Documetație

Problema 1 Numere complexe:

Creați un program care lucrează cu numere complexe (a + bi). Programul

gestionează o listă de numere complexe și permite efectuarea repetată a

următoarelor acțiuni:

1. Adaugă număr în listă.

• Adaugă număr complex la sfârșitul listei

• Inserare număr complex pe o poziție dată.

2. Modifică elemente din listă.

• Șterge element de pe o poziție dată.

• Șterge elementele de pe un interval de poziții.

• Înlocuiește toate aparițiile unui număr complex cu un alt număr

complex.

3. Căutare numere.

• Tipărește partea imaginara pentru numerele din listă. Se dă intervalul

de poziții (sub secvența).

• Tipărește toate numerele complexe care au modulul mai mic decât 10

• Tipărește toate numerele complexe care au modulul egal cu 10

4. Operații cu numerele din listă

• suma numerelor dintr-o subsecventă dată (se da poziția de început și

sfârșit).

• Produsul numerelor dintr-o subsecventă dată (se da poziția de început

și sfârșit).

• Tipărește lista sortată descrescător după partea imaginara

5. Filtrare.

• Filtrare parte reala prim – elimină din listă numerele complexe la care

partea reala este prim.

• Filtrare modul – elimina din lista numerele complexe la care modulul

este <,= sau > decât un număr dat.

6. Undo

• Reface ultima operație (lista de numere revine la numerele ce existau

înainte de ultima operație care a modificat lista)

Plan iterații:

-Iterația 1:

terminare funcționalitățile 1,2,3(varianta procedurală) + documentație

-Iterația 2:

terminare funcționalități 4,5,6 (proiect final variantă modulară) + documentație finală

-Iterația 3:

optimizări extra pentru posibile probleme

Notă: Toate numere complexe vor fi stocate intr-o listă numită numere\_complexe, inițial vidă, declarată la începutul programului, numere complexe fiind reprezentate sub forma unui tuplu de tipul (real, imag), unde real este partea imaginara si imag partea imaginara a numarului. Exsita si o lista numita undo\_list folosita pentru functia undo din functionalitatea 6

Funcții folosite general

|  |  |
| --- | --- |
| Denumire | Scop |
| alegere\_optiune() | Folosit pentru a nagiva interfata |
| citire\_nr\_complex() | Folosit pentru citirea unui numar complex de la tastura returnand numarul format sub forma unui tuplu |
| citire\_index() | Folosit pentru a putea accesa secvente sau anumite pozitii dintr-un sir |
| afisare\_lista(lista) | Afiseaza pe ecran o lista data |
| get\_parte\_reala(numar) | Returneaza partea reala a numarului complex numar |
| get\_parte\_imaginara(numar) | Returneaza partea imaginara a numarului complex numar |
| creare\_nr\_complex(real, imag) | Creeaza si returneaza un numar complex format din real si imag sub forma unui tuplu |
| suma\_numere\_complexe(numar\_1, numar\_2) | Calculeaza si returneaza suma dintre numerele complexe numar\_1 si numar\_2 |
| produs\_numere\_complexe(numar\_1, numar\_2) | Calculeaza si returneaza produsul dintre numere complexe numar\_1 si numar\_2 |
| modul\_numar\_complex(numar) | Calculeaza si returneaza modulul numarului complex numar |
| citire\_valoare() | citeste si returneaza o valoare de tipul int |

1. **Adaugă număr în listă:**

Funcții

|  |  |
| --- | --- |
| Denumire | Descriere |
| add\_la\_final(numar, lista) | Adauga la finalul listei date prin parametru un numar complex dat prin parametrul numar |
| add\_cu\_index(numar, index, lista) | Inserează pe pozitie data prin parametrul index in lista data prin parametru un numar complex dat prin parametrul numar |

Scenariu de rulare pentru adăugarea numărului 13+14j la finalul listei:

* Afisare meniu principal
* p = 1 (user input prin alegere\_optiune() )
* afisare meniu functionalitate 1
* p = 1(user input prin alegere\_optiune() )
* apelare citire\_nr\_complex()
* input1 = (13,14) din citire\_nr\_complex()
* apelare add\_la\_final(input\_1, numere\_complexe)
* la finalul listei numere\_complexe va fi adaugat numarul (13, 14)

Scenariu de rulare pentru adăugarea numărului 13+14i pe poziția 3:

* Afisare meniu principal
* p = 1 (user input prin alegere\_optiune() )
* afisare meniu functionalitate 1
* p = 2(user input prin alegere\_optiune() )
* index = 3 prin citire\_index()
* input2 = (13,14) print citire\_nr\_complex()
* apelare add\_cu\_index(input2, index, numere\_complexe)
* se insereaza inaintea pozitei 3 numarul complex (13, 14)

1. **Modifică elemente din listă:**

Funcții utilizate

|  |  |
| --- | --- |
| Denumire | Descriere |
| stergere\_numar(index, lista) | Sterge numarul de pe o pozitie data prin index in lista data; in cazul in care pe pozitia data nu exista un element se ridica o eroare si se afiseaza un mesaj |
| stergere\_interval(start, stop, lista) | Sterge numere dintr-un intervat dat prin start stop din lista data |
| inlocuire\_numar(inlocuit, inlocuitor, lista) | Se inlocuieste fiecare aparitie a numarului inlocui cu numarul inlocuitor in lista data si returneaza intr-o lista pozitia fiecarei aparitii inlocuitului |

Scenariu rulare pentru stergerea elementului de pe pozitia 2 din lista [(1, 2), (3, 4), (5, 6), (7, 8)]:

* Apelare main()
* p = 2 prin alegere\_optiune()
* p = 1 prin alegere\_optiune()
* index = 2 prin citire\_index()
* apelare stergere\_numar(index, numere\_complexe)
* numarul (5, 6) va fi sters de pe pozita 2
* lista finala va fi [(1, 2), (3, 4), (7, 8)]

Scenariu rulare pentru stergerea elementelor din intervalul [1,3) din lista [(1, 2), (3, 4), (5, 6), (7, 8)]:

* Apelare main()
* p = 2 prin alegere\_optiune()
* p = 2 prin alegere\_optiune()
* start = 1 prin citire\_index()
* stop = 3 prin citire\_index()
* apelare stergere\_interval(start, stop, numere\_complexe)
* numerele (3, 4), (5, 6) vor fi sterse de pe pozitiile 1,2
* lista finala va fi [(1, 2), (7, 8)]

Scenariu de rulare pentru inlocuirea numarului (1, 2) cu (3, 4) din lista [(1, 2), (5, 7), (1, 2), (24, 13), (1, 2)]:

* Apelare main()
* p = 2 prin alegere\_optiune()
* p = 2 prin alegere\_optiune()
* inlocuit = (1, 2) prin citire\_nr\_complex()
* inlocuitor = (3, 4) print citire\_nr\_complex()
* apelare inlocuire\_numar(inlocuit, inlocuitor, numere\_complexe)
* i = 0 lista[i]==(1, 2) = > lista[i] = (3, 4) , lista\_aux[0] = 0
* i = 2 lista[i]==(1, 2) = > lista[i] = (3, 4) , lista\_aux[1] = 2
* i = 4 lista[i]==(1, 2) = > lista[i] = (3 ,4) , lista\_aux[2] = 4
* lista din final va fi [(3, 4), (5, 7), (3, 4), (24, 13), (3, 4)]

**3.Căutare numere**

Funcții utilizate

|  |  |
| --- | --- |
| Denumire | Descriere |
| parte\_imag(start, stop, lista) | Returneaza o lista formata din partile imaginare din numerele din lista data din secventa data de start si stop |
| modul\_mai\_mic\_ca10(lista) | Returneaza o lista formata din toate numerele complexe cu modulul mai mic ca 10 din lista data |
| modul\_egal\_cu10(lista) | Returneaza o lista formata din toate numerele complexe cu modulul egal cu 10 din lista data |

Scenariu de rulare pentru afisarea parti imaginare a numerelor din secventa 0, 3 din lista [(1, 2), (3, 4), (7, 8)]:

* apelare main()
* p = 3 din alegere\_optiune()
* p = 1 din alegere\_optiune()
* start = 0 din citire\_index()
* stop = 3 din citire\_index()
* apelare parte\_imag(start, stop, numere\_complexe)
* i = 0, lista\_aux[0] = 2.0
* i = 1, lista\_aux[1] = 4.0
* i = 2, lista\_aux[2] = 8.0
* se afiseaza lista\_aux == [2, 4, 8]

Scenariu de rulare pentru afisarea numerelor complexe cu modulul mai mic ca 10 din lista [(1, 2), (3, 4), (6, 8)]:

* apelare main()
* p = 3 din alegere\_optiune()
* p = 2 din alegere\_optiune()
* apelare modul\_mai\_mic\_ca10(numere\_complexe)
* item = (1, 2) cu modul mai mic ca 10, lista\_aux[0] = (1, 2)
* item = (3, 4) cu modul mai mic ca 10, lista\_aux[1] = (3, 4)
* se afiseaza lista\_aux = [(1, 2), (3, 4)]

Scenariu de rulare pentru afisarea numerelor complexe cu modulul egal cu 10 din lista [(6, 8), (8, 6), (2, 3), (5, 6)]:

* apelare main()
* p = 3 din alegere\_optiune()
* p = 3 din alegere\_optiune()
* apelare modul\_egal\_cu10(numere\_complexe)
* item = (6, 8) cu modul egal cu 10, lista\_aux[0] = (6, 8)
* item = (8, 6) co modul egal cu 10, lista\_axu[1] = (8, 6)
* se afiseaza lista\_aux = [(6, 8), (8, 6)]

**4.** **Operații cu numerele din listă**

Funcții utilizate

|  |  |
| --- | --- |
| Denumire | Descriere |
| suma\_secventa(start, stop, lista) | Calculeaza si returneaza suma numerelor complexe din intervalul dat de start si stop din lista data |
| produs\_secventa(start, stop, lista) | Calculeaza si returneaza produsul numerelor complexe din intervalul dat de start si stop din lista data |
| afisare\_descrescatoare(lista) | Afiseaza lista in ordine descrescatoare dupa partea imaginara a numerelor din lista |

Scenariu de rulare pentru calcularea suma numerelor din intervalul [0,3) din lista [(1, 2), (4, 5), (7, 8), (6, 9), (15, 16)]:

|  |
| --- |
| Apelare main() din fisierul main |
| p = 4 din alegere\_optiune() |
| Apelare main() din fisierul func4 |
| p = 1 din alegere\_optiune() |
| start = 0 din citire\_index() |
| stop = 3 din citire\_index() |
| Apelare suma\_secventa(start, stop, numere\_complexe) |
| se vor parcurge elementele de pe pozitiile 0,1,2 , suma lor fiind retinuta in variabila suma |
| Suma este returnata si afisata pe ecran, suma = (12, 15) |

Scenariu pentru calcularea produsul numerelor din intervalul [0,3) din lista [(1, 2), (4, 5), (7, 8), (6, 9), (15, 16)]:

|  |
| --- |
| Apelare main() din fisierul main |
| p = 4 din alegere\_optiune() |
| Apelare main() din fisierul func4 |
| p = 2 din alegere\_optiune() |
| start = 0 din citire\_index() |
| stop = 3 din citire\_index() |
| Apelare produs\_secventa(start, stop, numere\_complexe) |
| se vor parcurge elementele de pe pozitiile 0,1,2 , produsul lor fiind retinut in variabila produs |
| Produsul este returnat si afisat pe ecran, produs = (-146, 43) |

Scenariu de rulare pentru tiparirea in ordine descrescatoare dupa partea imaginara a numerelor din lista [(1, 2), (4, 5), (7, 8), (6, 9), (15, 16)]:

|  |
| --- |
| Apelare main() din fisierul main |
| p = 4 din alegere\_optiune() |
| Apelare main() din fisierul func4 |
| p = 3 din alegere\_optiune() |
| Apelare afisare\_descrescatoare(numere\_complexe) |
| lista va fi sortata descrescator dupa partea imaginara printr-o functie predefinita, sortarea fiind stocata intr-o lista auxiliara pentru a nu modifica lista originala |
| Lista auxiliara este returnata si afisata , lista\_aux = [(15, 16), (6, 9), (7, 8), (4, 5), (1, 2)] |

**5.Filtrare**

Funcții utilizate

|  |  |
| --- | --- |
| Denumire | Descriere |
| este\_prim(numar) | Verifica primalitatea unui numar |
| filtru\_parte\_reala\_prim(lista) | Elimina fiecare numar complex cu partea reala un numar prim dintr-o lista data |
| filtrare\_modul(optiune, numar, lista) | Depinzand de valoarea din optiune, functia elimina din lista data fiecare numar complex ce are modulul mai mare, egal, sau mai mic cu un numar dat |

Scenariu de rulare pentru filtrarea listei [(2, 6), (8, 9), (3, 5), (9, 3), (1, 2)] de numerele complexe cu partea reala un numar prim:

|  |
| --- |
| Apelare main() din fisierul main |
| p = 5 din alegere\_optiune() |
| p = 1 din alegere\_otiune() |
| apelare filtru\_parte\_reala\_prim(numere\_complexe) |
| Se pargurge lista si se va verifica fiecare element daca are partea reala un numar prim cu ajutorul functiei este\_prim() |
| Se vor gasi numerele (2, 6), (3, 5), si vor fi eliminate |
| Se retin intr-o lista auxiliara pozitiile elementelor eliminate |
| Lista va fi returnata de functie pentru testare |

Scenariu de rulare pentru eliminarea tuturor numerelor complexe cu modulul egal din lista [(6, 8), (8, 6), (2, 3), (5, 6)]:

|  |
| --- |
| Apelare main() din fisierul main |
| p = 5 din alegere\_optiune() |
| Apelare main() din fisierul func5 |
| p = 2 din alegere\_optiune() |
| nr = 10 din citire\_valoare() |
| p = 2 din alegere\_optiune() |
| Apelare filtrare\_modul(optiune, numar, numere\_complexe) |
| Optiunea fiind doi, se cauta numerele complexe cu modulul egal cu 10 |
| Sunt gasite numerele (6, 8) si (8, 6) si vor fi eliminate |
| Lista finala va fi [(2, 3), (5, 6)] |

**6.Undo**

La inceputul fiecarei functii din aplicatie ce modifica lista de stocare, lista va fi copiata in lista undo\_list

Functii utilizate

|  |  |
| --- | --- |
| Denumire | Descriere |
| undo(lista) | Functia readuce lista data la o valoare inainte de ultima modifcare efectuata |

Scenariu de rulare pentru undo a listei [(2, 6), (8, 9), (3, 5), (9, 3), (1, 2)] dupa ce a fost modificata folosit a doua optiune din functionalitatea 5:

|  |
| --- |
| Apelare main() din fisierul main |
| p = 5 din alegere\_optiune() |
| Apelare main() din fisierul func5 |
| p = 2 din alegere\_main() |
| undo\_list va primi [(2, 6), (8, 9), (3, 5), (9, 3), (1, 2)] |
| lista va fi modificata conform functionalitatii, ea devenind la final va fi [(8, 9), (9, 3), (1, 2)] |
| p = 3 din alegere\_optiune() |
| p = 6 din alegere\_optiune() |
| Apelare main() din func6 |
| p = 1 din alegere\_optiune() |
| Apelare undo(lista) |
| Lista va primi continutul din undo\_list, la final fiind [(2, 6), (8, 9), (3, 5), (9, 3), (1, 2)] |