tương tự chúng ta có thể sử dụng map như filter. Hãy tạo ra danh sách myLapPhuong lập phương các số trong myNumber

myLapPhuong = list(map(lambda x: x**3, myNumber))

print(myLapPhuong)

35. Viết chương trình in list sau khi đã xóa số tại vị trí thứ 1, thứ 2, thứ 3, thứ 6 trong [12,24,35,70,88,120,155].

Dùng list comprehension và enumerate()

2) LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG

LAB RECTANGE

Tạo file Rectangle.py

```
Cài đặt lớp hình chữ nhật theo thiết kế
Có 2 fields (thuộc tính): width, length
Có các phương thức:
- tính diện tích (area)
- tính chu vi (perimeter)
- hiện thị cơ bản (display)
Phạm vi khai báo class Rectangle được tính từ phím tab sau class Rectangle
'''
class Rectangle:
    '''
    Hàm (method) khởi tạo (constructor)
```

```
    Đây là method (phương thức) đặc biệt phải có khi khai báo class
    Mục đích: Để nạp những giá trị ban đầu cho các thể hiện (cụ thể)
    của đối tượng khi chạy chương trình
    '''
    def __init__(self,width,length):
        self.width = width
        self.length = length

    def area(self):
        result = self.width * self.length
        return result
    def perimeter(self):
        result = 2*(self.width + self.length)
        return result
    def display(self):
         print(f'rộng: {self.width}, dài: {self.length}, chu vi: {self.perimeter():.2f}, diện tích:
      {self.area():.2f}\n')
```

Tạo các file sau:

ChuNhat1.py

```
import Rectangle as hcn
# Hãy khai báo 2 hình chữ nhật cụ thể hay còn gọi là thể hiện (instances)
# Sau đó, in ra màn hình kích thước và diện tích , chu vi của mỗi cái
hcn1 = hcn.Rectangle(4,10)
hcn1.display()

hcn2 = hcn.Rectangle(7,25)
hcn2.display()
```

ChuNhat2.py

```
Chương trình menu
1- Thêm mới hình chữ nhật
2- Hiển thị danh sách hình chữ nhật
3- Tính tổng diện tích các hình chữ nhật
4- Hiển thị các hình chữ nhật có chu vi nhỏ nhất
Others- Thoát
'''
import Rectangle as rect

menu_options = {
    1: 'Thêm mới hình chữ nhật',
    2: 'Hiển thị danh sách hình chữ nhật',
    3: 'Tính tổng diện tích các hình chữ nhật',
```

```
    4: 'Hiển thị các hình chữ nhật có chu vi nhỏ nhất',
    'Others': 'Thoát chương trình'
}

def print_menu():
    for key in menu_options.keys():
        print (key, '--', menu_options[key] )

# Khai báo biến lưu trữ những hình chữ nhật
dsHCN = []

while(True):
        print_menu()
        userChoice = ''
        try:
            userChoice = int(input('Nhập tùy chọn: '))
        except:
            print('Nhập sai định dạng, hãy nhập lại.....')
            continue
        #Check what choice was entered and act accordingly
        if userChoice == 1:
          cr = float(input("Nhập chiều rộng: "))
          cd = float(input("Nhập chiều dài: "))

          hcn = rect.Rectangle(cr,cd)
          dsHCN.append(hcn)
        elif userChoice == 2:
            for item in dsHCN:
                item.display()
        elif userChoice == 3:
            dientich = 0.0
            for item in dsHCN:
                dientich = dientich + item.area()
            print(f'Tổng diện tích là: {dientich}')
        elif userChoice == 4:
            if dsHCN.count == 0:
                print('Danh sách rỗng')
            else:
                chuvinn = dsHCN[0].perimeter()
                for item in dsHCN:
                    if chuvinn > item.perimeter():
                        chuvinn = item.perimeter()
                for item in dsHCN:
```

```
                if item.perimeter() == chuvinn:
                    item.display()
        else:
            print('Thoát chương trình BYE BYE')
            break
```

ChuNhat3.py

```
Lưu trữ danh sách các hình chữ nhật từ file input.db
Lưu danh sách các hình chữ nhật xuống file output.db theo định dạng
    rong-dai-chuvi-dientich

Lưu ý: Trong các file mỗi hàng là thông tin một hình chữ nhật

'''
import Rectangle as rect

# Tải dữ liệu từ file vào listRectangle
fr = open('input.db',mode='r',encoding='utf-8')
listRectangle = []
for line in fr:
    stripLine = line.strip('\n')
    ds = stripLine.split(',')
    cr = float(ds[0])
    cd = float(ds[1])
    hcn = rect.Rectangle(cr,cd)
    listRectangle.append(hcn)
fr.close()

# Ghi dữ liệu từ listRectangle xuống file
fw = open('output.db',mode='w',encoding='utf-8')
for item in listRectangle:
    fw.write(f'{item.width}-{item.length}-{item.perimeter()}-{item.area()}\n')
fw.close()
'''
(*) Sinh viên tự thực hành
Viết chương trình menu
1- Đọc dữ liệu từ file input.db
2- Thêm mới hình chữ nhật
3- Hiển thị danh sách hình chữ nhật
4- Lưu danh sách hình chữ nhật xuống file demo4output.db
Others- Thoát
'''
import Rectangle as rect
```

```
menu ={
    1:'1- Đọc dữ liệu từ file input.db'   ,
    2:'2- Thêm mới hình chữ nhật',
    3:'3- Hiển thị danh sách hình chữ nhật',
    4:'4- Lưu danh sách hình chữ nhật xuống file demo4output.db',
    'Others':'Thoát'
}
def print_menu():
    for key in menu.keys():
        print (key,'--',menu[key])
#Khai báo biến lưu trữ hình chữ nhật

while(True):
    print_menu()
    chon=''
    try:
        chon =int(input('Nhập tùy chọn:'))
    except:
        print('Nhập sai định dạng, hãy nhập lại:')
        continue
    #Kiểm tra các lựa chọn
    if chon ==1:
        #1- Đọc dữ liệu từ file input.db
        fr=open('input.db', mode='r',encoding ='utf-8')
        dsHCN =[]
        for line in fr:
            stripLine = line.strip('\n')
            ds =stripLine.split(',')
            cr =float(ds[0])
            cd =float(ds[1])
            hcn=rect.Rectangle(cr,cd)
            dsHCN.append(hcn)
        fr.close()
    elif chon ==2:
       # 2- Thêm mới hình chữ nhật
       cr =float(input("Nhập chiều rộng:"))
       cd =float(input("Nhập chiều dài:"))
       hcn =rect.Rectangle(cr,cd)
       dsHCN.append(hcn)
    elif chon ==3:
        #3- Hiển thị danh sách hình chữ nhật
        if dsHCN.count ==0:
            print('Danh sách rỗng')
```

```
        else:
            for item in dsHCN:
                item.display()
    elif chon ==4:
        #4- Lưu danh sách hình chữ nhật xuống file demo4output.db
        fw =open('outpudemo4.db',mode='w',encoding ='utf-8')
        for item in dsHCN:
            fw.write(f'{item.width}-{item.length}-{item.perimeter()}-{item.area()}\n)')
        fw.close()
    else:
        print('Kết thúc chương trình')
        break
```

LAB EMPLOYEES

```
class Employee:
    def __init__(self,code,name,age,salary):
        self.code = code
        self.name = name
        self.age = age
        self.salary = salary
    def income(self):
        result = 0.9*12*self.salary
        return result
    def increaseSalary(self,amount):
        if amount > 0:
            self.salary = self.salary+ amount
    def display(self):
        print(f'code: {self.code}, name: {self.name}, age: {self.age}, salary: {self.salary},
     income: {self.income()}\n')
```

File 2: LabEmployee.py

```
'''
Khai báo đối tượng Employee có:
Các thuộc tính (fields, states) sau
- mã số (code)
- tên (name)
- tuổi (age)
- lương (salary)
```

```
Các phương thức (behaviors, methods)
- Tổng thu nhập hằng năm sau thuế (income) biết rằng tax = 10%
- In ra thông tin của nhân viên (display)
- Tăng lương cho nhân viên (increaseSalary), biết rằng số lương tăng phải lớn hơn 0
- (Sinh viên tự viết) Giảm lương cho nhân viên (decreaseSalary), biết rằng số lương giảm
    phải lớn hơn 0 và không vượt quá 20% lương hiện tại
========== YÊU CẦU CHƯƠNG TRÌNH==========
Khai báo biến danh sách (list) nhân viên (dsNhanVien) để lưu trữ các nhân viên và viết
    chương trình menu thực hiện các chức năng bên dưới
- Opt-1: Tải danh sách nhân viên từ file dbemp_input.db
- Opt-2: Thêm nhân viên vào danh sách
- Opt-3: Hiển thị danh sách nhân viên
- Opt-4: Hiển thị thông tin của một nhân viên khi biết mã nhân viên
- Opt-5: Chỉnh sửa thông tin một nhân viên
- Opt-6: Xóa một nhân viên ra khỏi danh sách
- Opt-7: Tăng lương cho một nhân viên
- Opt-8: Giảm lương cho một nhân viên
- Opt-9: Tính số lượng nhân viên (countEmp) và xuất ra màn hình
- Opt-10: Tính tổng tiền lương của công ty phải trả hàng tháng (sumSalary) và xuất ra màn
    hình
- Opt-11: Tính trung bình lương của nhân viên (avgSalary) và xuất ra màn hình
- Opt-12: Tính độ tuổi trung bình của nhân viên (avgAge) và xuất ra màn hình
- Opt-13: Tính tuổi lớn nhất của các nhân viên (maxAge) và hiển thị danh sách nhân viên
    có tuổi lớn nhất
- Opt-14: Sắp xếp danh sách nhân viên tăng dần theo lương
- Opt-15: Vẽ biểu đồ tương quan lương theo độ tuổi
- Opt-16: Vẽ biểu đồ so sánh lương trung bình của các nhóm tuổi: nhỏ hơn 35, từ 35 đến
    50, hơn 50 trở lên
- Opt-17: Vẽ biểu đồ thể hiện phần trăm tổng lương trên các nhóm tuổi như Opt-16
- Opt-18: Vẽ biểu đồ thể hiện phần trăm số lượng nhân viên theo các nhóm tuổi như Opt-
    16
- Opt-19: Lưu danh sách nhân viên xuống file dbemp_output.db, biết rằng mỗi nhân viên
    là một dòng và các thông tin nhân viên được phân cách bởi dấu '-'
- Opt-Khác: Thoát chương trình
'''
import matplotlib.pyplot as plt
```

```
import Employee as emp

menu_options = {
    1:'Load data from file',
    2:'Add new employee',
    3:'Display list of employee',
    4:'Show employee details',
    5:'Update employee information',
    6:'Delete employee',
    7:'Increase salary of employee',
    8:'Decrease salary of employee',
    9:'Show total employee a month',
    10:'Show total salary a month',
    11:'Show average of salary a month',
    12:'Show average of age',
    13:'Show maximum age',
    14:'Sort list of employee according to salary by ascending',
    15: 'Draw salary according to age',
    16:'Draw average of salary chart by age group',
    17:'Draw percentage of salary by age group',
    18:'Draw percentage of total employee by age group',
    19:'Store data to file',
    'Others': 'Exit program'
}

def print_menu():
    for key in menu_options.keys():
        print (key, '--', menu_options[key] )

# Khai báo biến lưu trữ những nhân viên
dsNhanVien = []

while(True):
        print_menu()
        userChoice = ''
        try:
```

```
        userChoice = int(input('Input choice: '))
    except:
        print('Invalid input, try again')
        continue
    #Check what choice was entered and act accordingly
    if userChoice == 1:
        fr = open('dbemp_input.db',mode='r',encoding='utf-8')
        for line in fr:
            stripLine = line.strip('\n')
            ds = stripLine.split(',')
            maso = ds[0]
            ten = ds[1]
            tuoi = int(ds[2])
            luong = float(ds[3])
            nv = emp.Employee(maso,ten,tuoi,luong)
            dsNhanVien.append(nv)
        fr.close()
    elif userChoice == 2:
       maso = input("Input code: ")
       ten = input("Input name: ")
       tuoi = int(input("Input age: "))
       luong = float(input("Input salary: "))
       nv = emp.Employee(maso,ten,tuoi,luong)
       dsNhanVien.append(nv)
    elif userChoice == 3:
        if dsNhanVien.count==0:
            print('Danh sach rong')
        else:
            for item in dsNhanVien:
                item.display()
    elif userChoice == 4:
    #4:Show employee details
        if dsNhanVien.count==0:
            print('Danh sach rong')
        else:
            ma =input("Input Code:")
```

```
            for item in dsNhanVien:
                if(item.code ==ma):
                    item.display()
        elif userChoice == 5:
        #5:Update employee information
            if dsNhanVien.count==0:
                print('Danh sach rong')
            else:
                ma =input("Input Code for Update:")
                for item in dsNhanVien:
                    if(item.code ==ma):
                        item.display()
                        menu={
                            1:'Update Name',
                            2:'Update Age',
                            3:'Update luong',
                            'Others':'Thoat'
                        }
                        def Xuat_menu():
                            for key in menu.keys():
                                print(key,'--',menu[key])
                        while (True):
                            Xuat_menu()
                            traloi=''
                            try:
                                traloi =int(input('Nhap cac tuy chon:'))
                            except:
                                print('Nhap sai dinh dang, nhap lai:')
                                continue
                            if traloi==1:
                                ten = input("Input name: ")
                                item.name =ten
                                item.display()
                            elif traloi ==2:
                                tuoi = int(input("Input age: "))
                                item.age =tuoi
```

```
                    item.display()
                elif traloi ==3:
                    luong = int(input("Input salary: "))
                    item.salary =luong
                    item.display()
                else:
                    print('Ket thuc chinh sua')
                    break

    elif userChoice == 6:
    #6:'Delete employee'
        if dsNhanVien.count==0:
            print('Danh sach rong')
        else:
            ma =input("Input Code for Update:")
            for item in dsNhanVien:
                if(item.code ==ma):
                    item.display()
                    tl = input('Ban co chac chan xoa nhan vien nay khong Y/N?')
                    if tl =='Y':
                        #del item
                        dsNhanVien.remove(item)
        for item in dsNhanVien:
                item.display()
    elif userChoice == 7:
    #7:Increase salary of employee
        if dsNhanVien.count==0:
            print('Danh sach rong')
        else:
            ma =input("Input Code for Update:")
            for item in dsNhanVien:
                if(item.code ==ma):
                    item.display()
                    luongtang = int(input('Nhap muc luong tang'))
                    item.salary = item.salary + luongtang
                    item.display()
```

```
    elif userChoice == 8:
    #8:Decrease salary of employee
        if dsNhanVien.count==0:
            print('Danh sach rong')
        else:
            ma =input("Input Code for Update:")
            for item in dsNhanVien:
                if(item.code ==ma):
                    item.display()
                    luonggiam = int(input('Nhap muc luong giam'))
                    item.salary = item.salary -luonggiam
                    item.display()
    elif userChoice == 9:
    #9:'Show total employee a month'
        if dsNhanVien.count==0:
            print('Danh sach rong')
        else:
            tongsnv=0
            for item in dsNhanVien:
                tongsnv = tongsnv+1
                item.display()
            print('Tong so nhan vien =',tongsnv)
    elif userChoice == 10:
        sumSalary = 0.0
        for item in dsNhanVien:
            sumSalary = sumSalary + item.salary
        print(f'Total salary: {sumSalary}')
    elif userChoice == 11:
        #11:'Show average of salary a month'
        if dsNhanVien.count==0:
            print('Danh sach rong')
        else:
            tongsnv=0
            tongluong=0
            for item in dsNhanVien:
                tongsnv = tongsnv+1
```

```
            tongluong =tongluong +item.salary
            item.display()
        luongtb = tongluong/tongsnv
        print(f'Luong trung binh nhan vien ={luongtb}')
    elif userChoice == 12:
        #12:'Show average of age'
        if dsNhanVien.count==0:
            print('Danh sach rong')
        else:
            tongtuoi=0
            tongsnv=0
            for item in dsNhanVien:
                tongsnv =tongsnv+1
                tongtuoi = tongtuoi+item.age
                item.display()
            tuoitb = tongtuoi/tongsnv
            print(f'Tuoi trung binh nhan vien ={tuoitb}')
    elif userChoice == 13:
        #13:'Show maximum age'
        if dsNhanVien.count==0:
            print('Danh sach rong')
        else:
            for item in dsNhanVien:
                tuoimax=item.age
                break
            for item in dsNhanVien:
                if(item.age>tuoimax):
                    tuoimax = item.age
            print('Tuoi lon nhat =',tuoimax)
    elif userChoice == 14:
        #14:'Sort list of employee according to salary by ascending'
        if dsNhanVien.count==0:
            print('Danh sach rong')
        else:
            tongsnv=0
            for item in dsNhanVien:
```

```
            tongsnv = tongsnv+1
            item.display()
        print('Tong so nhan vien =',tongsnv)
    elif userChoice == 15:
        #Draw salary according to age
        arrTuoi = []
        arrLuong = []
        for item in dsNhanVien:
            arrTuoi.append(item.age)
            arrLuong.append(item.salary)

        # Vẽ đồ thị
        plt.figure(figsize=(15,5))

        plt.title("Age and salary chart")
        plt.xlabel("Ox: age")
        plt.ylabel("Oy: salary")

        plt.plot(arrTuoi,arrLuong, "go")
        plt.show()
    elif userChoice == 16:
        #16:'Draw average of salary chart by age group',
        arrTuoi = []
        arrLuong = []
        for item in dsNhanVien:
            arrTuoi.append(item.age)
            arrLuong.append(item.salary)

        # Vẽ đồ thị
        plt.figure(figsize=(15,5))

        plt.title("Age and salary chart")
        plt.xlabel("Ox: age")
        plt.ylabel("Oy: salary")

        plt.plot(arrTuoi,arrLuong, "go")
```

```
        plt.show()
    elif userChoice == 17:
        #17:'Draw percentage of salary by age group'
        arrTuoi = []
        arrLuong = []
        for item in dsNhanVien:
            arrTuoi.append(item.age)
            arrLuong.append(item.salary)

        # Vẽ đồ thị
        plt.figure(figsize=(15,5))

        plt.title("Age and salary chart")
        plt.xlabel("Ox: age")
        plt.ylabel("Oy: salary")

        plt.plot(arrTuoi,arrLuong, "go")
        plt.show()
    elif userChoice == 18:
        #18:'Draw percentage of total employee by age group'
        arrTuoi = []
        arrLuong = []
        for item in dsNhanVien:
            arrTuoi.append(item.age)
            arrLuong.append(item.salary)

        # Vẽ đồ thị
        plt.figure(figsize=(15,5))

        plt.title("Age and salary chart")
        plt.xlabel("Ox: age")
        plt.ylabel("Oy: salary")

        plt.plot(arrTuoi,arrLuong, "go")
        plt.show()
    elif userChoice == 19:
```

```
        #19:'Store data to file'
        arrTuoi = []
        arrLuong = []
        for item in dsNhanVien:
            arrTuoi.append(item.age)
            arrLuong.append(item.salary)

        # Vẽ đồ thị
        plt.figure(figsize=(15,5))

        plt.title("Age and salary chart")
        plt.xlabel("Ox: age")
        plt.ylabel("Oy: salary")

        plt.plot(arrTuoi,arrLuong, "go")
        plt.show()
    else:
        print('BYE BYE')
        break
```

3) CÁC BÀI LAB KHÁC

1) Dựa vào bài code Hình Chữ Nhật. Viết chương trình cho các hình tròn, hình vuông.
2) Dựa vào code bài Nhân viên, mỗi sinh viên viết chương trình cho dữ liệu đã được phân công như: Sach, Sinhvien, BenhNhan, BacSi, KhachHang, SanPham, Xe, Lop, DonHang, PhieuMuonSach

BÀI TẬP TUẦN 2: CÁC THUẬT TOÁN SẮP XẾP – TÌM KIẾM

1) CÁC THUẬT TOÁN SẮP XẾP

Mục tiêu:

1. Cài đặt các giải thuật sắp xếp: chọn trực tiếp, chèn trực tiếp, đổi chỗ trực tiếp, nổi bọt và quick sort.
2. So sánh thời gian chạy thực tế của các phương pháp sắp xếpkhi kích thước mảng lớn.

Sắp xếp là quá trình xử lý một danh sách các phần tử (hoặc các mẫu tin) để đặt chúng theo một thứ tự thỏa mãn một tiêu chuẩn nào đó dựa trên nội dung thông tin lưu giữ tại mỗi phần tử.

Một số phương pháp sắp xếp thông dụng như: Chọn trực tiếp (Selection sort), chèn trực tiếp (Insertion sort), đổi chỗ trực tiếp (Interchange sort), nổi bọt (Bubble sort), shell sort, heap sort, quick sort, merge sort, radix sort.

Trong đó, các thuật toán như Interchange sort, Bubble sort, Insertion sort, Selection sort là những thuật toán đơn giản dễ cài đặt nhưng chi phí cao. Các thuật toán Shell sort, Heap sort, Quick sort, Merge sort phức tạp hơn nhưng hiệu suất cao hơn. Cả hai nhóm thuật toán trên đều có một điểm chung là đều được xây dựng dựa trên cơ sở việc so sánh giá trị của các phần tử trong mảng (hay so sánh các khóa tìm kiếm). Riêng phương pháp Radix sort đại diện cho một lớp các thuật toán sắp xếp khác hẳn các thuật toán trước. Lớp thuật toán này không dựa trên giá trị của các phần tử để sắp xếp.

1. Viết chương trình sắp xếp tăng dần cho mảng số nguyên có n phần tử bằng các thuật toán sắp xếp *đã được học*. Chương trình hiện menu gồm các mục sau:

1. Phương pháp Đổi chỗ trực tiếp (Interchange sort)
2. Phương pháp Nổi bọt (Bubble sort)
3. Phương pháp Chèn trực tiếp (Insertion sort)
4. Phương pháp Chọn trực tiếp (Selection sort)
5. Phương pháp dựa trên phân hoạch (Quick sort)
6. Phương pháp sắp xếp vun đống Heapsort
7. Phương pháp sắp xếp MergeSort
8. Phương pháp ShellSort

Người dùng chọn một trong các mục trong menu thì sẽ thực hiện sắp xếp mảng theo thuật toán đã chọn. Mỗi thuật toán viết bằng hàm

Vấn đề 2: So sánh thời gian thực tế của các thuật toán sắp xếp.

Thuật toán sắp xếp	**Thời gian chạy (mili giây)**	**Số lần so sánh (lệnh if)**	**Số lần hoán vị**
1. Phương pháp Đổi chỗ trực tiếp (Interchange sort)			
2. Phương pháp Nổi bọt (Bubble sort)			
3. Phương pháp Chèn trực tiếp (Insertion sort)			
4. Phương pháp Chọn trực tiếp (Selection sort)			
5. Phương pháp dựa trên phân hoạch (Quick sort)			
6. Phương pháp sắp xếp vun đống Heapsort			
7. Phương pháp sắp xếp MergeSort			
8. Phương pháp ShellSort			

2. Để lưu trữ các thông tin về sinh viên trong một trường Đại Học người ta dùng cấu trúc sau

```
struct SINHVIEN{   int MASV;
                   char HOTEN[40];
                   char DIACHI[50];
                   float DIEM;};
```

Viết chương trình yêu cầu thực hiện các công việc sau:

1. Nhập n sinh viên
2. Xuất danh sách sinh viên vừa nhập
3. Xuất danh sách theo thứ tự tăng dần của Masv bằng các thuật toán *Sắp xếp đã được học.*
4. Xuất danh sách theo thứ tự tăng dần của Hoten bằng các thuật toán *Sắp xếp đã được học.*
5. Viết thủ tục tìm kiếm sinh viên có mã số X (X được nhập từ bàn phím)

3. Để lưu trữ các thông tin về sinh viên trong một trường Đại Học người ta dùng cấu trúc sau

```
struct SINHVIEN{   int MASV;
                   char HOTEN[40];
```

char DIACHI[50];
float DIEM;};

Viết chương trình yêu cầu thực hiện các công việc sau:

1. Nhập n sinh viên → thông tin sinh viên sẽ được lưu vào file sinhvien.txt. Các câu còn lại được thực hiện trên File
2. Xuất danh sách sinh viên vừa nhập từ file trên.
3. Xuất danh sách theo thứ tự tăng dần tuỳ theo các tiêu chí do người dùng chọn (Masv, HoTen, DiaChi, Diem) bằng các thuật toán *sắp xếp đã được học.*
4. Xuất danh sách theo thứ tự giảm dần tuỳ theo các tiêu chí do người dùng chọn (Masv, HoTen, DiaChi, Diem) bằng các thuật toán *sắp xếp đã được học.*
5. Viết thủ tục tìm kiếm sinh viên tuỳ theo các tiêu chí do người dùng chọn (Masv, HoTen, DiaChi, Diem) (X là dữ liệu được nhập từ bàn phím tuỳ thuộc vào tiêu chí do người dùng chọn)

4) TÌM KIẾM

Mục tiêu :

1. Thuật toán tìm kiếm tuyến tính.
2. Thuật toán tìm kiếm nhị phân.
3. So sánh thời gian chạy thực tế của hai thuật toán tìm tuyến tính và tìm nhị phân khi kích thước mảng rất lớn.

Nhận xét: Tìm kiếm tuyến tính là một giải thuật rất đơn giản khi hiện thực. Giải thuật này tỏ ra khá hiệu quả khi cần tìm kiếm trên một danh sách đủ nhỏ hoặc một danh sách chưa sắp thứ tự đơn giản. Trong trường hợp cần tìm kiếm nhiều lần, dữ liệu thường được xử lý một lần trước khi tìm kiếm: có thể được sắp xếp theo thứ tự, hoặc được xây dựng theo một cấu trúc dữ liệu đặc trưng cho giải thuật hiệu quả hơn

Thuật toán tìm kiếm nhị phân dùng để tìm kiếm phần tử trong một danh sách đã được sắp xếp, ví dụ như trong một danh bạ điện thoại sắp xếp theo tên, có thể tìm kiếm số điện thoại của một người theo tên người đó.

Bài tập

1. Viết chương trình quản lý thông tin của các nhân viên trong một lớp học (tối đa 50 sinh viên). Mỗi sinh viên gồm các thông tin: MSNV, họ và tên, giới tính, Luong và Thưởng. Viết chương trình thực hiện các yêu cầu sau:

1) Nhập các nhân viên vào danh sách.
2) In danh sách nhân viên.
3) Tìm nhân viên có mã số là x theo thuật toán tìm kiếm tuyến tính. Xuất thông tin tìm thấy.
4) Tìm nhân viên có mã số là x theo thuật toán tìm kiếm nhị phân. Xuất thông tin tìm thấy.
5) Tìm nhân viên có tên là x theo thuật toán tìm kiếm tuyến tính. Xuất thông tin tìm thấy.
6) Tìm nhân viên có tên là x theo thuật toán tìm kiếm nhị phân. Xuất thông tin tìm thấy.
7) Sắp xếp danh sách nhân viên theo thứ tự tăng dần của lương.
8) Sắp xếp danh sách nhân viên theo thứ tự tăng dần của họ và tên.

2. Áp dụng các thuật toán tìm kiếm để xây dựng chương trình tra từ điển Anh-Việt.

Ghi chú: Định nghĩa cấu trúc WORD trong từ điển bao gồm từ gốc tiếng Anh và nghĩa của từ (tiếng Việt không dấu).

Yêu cầu: Xây dựng các hàm thực hiện các chức năng sau

```
struct WORD
{     char Name[256];
      char Meaning[512];
};
```

1. Nhập thông tin một từ (Word)
2. Xuất thông tin một từ (Word)
3. Sắp xếp theo Name tăng dần
4. Tìm kiếm từ theo Name
5. Ghi một từ (Word) vào tệp
6. Đọc một từ (Word) trong tệp
7. Cập nhật Meaning của một Word trong tệp
8. Xóa một từ khỏi tệp

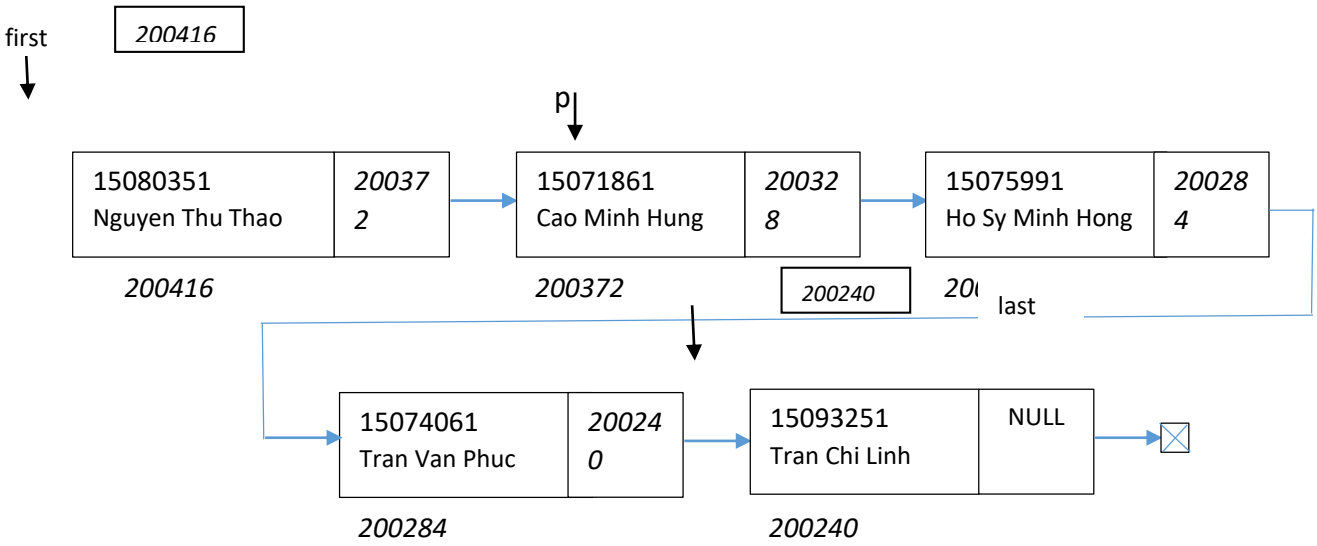
BÀI TẬP TUẦN 3 - 4: DANH SÁCH LIÊN KẾT

Mục đích :

1. Hiểu được các thành phần của danh sách liên kết.
2. Thực hiện được một số thao tác cơ bản trên danh sách liên kết đơn: Tạo danh sách, thêm một phần tử vào đầu/ cuối danh sách, duyệt, tìm kiếm trong danh sách.

Vấn đề 1: Hiểu được các thành phần của danh sách liên kết.

Danh sách liên kết dưới đây lưu thông tin của các sinh viên. Mỗi Node có hai trường dữ liệu (data) và trường liên kết (link). Trường data lưu thông tin của một sinh viên gồm có mã sinh viên và họ tên. Trường link lưu địa chỉ của của Node kế tiếp. Trong hình, dòng chữ in nghiêng phía dưới mỗi Node là địa chỉ của Node đó được lưu trong bộ nhớ.



Vấn đề 2: Thực hiện một số thao tác cơ bản trên danh sách liên kết đơn.

1) BÀI TẬP DANH SÁCH LIÊN KẾT ĐƠN

1) Viết chương trình nhập từ bàn phím các số nguyên cho đến khi nhập vào chuỗi rỗng. Các số nguyên này được đưa vào 1 danh sách. Xuất ra:
1. Số lượng các phần tử trong danh sách
2. Tổng giá trị của các số trong danh sách
3. Giá trị lớn nhất trong danh sách
4. Giá trị nhỏ nhất trong danh sách
5. Sắp xếp Danh sách theo thứ tự tăng dần

6. Sắp xếp Danh sách theo thứ tự giảm dần
7. Danh sách các phần tử có giá trị là chẵn
8. Tìm 1 phần tử có giá trị X trong danh sách

Yêu cầu: Viết bằng danh sách liên kết

2) Tạo menu và thực hiện các chức năng sau trên DSLK đơn chứa số nguyên:

1. Thêm một số pt vào cuối ds
2. Thêm 1 pt vào trước pt nào đó
3. In ds
4. In ds theo thứ tự ngược
5. Tìm GTNN, GTLN trong ds
6. Tính tổng số âm, tổng số dương trong ds
7. Tính tích các số trong ds
8. Tính tổng bình phương của các số trong ds
9. Nhập x, xuất các số là số nguyên tố của x
10. Nhập x, xuất các số là ước số của x
11. Nhập x, tìm giá trị đầu tiên trong ds mà >x.
12. Xuất số nguyên tố cuối cùng trong ds
13. Đếm các số nguyên tố
14. Kiểm tra xem ds có phải đã được sắp tăng không
15. Kiểm tra xem ds có các pt đối xứng nhau hay không
16. Xóa pt cuối
17. Xóa pt đầu
18. Hủy toàn bộ ds

3) Xây dựng list để lưu trữ danh sách sinh viên (Masv, TenSV, MonHoc, Diem). Ghi nhiệm vụ sau:

1. Chèn một học sinh mới vào danh sách cuối cùng.
2. Sắp xếp danh sách theo thứ tự tăng dần của Điểm
3. Chèn một học sinh mới vào danh sách đã sắp xếp để có danh sách cũng đã sắp xếp
4. Lấy danh sách sinh viên có Điểm lớn hơn x.
5. Tìm kiếm k sinh viên có Điểm cao nhất
6. Loại bỏ tất cả sinh viên có Điểm nhỏ hơn x.
7. Hợp nhất vào danh sách sinh viên.
8. In ra màn hình danh sách sinh viên.
9. Ghi danh sách học sinh ra file txt.

4). Thông tin của một quyển sách trong thư việc gồm các thông tin: Tên sách (chuỗi), Tác giả (chuỗi, tối đa 5 tác giả), Nhà xuất bản (chuỗi), Năm xuất bản (số nguyên), giá int

1. Hãy tạo danh sách liên kết đơn chứa thông tin các quyển sách có trong thư viện (được nhập từ bàn phím). Người dùng không nhập nữa thì thôi
2. Thêm 1 sách mới vào đầu, cuối danh sách. Chú ý nếu tên sách có rồi thì bắt nhập tên khác
3. Xuất danh sách các cuốn sách
4. Cho biết số lượng các quyển sách của một tác giả bất kỳ (nhập từ bàn phím).
5. Trong năm YYYY (nhập từ bàn phím), nhà xuất bản ABC (nhập từ bàn phím) đã phát hành những quyển sách nào.
6. Tìm 1 sách có tên là x, nếu có xuất thông tin sách vừa tìm thấy và cho phép người dung thêm vào 1 sách mới, nếu tên sách đó có rồi thì thông báo đã có nhập lại tên khác
7. Xóa 1 sách khỏi danh sách theo 3 cách: xóa đầu, xóa cuối và xóa 1 sách có tên là
8. Sắp xếp danh sách các sách theo Tên Sách.

4) Hãy viết hàm thêm một sách mới vào danh sách sau cho sau khi thêm danh sách các sách vẫn còn là một danh sách có thứ tự.

Hướng dẫn:

Để thêm vào danh sách có thứ tự ta phải tìm vị trí thích hợp để thêm. Nếu phần tử cần thêm xbé hơn phần tử đầu danh sách thì thực hiện thêm xvào đầu danh sách. Ngược lại, sẽ tìm vị trí p (p là con trỏ) thỏa điều kiện, dữ liệu tại p bé hơn x, và dữ liệu tại node sau p lớn hơn x. Khi đó ta sẽ tiến hành thêm x sau p.

BÀI TẬP TUẦN 5: STACK – NGĂN XẾP

1) Mục tiêu :

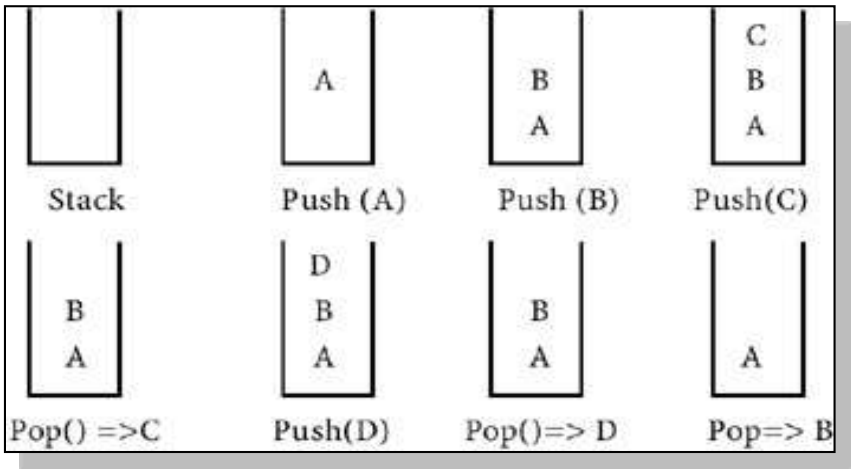
1) Cài đặt các thao tác cơ bản trên stack dùng danh sách liên kết.
2) Ứng dụng stack trong những bài toán đơn giản.

2) Lý thuyết

Định nghĩa

Stack là một danh sách mà các đối tượng được thêm vào và lấy ra chỉ ở một đầu của danh sách (A stack is simply a list of elements with insertions and deletions permitted at one end)

Các phần tử được đưa vào và lấy ra trong ngăn xếp theo nguyên tắc "vào sau ra trước" và cấu trúc dữ liệu này còn được gọi là cấu trúc LIFO (Last In – First Out - *Vào sau ra trước*)



- ❑ Stack hỗ trợ 2 thao tác chính:
- ☐ **S.Push(x)**:**Thêm** 1 đối tượng vào Stack (Push(x,S)
- ☐ **S.Pop()**: **Lấy** 1 đối tượng ra khỏi Stack (Pop(S)
- ❑ Ví dụ:

  5 2 3 - - 4

- ❑ Stack cũng hỗ trợ một số thao tác khác:
- ◘ **S.is_Empty**(S): Kiểm tra xem Stack có rỗng không(Empty(S)
- ◘ **S.Top**(): Trả về giá trị của phần tử nằm ở đầu Stack mà không hủy nó khỏi Stack. Nếu Stack rỗng thì lỗi sẽ xảy ra (Top(S)
- ◘ Len(S): Trả về số phần tử trong stack

◘ S.Create_Stack(): tạo ngăn xếp rỗng (CreateStack(S)

Khởi tạo Stack:

```
5   class ArrayStack:
6       '''LIPO stack
7       '''
8       def __init__(self):
9           #Tạo stack rỗng
10          self.data =[]
```

☐ **Kiểm tra Stack có rỗng hay không:**

```
14      def is_empty(self):
15          #return True if the stack is empty
16          return len(self.data) == 0
```

☐ **Thêm một phần tử x vào Stack**

```
17      def push(self,e):
18          #Add element a to the top of the stack
19          self.data.append(e)
```

☐ **Lấy một phần tử ra khỏi Stack**

```
26      def pop(self):
27          #Remove and return the element from the top of the stack
28          #Raise Empty excepty if the stack is empty
29          if self.is_empty():
30              raise Empty('Stack is EMpty')
31          return self.data.pop()
```

☐ **Xem phần tử ở đỉnh Stack**

```
20      def top(self):
21          #Return the element at the top of the stack
22          #Raise Empty Exception if the stack is empty
23          if self.is_empty():
24              raise Empty('Stack is Empty')
25          return self.data[-1] #the last in the list
```

☐ **Đổi về kiểu chuỗi:**

```
32      def __str__(self):
33          #A string representation of the stack
34          # An arrow show the top of the stack
35          return ''.join(str(self.data))+'->'
36      ############################
```

☐ **Thực thi chương trình**

```
37      print('=============Demo Stack===============')
38  if __name__ =='__main__':
39      S = ArrayStack()
40      S.push(5)
41      S.push(3)
42      print('Stack Length: ',len(S))
43      print('S: ',S)
44      print('Pop ',S.pop())
45      print('Is stack Empty? ',S.is_empty())
46      print('Pop ',S.pop())
47      print('Is stack Empty? ', S.is_empty())
48      print('S:',S)
49      S.push(7)
50      S.push(9)
51      print('Top Element in Stack : ', S.top())
52      S.push(4)
53      S.push(6)
54      print('S: ',S)
```

Hiện thực Stack dùng DSLK

☐ Các phương thức cần cài đặt:

◘ __init__(self): Khởi tạo ngăn xếp (Stack) với 1 danh sách

```
#Tạo lớp Stack
class NganXep:
    #Khởi tạo Ngăn xếp Stack
    def __init__(self):
        self.danh_sach = []
    #def
    #Kiểm tra ngăn xếp có rỗng không
```

◘ Is_Empty(self): Kiểm tra ngăn xếp rỗng

```
    #Kiểm tra ngăn xếp có rỗng không
    def is_Empty(self):
        return len(self.danh_sach) == 0
    #def
```

◘ __str__(self): Chuyển ngăn xếp về kiểu chuỗi

```
11      #Đổi về chuỗi
12      def __str__(self):
13          kq = 'Ngăn xếp ['
14          stt = 0
15          for x in self.danh_sach:
16              stt = stt +1
17              if stt ==1:
18                  kq = kq + str(x)
19              else:
20                  kq = kq +' ->' +str(x)
21              #if
22          #for
23          kq = kq +']'
24          return kq
25      #def
```

- Push(self , Gia_tri): Thực hiện đưa Gia_tri vào ngăn xếp bằng cách them vào đầu danh sách.

```
26      #Đẩy vào stack
27      def Push(self,gia_tri):
28          #đẩy vào
29          self.danh_sach.insert(0,gia_tri) #Thêm vào đầu
30      #def
31      #Lấy ra khỏi stack
```

- Pop(self): Thực hiện lấy ra 1 giá trị từ ngăn xếp

```
31      #Lấy ra khỏi stack
32      def Pop(self):
33          #Kiểm tra rỗng
34          if self.is_Empty():
35              return None
36          else:
37              return self.danh_sach.pop(0)
38      #def
39  #Class
```

- Tạo đối tượng ngăn xếp

```
40  #Tạo đối tượng ngăn xếp
41  if __name__ =='__main__':
42      ngan_xep =NganXep()
43      print(ngan_xep)
44      print('-----Đẩy vào-----')
45      for i in range (1,6):
46          print(f'* Đẩy vào {i}')
47          ngan_xep.Push(i)
48          print(ngan_xep)
49      #for
50      print('-----Lấy ra-----')
51      while not ngan_xep.is_Empty():
52          gt =ngan_xep.Pop()
53          print(f'* Lấy ra -> {gt}')
54          print(ngan_xep)
55      #while
```

BÀI TẬP

1) Viết chương trình thực hiện các công việc sau:

a. Tạo danh sách có các phần tử được thêm vào STACK lần lượt theo thứ tự: 6 3 5 10 25 7 1 2 9 15.

b. Xuất danh sách vừa tạo.

c. Tiếp tục thực hiện các thao tác dưới đây. Xuất lại danh sách sau mỗi lần thao tác.

   i. Xóa phần tử 10

   ii. Xóa phần tử 6

   iii. Thêm phần tử 15

   iv. Xóa phần tử 10

2) Xét một biểu thức số học với các phép toán cộng, trừ, nhân, chia, lũy thừa, ...

Ví dụ: biểu thức a + (b - c) * d+ e. Biểu thức như trênđược viết theo ký pháp trung tố, nghĩa là toán tử (dấu phép toán) đặt giữa hai toán hạng.Với ký pháp trung tố, để phân biệt toán hạng ứng với toán tử nào ta phải dùng các cặp dấu ngoặc đơn, và phải chấp nhận một thứ tự ưu tiên giữa các phép toán. Các phép toán cùng thứ tự ưu tiên thì sẽ thực hiện theo trình tự từ trái sang phải. Thứ tự ưu tiên như sau:

1. Phép lũy thừa
2. Phép nhân, chia
3. Phép cộng, trừ

Cách trình bày biểu thức theo ký pháp trung tố là tự nhiên với con người nhưng lại "khó chịu" đối với máy tính, vì nó không thể hiện một cách tường minh quá trình tính toán để đưa ra giá trị của biểu thức.Để đơn giản hóa quá trình tính toán này, ta phải biến đổi lại biểu thức thông thường về dạng ký pháp Ba Lan, gồm có có hai dạng là tiền tố (prefix) và hậu tố (postfix). Đó là một cách viết biểu thức đại số rất thuận lợi cho việc thực hiện các phép toán. Đặc điểm cơ bản của cách viết này là không cần dùng đến các dấu ngoặc và luôn thực hiện từ trái sang phải.

Ta có thể biến đổi biểu thức dạng trung tố sang tiền tố hoặc hậu tố. Ví dụ:

Dạng trung tố	**Dạng tiền tố**	**Dạng hậu tố**
A + B	+ A B	A B +
A / B	/ A B	A B /
(A + B) * C	* + A B C	A B + C*
(A + B) / (C – D)	/ + A B – C D	A B + C D - /
A + B / C – D	- + A / B C D	A B C / + D -

Để hiểu rõ cách chuyển biểu thức sang các dạn khác nhau, sinh viên hãy thực hiện chuyển các biểu thức trung tố dưới đây sang dạng tiền tố và hậu tố:

1. A + B – C
2. A * (B – C)
3. A + B * C / D
4. A – B – (C + D) / E
5. A + (B - C) * D+ E

Cách tính giá trị của một biểu thức dạng hậu tố

Xét biểu thức dạng hậu tố sau đây: 128 + 4 - 2 4 * /

Nếu đọc biểu thức dang hậu tố từ trái qua phải ta sẽ thấy khi một toán tử xuất hiện thì hai toán hạng vừa đọc sát nó sẽ được kết hợp với toán tử này để tạo thành toán hạng mới ứng với toán tử sẽ được đọc sau nó, và cứ như vậy.

Với biểu thức trên, các bước thực hiện lần lượt như sau:

Khi đọc phép: +, thực hiện 12 + 8 = 20

Khi đọc phép: -, thực hiện 20 – 4 = 16

Khi đọc phép: *, thực hiện 2 * 4 = 8

Khi đọc phép: /, thực hiện 16 / 8 = 2

Nhận xét: Trước khi đọc tới toán tử thì giá trị của các toán hạng phải được lưu lại để chờ thực hiện phép tính. Hai toán hạng được đọc sau thì lại kết hợp với toán tử đọc trước, điều đó cũng có nghĩa là hai giá trị được lưu lại sau thì phải lấy ra trước để tính toán. Điều này trùng khớp với cơ chế "last in fist out" của stack. Vì thế, để tính giá trị của biểu thức hậu tố người ta cần dùng một stack để lưu các giá trị của toán hạng.

Cách thực hiện tính giá trị của biểu thức hậu tố có thể được tóm tắt như dưới đây. Lưu ý, quy ước trình bày biểu thức là: biểu thức là một mảng ký tự, trong đó các toán tử và các toán hạng được viết cách nhau bằng một ký tự khoảng trắng.

Ví dụ, ta có biểu thức hậu tố 5 17 3 + * 6 - . Biểu thức này được lưu trong một mảng ký tự như sau:

Tính giá trị của biểu thức dạng hậu tố:

Đọc từng "từ" của biểu thức hậu tố từ trái sang phải (các "từ" cách nhau bằng một khoảng trắng). "Từ" đọc được gọi là X.

Nếu X là toán hạng thì đưa X vào stack.

Nếu X là toán tử thì:

- Lần lượt lấy từ stack ra hai giá trị a và b *(a lấy trước, b lấy sau)*.
- Tính: kq = b X a *(với X là phép toán*).
- Đưa kq vào stack.

Quá trình trên được tiếp tục cho tới khi kết thúc biểu thức. Lúc đó giá trị còn trong stack chính là giá trị của biểu thức.

Câu hỏi: Tại bước tính: kq = b X a có thể thay bằng: kq = b X a được không?

5		1	7		3		+		*		6		-		#
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	...

Yêu cầu: Viết chương trình để chuyển đổi biểu thức có dạng trung tố sang dạng hậu tố.

3) Viết chương trình tính giá trị của 1 biểu thức

Lập trình Queue giải quyết bài toán quản lý kho (nhập trước – xuất trước) như sau:

1. Nhập một mặt hàng
2. Xuất một mặt hàng
3. Xem mặt hàng chuẩn bị xuất
4. Xem mặt hàng mới nhập
5. Xem các mặt hàng có trong kho
6. Xuất toàn bộ kho hàng
7. Kết thúc chương trình

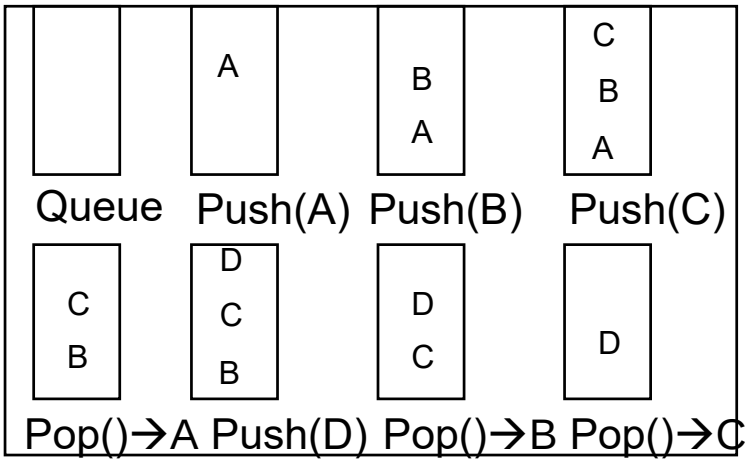
CHỨC NĂNG BẠN CHỌN :[1..7]:

BÀI TẬP TUẦN 6: QUEUE

Mục tiêu :

1) Cài đặt các thao tác cơ bản trên QUEUE dùng danh sách liên kết.
2) Ứng dụng QUEUE trong những bài toán đơn giản.

- Queue là một danh sách mà các đối tượng được **thêm vào** ở một đầu của danh sách và **lấy ra** ở một đầu kia của danh sách. (A *queue is also a list of elements with insertions permitted at one end and deletions permitted from the other end)*
- Việc thêm một đối tượng luôn diễn ra ở **cuối** Queue và việc lấy ra một đối tượng luôn diễn ra ở **đầu** Queue
- Vì thế, thao tác trên Queue được thực hiện theo cơ chế FIFO (First In First Out - *Vào trước ra trước*)



Cài đặt các thao tác cơ bản trên queue dùng danh sách liên kết.

Hai thao tác chính trên queue gồm có: thêm và lấy phần tử ra khỏi queue. Đối với queue dùng danh sách liên kết thì thêm phần tử vào queue chính là thao tác thêm phần tử vào cuối danh sách liên kết. Lấy phần tử ra khỏi queuechính là thao tác lấy phần tử ở đầu danh sách ra khỏi danh sách liên kết. Ta cũng cần một thao tác hỗ trợ là kiểm tra danh sách rỗng.

(Lưu ý: ta cũng có thể thêm phần tử vào đầu danh sách liên kết, khi đó, để lấy phần tử ra khỏi danh sách, ta thực hiện lấy phần tử ở cuối danh sách liên kết.)

CÁC PHÉP TOÁN TRÊN HÀNG ĐỢI BẰNG MẢNG

- **Q.enqueue(e):** Thêm phần tử e vào cuối hàng đợi Q.
- **Q.dequeue():** Loại bỏ và trả về phần tử đầu tiên từ hàng đợi Q; lỗi xảy ra nếu hàng đợi trống.
- **Q.first():** Trả về một tham chiếu đến phần tử ở đầu hàng đợi Q mà không xóa nó; lỗi xảy ra nếu hàng đợi trống.
- **Q.is_empty():** Trả về True nếu hàng đợi Q không chứa phần tử nào.
- **len(Q):** Trả về số phần tử có trong hàng đợi Q;
- **_data:** là một tham chiếu đến một thể hiện danh sách có giá trị cố định.
- **_size:** là số nguyên biểu thị số hiện tại của các phần tử được lưu trữ trong hàng đợi (ngược lại với độ dài của danh sách dữ liệu).
- **_front:** là số nguyên biểu thị chỉ số trong dữ liệu của phần tử đầu tiên của hàng đợi (giả sử hàng đợi không trống).

CÁC PHÉP TOÁN TRÊN HÀNG ĐỢI BẰNG DANH SÁCH LIÊN KẾT

Định nghĩa lớp DSLienKet trong file DSLK.PY mô tả 1 danh sách liên kết gồm các phương thức:

1. **__init__(self):** khởi tạo
2. **__str(self)__:** Đổi sang kiểu chuỗi
3. **them_dau(self, gia_tri):** Thêm phần tử vào đầu danh sách
4. **them_duoi(self, gia_tri):** Thêm phần tử vào cuối danh sách
5. **lay_dau(self):** Lấy giá trị phần tử đầu danh sách
6. **xoa_dau(self):**Xoá phần tử đầu danh sách

Tạo tập tin DSLKStack.PY:

- Nhập vào từ modun DSLK lớp DSLienKet
- Định nghĩa lớp NganXep gồm các phương thức:
  1. **__init__(self):** khởi tạo
  2. **__str(self)__:** Đổi sang kiểu chuỗi
  3. **is_rong(self):** Kiểm tra stack(danh sách) rỗng

4. **Day_vao(self, gia_tri):** Đẩy 1 phần tử vào đầu danh sách
5. **lay_ra(self):** Lấy 1 phần tử ra khỏi danh sách

Bài tập:

1) Viết đoạn mã thực thi tạo 1 ngăn xếp. Lần lượt them các giá trị từ 1 đến 5 vào hàng đợi qua phương thức **day_vao**. Lần lượt lấy và xuất ra màn hình các giá trị từ hàng đợi qua phương thức **lay_ra**.

2) Viết chương trình thực hiện các công việc sau:

a. Tạo danh sách có các phần tử được thêm vào QUEUE lần lượt theo thứ tự: 6 3 5 10 25 7 1 2 9 15.

b. Xuất danh sách vừa tạo.

c. Tiếp tục thực hiện các thao tác dưới đây. Xuất lại danh sách sau mỗi lần thao tác.

i. Xóa phần tử 10

ii. Xóa phần tử 6

iii. Thêm phần tử 15

iv. Xóa phần tử 10

3). Lập trình Queue giải quyết bài toán quản lý kho (nhập trước – xuất trước) như sau:

8. Nhập một mặt hàng
9. Xuất một mặt hàng
10. Xem mặt hàng chuẩn bị xuất
11. Xem mặt hàng mới nhập
12. Xem các mặt hàng có trong kho
13. Xuất toàn bộ kho hàng
14. Kết thúc chương trình

CHỨC NĂNG BẠN CHỌN :[1..7]:

BÀI TẬP TUẦN 7 - 8: TREE - CÂY

1) CÂY NHỊ PHÂN

Mục đích :

Hoàn tất bài thực hành này, sinh viên có thể:

1. Hiểu được các thành phần của cây nhị phân tìm kiếm.
2. Thành thạo các thao tác trên cây nhị phân tìm kiếm: tạo cây, thêm phần tử, xóa phần tử, cập nhật, duyệt cây, đếm số các nút, tình chiều cao, tìm kiếm, sắp xếp cây nhị phân tìm kiếm.
3. Áp dụng cấu trúc dữ liệu cây nhị phân tìm kiếm vào việc giải quyết một số bài toán.

LÝ THUYẾT

Định nghĩa cây: Cây là một tập gồm 1 hay nhiều nút T, trong đó có một nút đặc biệt được gọi là gốc, các nút còn lại được chia thành những tập rời nhau T_1, T_2 , ... , T_n theo quan hệ phân cấp trong đó T_i cũng là một cây

- ☐ Bậc của một nút (Degree of a Node of a Tree):
  - ◘ Là số cây con của nút đó. Nếu bậc của một nút bằng 0 thì nút đó gọi là nút lá (leaf node)
- ☐ Bậc của một cây (Degree of a Tree):
  - ◘ Là bậc lớn nhất của các nút trong cây. Cây có bậc n thì gọi là cây n-phân
- ☐ Nút gốc (Root node):
  - ◘ Là nút không có nút cha
- ☐ Nút lá (Leaf node):
  - ◘ Là nút có bậc bằng 0
- ☐ Nút nhánh:
  - ◘ Là nút có bậc khác 0 và không phải là gốc
- ☐ Mức của một nút (Level of a Node):
  - ◘ Mức (T_0) = 0, với T_0 là gốc

- Gọi $T_1, T_2, T_3, \dots, T_n$ là các cây con của T_0:

$Mức(T_1) = Mức(T_2) = \dots = Mức(T_n) = Mức(T_0) + 1$

- Chiều cao của cây (độ sâu) (Height – Depth of a Tree):
  - Là mức cao nhất của nút + 1 (mức cao nhất của 1 nút có trong cây)
- Có 3 kiểu duyệt chính có thể áp dụng trên cây nhị phân:
  - Duyệt theo thứ tự trước - preorder (Node-Left-Right: NLR)

```
161     #Duyet NLR
162     def duyet_nut_trai_phai(self,goc=0):
163         #Duyệt theo NLR
164         nut_ht = goc
165         if goc == 0:
166             nut_ht = self.goc
167         #if
168         #kiểm tra nút hiện tại có bằng None không
169         if nut_ht == None:
170             return []
171         else: #cây có giá trị
172             kq = []
173             kq.append(nut_ht.khoa)
174             kq_trai = self.duyet_nut_trai_phai(nut_ht.trai)
175             for x in kq_trai:
176                 kq.append(x)
177             #For duyệt trái
178 
179             #Duyệt phải
180             kq_phai = self.duyet_nut_trai_phai(nut_ht.phai)
181             for x in kq_phai:
182                 kq.append(x)
183             #for
184             return kq
```

- Duyệt theo thứ tự giữa - inorder (Left-Node-Right: LNR)

```
135     #Duyệt cây
136     def duyet_trai_nut_phai(self,goc=0):
137         #Duyệt theo LNR
138         nut_ht = goc
139         if goc == 0:
140             nut_ht = self.goc
141         #if
142         #kiểm tra nút hiện tại có bằng None không
143         if nut_ht == None:
144             return []
145         else: #cây có giá trị
146             kq = []
147             kq_trai = self.duyet_trai_nut_phai(nut_ht.trai)
148             for x in kq_trai:
149                 kq.append(x)
150             #For duyệt trái
151             kq.append(nut_ht.khoa)
152             #Duyệt phải
153             kq_phai = self.duyet_trai_nut_phai(nut_ht.phai)
154             for x in kq_phai:
155                 kq.append(x)
156             #for
157             return kq
```

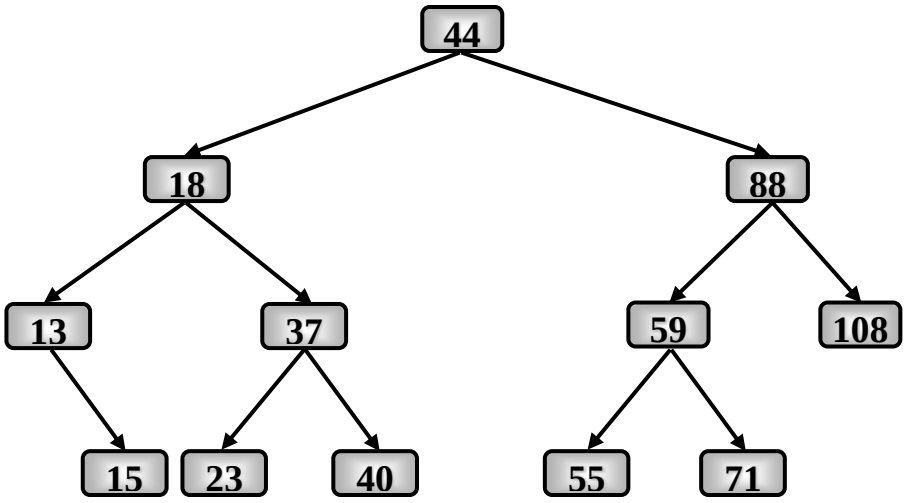
- Duyệt theo thứ tự sau - postorder (Left-Right-Node: LRN)

```
188     def duyet_trai_phai_nut(self,goc=0):
189         #Duyệt theo LRN
190         nut_ht = goc
191         if goc == 0:
192             nut_ht = self.goc
193         #if
194         #kiểm tra nút hiện tại có bằng None không
195         if nut_ht == None:
196             return []
197         else: #cây có giá trị
198             kq = []
199             kq_trai = self.duyet_trai_phai_nut(nut_ht.trai)
200             for x in kq_trai:
201                 kq.append(x)
202             #For duyệt trái
203 
204             #Duyệt phải
205             kq_phai = self.duyet_trai_phai_nut(nut_ht.phai)
206             for x in kq_phai:
207                 kq.append(x)
208             #Not
209             kq.append(nut_ht.khoa)
210             #for
211             return kq
```

Định nghĩa cây nhị phân tìm kiếm

Cây nhị phân tìm kiếm (CNPTK) là cây nhị phân trong đó tại mỗi nút, khóa của nút đang xét lớn hơn khóa của tất cả các nút thuộc cây con trái và nhỏ hơn khóa của tất cả các nút thuộc cây con phải

- ☐ Nhờ ràng buộc về khóa trên CNPTK, việc tìm kiếm trở nên có định hướng
- ☐ Nếu số nút trên cây là N thì chi phí tìm kiếm trung bình chỉ khoảng $\log_2 N$
- ☐ Trong thực tế, khi xét đến cây nhị phân chủ yếu người ta xét CNPTK



BÀI TẬP

1) Hãy viết các chương trình con sau thực hiện trên cây nhị phân:

1. Kiểm tra cây rỗng
2. Kiểm tra nút n có phải là nút lá không.
3. Kiểm tra nút n có phải là nút cha của nút m không.
4. Tính chiều cao của cây.
5. Tính số nút của cây
6. Duyệt tiền tự, trung tự, hậu tự.
7. Đếm số nút lá của cây.
8. Đếm số nút trung gian của cây.
9. Nút có giá trị lớn nhất, nhỏ nhất, tổng giá trị các nút, trung bình giá trị các nút

2) Viết chương trình thực hiện các yêu cầu sau:

1. Khởi tạo một cây nhị phân.
2. Tính và trả về tổng giá trị các node trên cây nhị phân gồm các giá trị nguyên
   Gợi ý: tham khảo hàm NLR để viết hàm SumTree
3. Tìm giá trị nguyên lớn nhất và nhỏ nhất trong số các phần tử nguyên trên cây nhị phân tìm kiếm gồm các giá trị nguyên

Gợi ý: dựa vào tính chất của cây nhị phân

4. Tính và trả về số lượng các node của cây nhị phân gồm các giá trị nguyên
   Gợi ý: tham khảo hàm NLR để viết hàm CountNode
5. Tính và trả về số lượng các node lá trên cây.
   Gợi ý: tham khảo hàm duyệt cây nhị phân NLR

3) Viết chương trình theo các yêu cầu sau:

1. Nhập dữ liệu cho cây. Mỗi node có giá trị là một số nguyên.
2. Duyệt và xuất dữ liệu của cây theo thức tự LNR ra màn hình.
3. Đếm số nút lá của cây
4. Tính chiều cao của cây
5. Chèn một Node vào cây.
6. Tìm kiếm một Node có giá trị được nhập vào từ bàn phím.
7. Xóa một Node có giá trị được nhập vào từ bàn phím.

4) Viết chương trình theo các yêu cầu sau:

1. Nhập dữ liệu cho cây. Mỗi node là thông tin một sinh viên gồm: Masv, Hoten, Malop, ĐiemTB. Kiểm tra nếu masv trùng thì thông báo trùng và nhập lại masv khác. Trong đó MASV là khoá chính.
2. Duyệt và xuất dữ liệu của cây theo thức tự LNR ra màn hình theo MASV.
3. Sắp xếp cây theo trường dữ liệu theo MASV.
4. Đếm số nút lá của cây
5. Tính chiều cao của cây
6. Chèn một Node vào cây.
7. Tìm kiếm một Node có giá trị MASV được nhập vào từ bàn phím.
8. Xóa một Node có MASV được nhập vào từ bàn phím.

5) Sử dụng cây nhị phân tìm kiếm để giải bài toán (thống kê) số lượng ký tự có trong văn bản (không dấu)

1. Xây dựng cây cho biết nỗi ký tự có trong văn bản xuất hiện mấy lần
2. Nhập vào 1 ký tự. Kiểm tra ký tự đó xuất hiện bao nhiêu lần trong văn bản

6) Sử dụng cây nhị phân tìm kiếm để giải bài toán:

1. Đếm có bao nhiêu giá trị phân biệt trong dãy số cho trước
2. Với mỗi giá trị phân biệt, cho biết số lượng phần tử

BÀI THỰC HÀNH SỐ 9
CÂY NHỊ PHÂN TÌM KIẾM CÂN BẰNG(AVL)

(Số tiết: 3)

1) CÂY NHỊ PHÂN TÌM KIẾM CÂN BẰNG

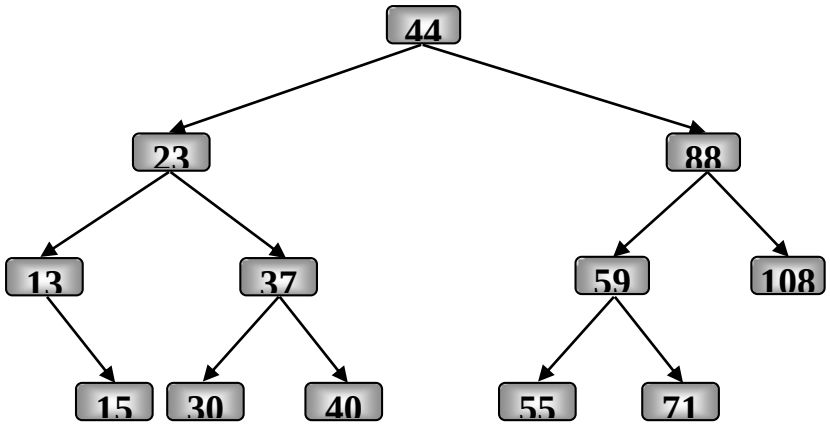
Mục đích :

Hoàn tất bài thực hành này, sinh viên có thể:

1. Hiểu được các thành phần của cây nhị phân tìm kiếm cân bằng.
2. Thành thạo các thao tác trên cây nhị phân tìm kiếm cân bằng: cân bằng, tạo cây, thêm phần tử, xóa phần tử, cập nhật, duyệt cây, đếm số các nút, tình chiều cao, tìm kiếm, sắp xếp cây nhị phân tìm kiếm cân bằng.
3. Áp dụng cấu trúc dữ liệu cây nhị phân tìm kiếm cân bằng vào việc giải quyết một số bài toán.

LÝ THUYẾT

Cây nhị phân tìm kiếm cân bằng (AVL) là cây nhị phân tìm kiếm mà tại mỗi nút độ cao của cây con trái và của cây con phải chênh lệch không quá một.



☐ **Chỉ số cân bằng của một nút:**

- Định nghĩa: Chỉ số cân bằng của một nút là hiệu của chiều cao cây con phải và cây con trái của nó
- Đối với một cây cân bằng, chỉ số cân bằng (CSCB) của mỗi nút chỉ có thể mang một trong ba giá trị sau đây:
  - CSCB(p) = 0 ⇔ Độ cao cây phải (p) = Độ cao cây trái (p)

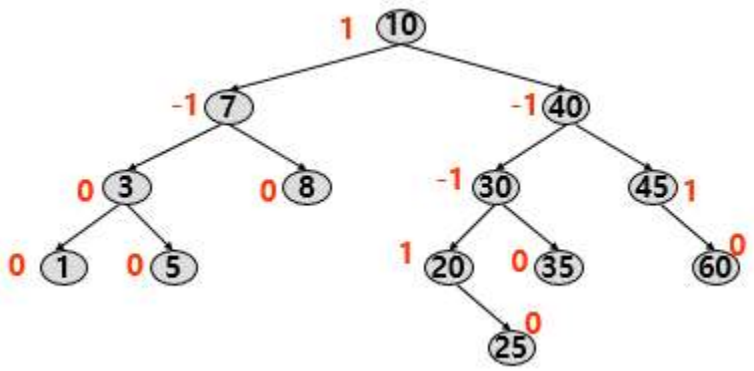
- CSCB(p) = 1 ⇔ Độ cao cây phải (p) > Độ cao cây trái (p)
- CSCB(p) = -1 ⇔ Độ cao cây phải (p) < Độ cao cây trái (p)

☐ Để tiện trong trình bày, chúng ta sẽ ký hiệu như sau:

p->balFactor = CSCB(p);

☐ Độ cao cây trái (p) ký hiệu là hL

☐ Độ cao cây phải(p) ký hiệu là hR



BÀI TẬP

1. Cho dãy A như sau:

1	3	5	7	9	12	15	17	21	23	25	27

a. Tạo cây AVL từ dãy A. Cho biết số phép so sánh cần thực hiện để tìm phần tử 21 trên cây AVL vừa tạo.

b. Tạo cây nhị phân tìm kiếm từ dãy A dùng lại đoạn code tạo cây của bài thực hành trước). Cho biết số phép so sánh cần thực hiện để tìm phần tử 21 trên cây nhị phân tìm kiếm vừa tạo.

c. So sánh 2 kết quả trên và rút ra nhận xét?

2. Cài đặt chương trình đọc các số nguyên từ tập tin input.txt (không biết trước số lượng số nguyên trên tập tin) và tạo cây AVL từ dữ liệu đọc được

3. Cài đặt cây cân bằng AVL trong đó mỗi node trên cây lưu thông tin sinh viên.

4. Tự tìm hiểu và cài đặt chức năng xóa một node ra khỏi cây AVL.

5. Viết chương trình cho phép tạo, tra cứu và sửa chữa từ điển Anh-Việt (sinh viên liên hệ với GVLT để chép file từ điển Anh-Việt)

6. Cài đặt lại các bài tập thêm của cây NPTK bằng cách dùng cây AVL