Львівський національний Університет ім. Івана Франка

Міністерство освіти і науки України

Кафедра обчислювальної математики

Камінська Софія Богданівна

Звіт з обчислювальної практики(Python, C++)

Студентки групи ПМп-11

Кервіник: Гарасим Я.С.

Львів-2016

Зміст

Вступ 3

Задача 1 4

Задача 2 5

Задача 3 6

Задача 4 8

Задача 5 9

Задача 6 12

Задача 7 13

Задача 8 14

Задача 9 18

Задача 10 22

Задача 11 25

Висновок 27

Список літератури 28

Вступ

Мета цього звіту - показати результати курсу, а саме програми на мовах Python i C++.

Python – одна з найперспективніших мов програмування на сьогодні, програмісти всього світу визнають її великий потенціал. Багатоскладних операцій можна виконати кількома рядками , що говорить про простоту. Ця мова чудово підходить для початківців. Розробляючи спочатку прості, а пізніше складніші програми, ми легко навчились реалізовувати їх мові Python .

Коли в другому семестрі ми почали вчити мову C++ вона стала для мене чимось іншим, чим зовсім не схоже на Python. С++ – це мова, що базується на досить старій мові С (тоді як Python – просто написаний на ній) . Тут потрібно вручну реалізовувати чимало того , що на Python робилося в один рядок (або просто підключити потрібні бібліотеки С++) . Програмування на цій мові дає розуміння алгоритмів і підходить тим людям, які опанували мови типу Pascal, але розчарує прихильників мови Python. Лише добре орієнтуючись в алгоритмах ,можна ефективно програмувати на С++ .

Задача 1

Для заданого n обчислити суму

Алгоритм:

Створюємо зміну для циклу; Створюємо цикл який повторює виконання програми; задаємо з клавіатури число n; присвоюємо початкове значення суми; створюємо цикл, який рахує суму; присвоюємо початкове значення добутку; створюємо цикл добутку; рахуємо добуток; рахуємо суму з отриманих добутків; виводимо суму

Код програми:

a=True #змінна для циклу

while a==True: #цикл який повторює виконання програми

n=int(raw\_input('n=')) #задаємо з клавіатури число n

sum=0 #початкове значення суми

for i in range(1,n+1): #цикл, який рахує суму

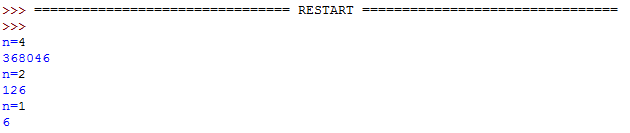
p=1 #початкове значення добутку

for k in range(1,2\*i+1): #цикл добутку

p\*=(k+1) #рахуємо добуток

sum+=p #рахуємо суму з отриманих добутків

print sum #виводимо суму

Приклади:

Задача 2

Використовуючи алгоритм Евкліда знайти найбільший спільний дільник двох цілих чисел.

Алгоритм:

Користувач вводить два цілих числа a та b. В циклі порівнюються два числа і більшому числу присвоюється остача від ділення на менше. Цикл виконується поки одне з чисел не дорівнює нулю. Виводиться результат a+b, оскільки не відомо яке з чисел рівне нулю.

Код програми:

a=int(raw\_input('a='))

b=int(raw\_input('b='))

while a!=0 and b!=0:

if a > b:

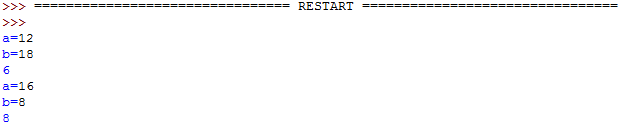
a = a % bd

else:

b = b % a

print a+b

Приклади:



Задача 3

Прочитати дійсну матрицю розміром n×n(n>5). Вияснити чи задана матриця є симетричною відносно головної діагоналі.

Алгоритм:

Вводимо розмір матриці (nxn); в циклі заповняємо матрицю зчитуючи по стрічка; розділяємо стрічку і по елементах додаємо до матриці; переглядаєм в циклі матрицю; порівнюємо дзеркальні елементи відносно діагоналі; якщо якийсь елемент не рівний оберненому елементу, то змінній t присвоюється значення False і переривається цикл. Виводимо результат який залежить від змінної t.

Код програми:

n=int(raw\_input('n='))

m=[]

if n>5:

for i in range(n):

e=raw\_input().split()

e[i]=int(e[i])

m.append(e)

for i in range(n):

for j in range(n):

if m[i][j]==m[j][i]:

t=True

else:

t=False

break

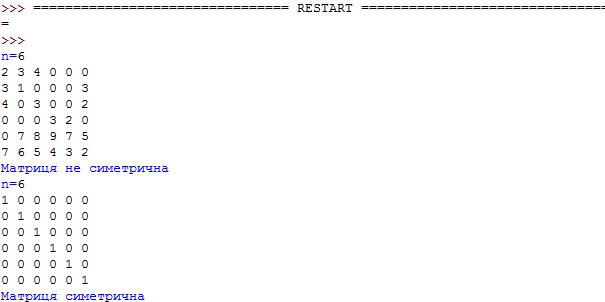
if t==True:

print 'Матриця симетрична'

else: print 'Матриця не симетрична'

else: print 'n>5'

Приклади:



Задача 4

Прочитати текстову стрічку довжтною не більше ніж 253 символи. Вважається, що у стрічку входять слова, які розділені хоча б одним пропуском. Надрукувати слова, які в заданому тексті є найдовшими і найкоротшими.

Алгоритм:

Ввожу стрічку s, після чого розділяю її по пробілах. Створюю словник d. Перевіряю чи довжина стрічки не більша ніж 253 символи. Якщо більша, результатом буде стрічка “Можна вводити не більше 253 символів”. Якщо стрічка не перевищує дану довжину, в циклі я переглядаю всі слова і кожне слово зі стрічки стає ключем словника d, а його довжина значенням. Для того щоб вивести слова за довжиною, я створюю обернений словник inverse. Я переглядаю в циклі всі ключі словника. В новому словнику ключами є довжини слів, а значеннями слова які відповідають даній довжині. Вивожу значення за мінімальним ключем і за максимальним.

Код програми:

s=raw\_input('text ')

s=s.split()

d=dict()

if len(s)>253:

print 'Можна вводити не більше 253 символів'

else:

for x in s:

d[x]=len(x)

inverse=dict()

for k in d:

val=d[k]

if val in inverse:

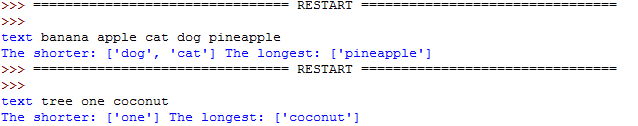
inverse[val].append(k)

else:

inverse[val]=[k]

print 'The shorter:',inverse[min(inverse)],'The longest:',inverse[max(inverse)]

Приклади:



Задача 5

Прочитати m груп (2<=m<=10) до n штук в групі (4<=n<=50) трицифрових цілих додатних чисел. Надрукувати ту групу чисел, яка містить найбільшу кількість простих чисел. Написати підпрограми (функції або/та процедури) для розпізнавання простоти, читання і друкування чисел.

Алгоритм:

Для цієї програми я написала чотири функції.

1. Функція читання чисел. Ця функція приймає два числа m та n. Перевіряє чи дані числа відповідають умові. Якщо ні, виводить на екран, що введені неправильні дані. Якщо числа введені правильно, то користувач в циклі вводить групи. Функція повертає матрицю t розмірністю m на n
2. Функція яка перевіряє чи число є простим. В циклі переглядаємо всі числа від 2 до заданого (не включно). Якщо дане число не ділиться на жодне з переглянутих ( остача від ділення не рівна 0), то функція повертає значення True. Інакше False.
3. Функція яка рахує кількість простих чисел в групі. Функція приймає масив; створюється зміна k –кількість простих чисел; в циклі перевіряє чи число є простим,якщо просте, то добавляє 1 до k; Функція повертає зміну k.
4. Функція яка друкує числа. Функція приймає двовимірний масив і повертає ту стрічку, в якій найбільше простих чисел.

В програмі я зчитую цілі числа m та n. Створюю матрицю за допомогою першої функції. В циклі переглядаю стрічки двовимірного масиву; Число в масиві а відповідає кількості простих чисел в стрічці, яка на тому ж індексі в двовимірному масиві. Вивожу результат за допомогою функції 4.

Код програми:

def importation(m,n): #function for inputing elements

if m>=2 and m<=10 and n>=4 and n<=50:

t = [ ]

mr=[]

for i in range(m):

row = raw\_input('Type the ' + str(n) + ' elements in group ' + str(i+1) + ': ').split()

if len(row)==n:

for i in range(len(row)):

row[i] = int(row[i])

if row[i]>99 and row[i]<1000:

if i==len(row)-1:

t.append(row)

else:

print'Error, you typed wrong data'

else:

print'Error, you typed wrong data'

else:

print'Error, you typed wrong data'

return t

#print importation(3,5)

def isprime(n):

if n == 1:

return False

for x in range(2, n):

if n % x == 0:

return False

else:

return True

#print isprime(7), isprime(12)

def ShowList(all\_list): #outputing data

f = all\_list[a.index(max(a))]

return f

def count\_primes(z):

k=0

for x in z:

if isprime(x):

k+=1

return k

#- - - - - - - - - - - - - - - - - - -

a = [ ]

print'Dear user, please enter those numbers: 2<=m<=10; 4<=n<=50; 99<elements in n<1000'

m = int(raw\_input('m = '))

n = int(raw\_input('n = '))

all\_list = importation(m,n)

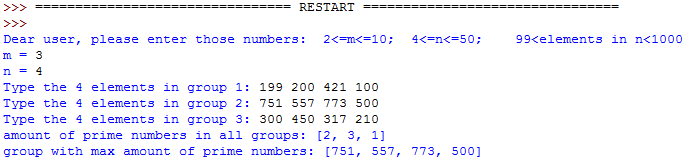
for i in all\_list:

a.append(count\_primes(i))

print 'amount of prime numbers in all groups:',a

print 'group with max amount of prime numbers:',ShowList(all\_list)

Приклади:



Задача 6

Описати рекурсивну функцію, яка б використовуючи схему Горнера обчислювала значення многочлена у заданій точці .

Алгоритм:

Вводимо x; вводимо кількість доданків; в циклі вводимо константи для кожного доданку; використовуючи рекурсивну функцію обчислюємо результат.

Ідея рекурсивної функції полягає в тому, що вона на певному етапі викликає саму себе. Обов’язково треба вказати умову зупинки, коли вона поверне певне точне значення. В моєму випадку умова зупинки –і на одиницю менше кількості доданків.

Код програми:

x=int(raw\_input('Enter x '))

n=int(raw\_input('Enter number of terms '))

a=[]

print 'Enter constants '

for i in range(n):

b=int(raw\_input())

a.append(b)

def horner(x,a,i=0):

if i==len(a)-1:

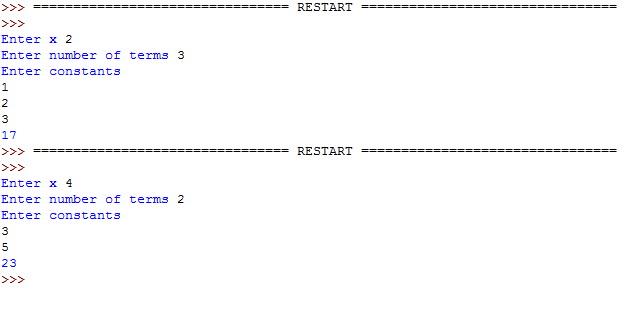
return a[i]

else:

return a[i]+horner(x,a,i+1)\*x

print horner(x,a)

Приклади:



Задача 7

Нехай задано натуральне число n. Поміняти порядок цифр числа n на зворотній (стрічок не використовувати).

Алгоритм:

Починаючи з цього завдання всі наступні програми будуть написані на С++. Кожна программа починається підключенням бібліотеки iostreаm:

#include<iostream>

using namespace std;

Вона дозволяє використовувати стандартні функції, як cin та cout.

Користувач вводить число n; створюємо змінну у-перевернуте число; цикл виконується поки число n не рівне 0; в циклі числу у присвоюємо його попереднє значення множене на 10 і додаємо до нього остачу від ділення числа n на 10 (тобто останню цифру числа n); число n ділимо на 10, щоб відкинути останню цифру; виводимо результат - число у.

Код програми:

#include<iostream>

using namespace std;

void main()

{

int n;

int y = 0;

cout << "Enter n" << endl;

cin >> n;

while (n != 0)

{

y = y \* 10 + n % 10;

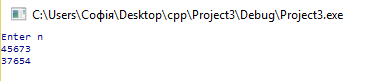
n /= 10;

}

cout << y << endl;

}

Приклади:



Задача 8

Прочитати m (2<=m<=10) груп до n (4<=n<=50) штук в групі трицифрових цілих додатних чисел. Надрукувати ту групу чисел, яка містить найбільшу кількість простих чисел. Написати підпрограми (функції або/та процедури) для розпізнавання простоти, читання і друкування чисел. Декларацію підпрограм оформити у вигляді \*.h файла, а код підпрограми розмістити в окремому \*.cpp файлі. Обидва файли додати до проекту. Код основної програми повинен містити лише виклики підпрограм.

Алгоритм:

Вводимо кількість груп і кількість чисел в групі; в циклі вводимо по черзі елемент; обчислюємо кількість додатних чисел кожної групи і друкуємо її.

Код програми:

cpp files:

#include<iostream>

using namespace std;

bool isPrime(int n)

{

if (n > 1)

{

for (int i = 2; i < n; i++)

if (n % i == 0)

return false;

return true;

}

return false;

}

#include<iostream>

using namespace std;

int\*\* readGroups()

{

int m, n;

cout << "Enter how many groups (2-10)" << endl;

cin >> m;

cout << "Enter how many numbers in group(4-50)" << endl;

cin >> n;

int \*\*groups;

groups = new int\*[m + 1];

groups[m] = NULL;

for (int i = 0; i < m; i++) {

groups[i] = new int[n + 1];

groups[i][n] = NULL;

}

for (int i = 0; i < m; i++)

{

cout << "Enter numbers (100-999) for group " << i + 1 << endl;

for (int j = 0; j < n; j++)

{

cin >> groups[i][j];

}

}

return groups;

#include<iostream>

#include<isPrime.h>

#include<readGroups.h>

using namespace std;

int MaxPrime(int\*\* gr)

{

int imax = 0;

int max = 0;

int\*\* groups = gr;

for (int i = 0; groups[i] != NULL; i++) {

int temp = 0;

for (int j = 0; groups[i][j] != NULL; j++) {

if (isPrime(groups[i][j]) == true) {

temp++;

}

}

if (temp > max) {

max = temp;

imax = i;

}

}

return imax;

}

#include<iostream>

#include<isPrime.h>

#include<readGroups.h>

#include<MaxPrime.h>

using namespace std;

void outres(int\*\* gr, int m)

{

int\*\* group = gr;

int maxi = m;

for (int i = 0; group[maxi][i] != NULL; i++)

cout << group[maxi][i] << " ";

}

#include<iostream>

#include<isPrime.h>

#include<readGroups.h>

#include<MaxPrime.h>

#include<outres.h>

using namespace std;

void main()

{

int\*\* groups = readGroups();

int imax = MaxPrime(groups);

outres(groups, imax);

system("pause");

}

header files:

#ifndef ISPRIME\_H

#define ISPRIME\_H

bool isPrime(int n);

#endif

#ifndef READGROUPS\_H

#define READGROUPS\_H

int\*\* readGroups();

#endif

#ifndef MAXPRIME\_H

#define MAXPRIME\_H

int MaxPrime(int\*\* gr);

#endif

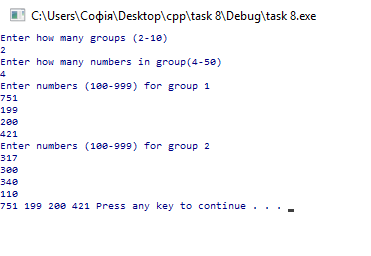
#ifndef OUTRES\_H

#define OUTRES\_H

void outres(int\*\* gr, int m);

#endif

Приклади:



Задача 9

Написати підпрограми (оформити з використанням \*.h файлів) для роботи з кількома стеками на основі двонаправленого списку. Зовні бібліотеки повинні бути видимі лише методи запису в стек(типу POP та PUSH) та модифікації вказівника стеку (типу роботи з SP). Використовуючи створені підпрограми в основній програмі прочитати масив цілих чисел та вивести в оберненому порядку спочатку числа на позиціях, що кратні трьом, а потім всі інші. Масив переглядати лише один раз.

Алгоритм:

Стек – це список елементів, який працює за принципом «останній прийшов – перший вийшов». Тобто забрати зі стеку можна тільки з останній елемент. Якщо ж ви хочете отримати елемент зі середини списку, спочатку доведеться витягнути всі елементи, що стоять після нього. Для цієї програми об’єктом, який зберігає стек, є клас (class). Він об’єднює інші об’єкти або змінні. Для роботи зі стеком я написала три основні функції: push, яка добавляє елемент до стеку і створює зв’язки з сусідніми елементами використовуючи клас Node; pop – повертає останній елемент зі стеку; empty – перевіряє чи пустий стек.

Код програми:

#include <iostream>

using namespace std;

class Node {

private:

int value;

Node \* left;

Node\* right;

public:

Node(int value) {

this->value = value;

this->left = NULL;

this->right = NULL;

}

Node(int value, Node\* left, Node\* right) {

this->value = value;

this->left = left;

this->right = right;

}

Node\* getLeft() {

return this->left;

}

void setLeft(Node\* left) {

this->left = left;

}

Node\* getRight() {

return this->right;

}

void setRight(Node\* right) {

this->right = right;

}

int getValue() {

return this->value;

}

void setValue(int value) {

this->value = value;

}

};

class Stack

{

private:

Node \*sp = NULL;

public:

Stack() {

}

void push(int value) {

Node\* temp = new Node(value);

temp->setValue(value);

if (sp == NULL) {

sp = temp;;

}

else {

sp->setRight(temp);

temp->setLeft(sp);

sp = temp;

}

}

int pop() {

if (sp != NULL) {

int result = sp->getValue();

Node\* temp = sp->getLeft();

if (temp != NULL) {

temp->setRight(NULL);

}

sp = temp;

return result;

}

}

int seek() {

return sp == NULL ? 0 : sp->getValue();

}

bool empty() {

return sp == NULL ? true : false;

}

}

int sizearr()

{

cout << "Enter size of array" << endl;

int size;

cin >> size;

return size;

}

int\* getarr(int size)

{

cout << "Enter values" << endl;

int\* arr = new int[size];

for (int i = 0; i < size; i++)

{

cin >> arr[i];

}

return arr;

}

void sortarr(int\* k, int size)

{

Stack\* stack1 = new Stack();

Stack\* stack2 = new Stack();

for (int i = 0; i < size; i++) {

if ((i % 3 == 0) && (i != 0))

stack1->push(k[i]);

else

stack2->push(k[i]);

}

while (!stack1->empty())

{

cout << stack1->pop() << " ";

}

while (!stack2->empty())

{

cout << stack2->pop() << " ";

}

}

void main() {

Stack\* stack1 = new Stack();

Stack\* stack2 = new Stack();

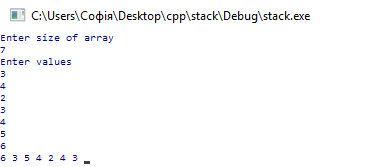
int z = sizearr();

int\* k = getarr(z);

sortarr(k, z);

}

Приклади:



Задача 10

Написати клас, який моделює роботу з множиною(використати масиви). Використовуючи розроблений клас написати програму, яка б заданому графу будувала новий граф, який є дзеркальним відображенням заданого(якщо в початковому графі присутнє ребро між двома вершинами, то в дзеркальному його немає і навпаки). Інформацію про граф захардкодити у коді.

Алгоритм:

Проходячи граф, за інформацією, яка міститься в коді надаємо існуючим ребрам значення 1, а не існуючим – 0;в циклі проходимо матрицю графа і замінюємо 0 на1, а 1 на 0; виводимо нові ребра.

Код програми:

#include <iostream>

#define MAX 4

using namespace std;

struct Top

{

int top;

int edge[MAX];

Top()

{

for (int i = 0; i < MAX; i++)

{

edge[i] = 0;

}

}

void set(int \_edge)

{

edge[\_edge] = 1;

}

};

class Earl

{

public:

Top count[MAX];

Earl()

{

for (int i = 0; i < MAX; i++)

{

count[i].top = i;

}

}

void set(int \_top, int \_edge)

{

count[\_top].set(\_edge);

}

void view()

{

for (int i = 0; i < MAX; i++)

{

cout << "Earl " << i << ": \n";

for (int j = 0; j < MAX; j++)

{

if (count[i].edge[j] == 1)

{

cout << "( " << i << " , " << j << " )\n";

}

}

}

}

void reset()

{

for (int i = 0; i < MAX; i++)

{

for (int j = 0; j < MAX; j++)

{

if (count[i].edge[j] == 1)

{

count[i].edge[j] = 0;

}

else

{

count[i].edge[j] = 1;

}

}

}

}

};

void main()

{

Earl test;

test.set(1, 3);

test.set(1, 2);

test.set(2, 3);

test.set(3, 1);

test.set(0, 3);

test.view();

test.reset();

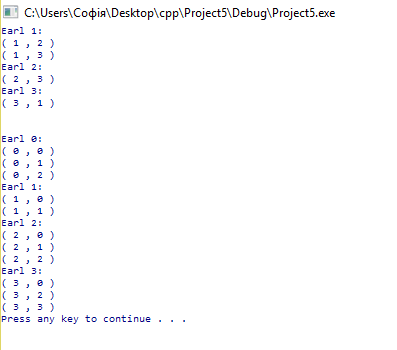
cout << "\n\n";

test.view();

system("pause");

}

Приклади:



Задача 11

Прочитати текстову стрічку довжиною не більше ніж 253 символи. Замінити кожне входження словосполучення ph на f. Використовувати лише функції обробки стрічок мови С. Контейнер String та його методи не використовувати.

Алгоритм:

Створюємо два масива char: перший містить стрічку яку ми вводимо, другий - нова стрічка; вводимо стрічку за допомогою функції get.s; в циклі переглядаємо введену стрічку; якщо буква рівна p і наступна буква рівнa h, добавляємо в новий масив f; інші букви просто переносяться в циклі з старого в новий масив; зміна j рахує індекс нової стрічки, бо в новій стрічці зменшується кількість символів; виводимо нову стрічку.

Код програми:

#include<iostream>

using namespace std;

void main(){

char mystring[253];

char newstring[253];

cout << "Enter string "<<endl;

gets\_s (mystring);

char\* a= new char[4];

a[0] = 'p';

a[1] = 'h';

a[2] = 'f';

a[3] = '\0';

int j = 0;

for (int i = 0; i < strlen(mystring); i++)

{

if (mystring[i] == a[0] || mystring[i + 1] == a[1])

{

newstring[j] = a[2];

j++;

}

else if (mystring[i] != a[1])

{

newstring[j] = mystring[i];

j++;

}

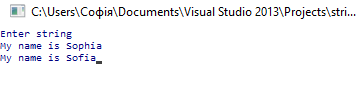
}

newstring[j] = a[3];

cout << newstring;

}

Приклади:



Висновок

Написавши 11 програм, я навчилась правильно будувати програму за допомогою різних алгоритмів. Починаючи з простих завдань на Python, я поступово переходила до складніших на С++, які все важче і важче було реалізовувати.

Для початківців, людей, які тільки починають програмувати, Python є ідеальною мовою. Його синтаксис дуже простий і нам не треба турбуватись про багато речей (наприклад, як дані виглядають в пам’яті), адже інтерпретатор робить це замість нас.

На С++ були програми складніші, що означає більші можливості цієї мови порівняно з Python. Ми можемо створювати структури, використовувати стеки, хідер файли, С++ дає можливість подивитись глибше, що робиться в процесорі при обробці коду. Як на мене, синтаксис будь-якої мови можна легко вивчити, але важливішим є розуміти як саме працює код і як виконуються процеси в компіляторі.

Список літератури

Лутц М . Програмирование Python. Том 2. / М. Лутц. – Санкт-Петербург : Издат. “Cимвол-Плюс” , 2011. – 348c.

Лафоре Р. “ Об’єктно-орієнтоване програмування . Том 4 . ” / Р. Лафоре. – , 2004. – 972c.

Swaroop C. A Byte of Python./ С. Swaroop. – , 2012. – 157с.

Дейтел Х. Как программировать на С++./ Х. Дейтел П. Дейтел. – , 2008. – 1454с.