UNIVERSITATEA DE STAT DIN MOLDOVA
FACULTATEA DE MATEMATICĂ ȘI INFORMATICĂ
SPECIALITATEA INFORMATICA

RAPORT

la disciplina „Algoritme, Structuri de Date și Complexitate”

Lucrarea de laborator nr. 3: Structuri dinamice de date

Conducătorul științific: Novac Ludmila, dr. conf. univ.
Autor: Cemîrtan Cristian, student din grupa I2101

CHIȘINĂU – 2023

Cuprins

Preliminări

Pentru a simplifica reprezentarea tipurilor de date dinamice, am implementat tehnica polimorfismului în limbajul C prin intermediul unei declarație de structură ce conține pointeri către funcții. Prin polimorfism, putem specifica moștenirea între tipuri de date (listă dublu înlănțuită extinde lista simplu înlănțuită).

Pentru a măsura consumul de memorie utilizat de fiecare tip dinamic de date, am implementat propria mea utilitate de gestionare a memoriei. Această utilitate contorizează numărul alocărilor de memorie într-o sesiune de lucru.

De asemenea, fișierul textual utilizat în cadrul lucrării de laborator a fost preluat din lucrarea precedentă.

Testarea tipurilor dinamice de date

Listă simplu înlănțuită.

Inițial, lista goală a alocat 8 octeți din memorie. După inserarea celor cinci elemente din fișier în listă, s-au alocat încă 80 octeți. Ștergerea sau căutarea unui element nu crește utilizarea memoriei. Deci, un singur nod din listă ocupă 16 octeți din memorie.

```
>mem
Amprenta de memorie: 1 alocari (8 bytes)

>insert
Nr. de elemente de inserat: 5
INSERT id = 5
succes

INSERT id = 1
Pozitia inserarii:
1. La inceputul listei;
2. La sfarsitul listei;
3. Dupa un element dat.
>1
succes

INSERT id = 2
Pozitia inserarii:
1. La inceputul listei;
2. La sfarsitul listei;
3. Dupa un element dat.
>3
WHERE id = 1
succes

INSERT id = 3
Pozitia inserarii:
1. La inceputul listei;
2. La sfarsitul listei;
3. Dupa un element dat.
>3
WHERE id = 2
succes

INSERT id = 4
Pozitia inserarii:
1. La inceputul listei;
2. La sfarsitul listei;
3. Dupa un element dat.
>3
WHERE id = 3
succes


>select
1,Genevra,Merriman,gmerriman0@dmoz.org,F,3409.550000,106.195.34.151,PL
2,Kellie,Colles,kcolles1@reuters.com,F,1209.730000,99.153.26.131,TN
3,Brooke,Causton,bcauston2@scientificamerican.com,F,3998.020000,13.32.196.48,BR
4,Rees,Durston,rdurston3@mayoclinic.com,M,1298.700000,67.126.118.194,PL
```

```
5,Jerad,Winpenny,jwinpenny4@360.cn,M,1164.350000,21.143.138.69,MP

>mem
Amprenta de memorie: 6 alocari (88 bytes)

>erase
WHERE id = 3

>find
WHERE id = 5
5,Jerad,Winpenny,jwinpenny4@360.cn,M,1164.350000,21.143.138.69,MP

>select
1,Genevra,Merriman,gmerriman0@dmoz.org,F,3409.550000,106.195.34.151,PL
2,Kellie,Colles,kcolles1@reuters.com,F,1209.730000,99.153.26.131,TN
4,Rees,Durston,rdurston3@mayoclinic.com,M,1298.700000,67.126.118.194,PL
5,Jerad,Winpenny,jwinpenny4@360.cn,M,1164.350000,21.143.138.69,MP

>mem
Amprenta de memorie: 6 alocari (88 bytes)
```

Figura 1

Listă dublu înlănțuită.

Operațiile sunt asemănătoare cu cele suportate de lista simplu înlănțuită, dar un singur nod ocupă dublu din memorie (16 octeți din 8).

```
>mem
Amprenta de memorie: 1 alocari (8 bytes)

>insert
Nr. de elemente de inserat: 5
INSERT id = 5
succes

INSERT id = 1
Pozitia inserarii:
1. La inceputul listei;
2. La sfarsitul listei;
3. Dupa un element dat.
>1
succes

INSERT id = 2
Pozitia inserarii:
1. La inceputul listei;
2. La sfarsitul listei;
3. Dupa un element dat.
>3
WHERE id = 1
succes

INSERT id = 3
Pozitia inserarii:
1. La inceputul listei;
2. La sfarsitul listei;
3. Dupa un element dat.
```

```
>3
WHERE id = 2
succes

INSERT id = 4
Pozitia inserarii:
1. La inceputul listei;
2. La sfarsitul listei;
3. Dupa un element dat.
>3
WHERE id = 3
succes


>select
1,Genevra,Merriman,gmerriman0@dmoz.org,F,3409.550000,106.195.34.151,PL
2,Kellie,Colles,kcolles1@reuters.com,F,1209.730000,99.153.26.131,TN
3,Brooke,Causton,bcauston2@scientificamerican.com,F,3998.020000,13.32.196.48,BR
4,Rees,Durston,rdurston3@mayoclinic.com,M,1298.700000,67.126.118.194,PL
5,Jerad,Winpenny,jwinpenny4@360.cn,M,1164.350000,21.143.138.69,MP

>mem
Amprenta de memorie: 6 alocari (128 bytes)

>erase
WHERE id = 3

>find
WHERE id = 5
5,Jerad,Winpenny,jwinpenny4@360.cn,M,1164.350000,21.143.138.69,MP

>select
1,Genevra,Merriman,gmerriman0@dmoz.org,F,3409.550000,106.195.34.151,PL
2,Kellie,Colles,kcolles1@reuters.com,F,1209.730000,99.153.26.131,TN
4,Rees,Durston,rdurston3@mayoclinic.com,M,1298.700000,67.126.118.194,PL
5,Jerad,Winpenny,jwinpenny4@360.cn,M,1164.350000,21.143.138.69,MP

>mem
Amprenta de memorie: 6 alocari (128 bytes)
```

Figura 2

Listă simplu înlănțuită circulară.

Consumul de memorie și operațiile asupra listei rămân identice, dar faptul că succesorul ultimului nod este primul (nu valoarea nulă), putem specifica numărul de iterații pentru a parcurge lista.

```
...
>select
Nr. de iteratii: 10
1,Genevra,Merriman,gmerriman0@dmoz.org,F,3409.550000,106.195.34.151,PL
2,Kellie,Colles,kcolles1@reuters.com,F,1209.730000,99.153.26.131,TN
4,Rees,Durston,rdurston3@mayoclinic.com,M,1298.700000,67.126.118.194,PL
5,Jerad,Winpenny,jwinpenny4@360.cn,M,1164.350000,21.143.138.69,MP
1,Genevra,Merriman,gmerriman0@dmoz.org,F,3409.550000,106.195.34.151,PL
2,Kellie,Colles,kcolles1@reuters.com,F,1209.730000,99.153.26.131,TN
4,Rees,Durston,rdurston3@mayoclinic.com,M,1298.700000,67.126.118.194,PL
```

```
5,Jerad,Winpenny,jwinpenny4@360.cn,M,1164.350000,21.143.138.69,MP
1,Genevra,Merriman,gmerriman0@dmoz.org,F,3409.550000,106.195.34.151,PL
2,Kellie,Colles,kcolles1@reuters.com,F,1209.730000,99.153.26.131,TN
```

Figura 3

Listă dublu înlănțuită circulară.

În deosebire de lista simplu înlănțuită circulară, putem specifica un număr negativ de iterații, pentru a parcurge invers lista (începând cu primul nod).

```
>select
Nr. de iteratii: -10
1,Genevra,Merriman,gmerriman0@dmoz.org,F,3409.550000,106.195.34.151,PL
5,Jerad,Winpenny,jwinpenny4@360.cn,M,1164.350000,21.143.138.69,MP
4,Rees,Durston,rdurston3@mayoclinic.com,M,1298.700000,67.126.118.194,PL
2,Kellie,Colles,kcolles1@reuters.com,F,1209.730000,99.153.26.131,TN
1,Genevra,Merriman,gmerriman0@dmoz.org,F,3409.550000,106.195.34.151,PL
5,Jerad,Winpenny,jwinpenny4@360.cn,M,1164.350000,21.143.138.69,MP
4,Rees,Durston,rdurston3@mayoclinic.com,M,1298.700000,67.126.118.194,PL
2,Kellie,Colles,kcolles1@reuters.com,F,1209.730000,99.153.26.131,TN
1,Genevra,Merriman,gmerriman0@dmoz.org,F,3409.550000,106.195.34.151,PL
5,Jerad,Winpenny,jwinpenny4@360.cn,M,1164.350000,21.143.138.69,MP
```

Figura 4

Stivă implementată prin listă simplu înlănțuită.

Parcurgerea unei stive implică modificarea stării interne a listei, deci utilizarea memoriei crește.

```
>mem
Amprenta de memorie: 1 alocari (8 bytes)

>insert
Nr. de elemente de inserat: 5
INSERT id = 5
succes

INSERT id = 4
succes

INSERT id = 3
succes

INSERT id = 2
succes

INSERT id = 1
succes


>mem
Amprenta de memorie: 6 alocari (88 bytes)

>find
1,Genevra,Merriman,gmerriman0@dmoz.org,F,3409.550000,106.195.34.151,PL
```

```
>erase

>mem
Amprenta de memorie: 6 alocari (88 bytes)

>select
2,Kellie,Colles,kcolles1@reuters.com,F,1209.730000,99.153.26.131,TN
3,Brooke,Causton,bcauston2@scientificamerican.com,F,3998.020000,13.32.196.48,BR
4,Rees,Durston,rdurston3@mayoclinic.com,M,1298.700000,67.126.118.194,PL
5,Jerad,Winpenny,jwinpenny4@360.cn,M,1164.350000,21.143.138.69,MP

>mem
Amprenta de memorie: 10 alocari (152 bytes)
```

Figura 5

Stivă implementată prin buffer.

În schimb, utilizarea memoriei este ridicată, dar numărul alocărilor de memorie este scăzută. Parcurgerea stivei nu implică alocări adiționale.

```
>mem
Amprenta de memorie: 2 alocari (88 bytes)

>insert
Nr. de elemente de inserat: 5
INSERT id = 5
succes

INSERT id = 4
succes

INSERT id = 3
succes

INSERT id = 2
succes

INSERT id = 1
succes


>mem
Amprenta de memorie: 2 alocari (88 bytes)

>select
1,Genevra,Merriman,gmerriman0@dmoz.org,F,3409.550000,106.195.34.151,PL
2,Kellie,Colles,kcolles1@reuters.com,F,1209.730000,99.153.26.131,TN
3,Brooke,Causton,bcauston2@scientificamerican.com,F,3998.020000,13.32.196.48,BR
4,Rees,Durston,rdurston3@mayoclinic.com,M,1298.700000,67.126.118.194,PL
5,Jerad,Winpenny,jwinpenny4@360.cn,M,1164.350000,21.143.138.69,MP

>mem
Amprenta de memorie: 2 alocari (88 bytes)

>insert
Nr. de elemente de inserat: 4
```

```
INSERT id = 14
succes

INSERT id = 13
succes

INSERT id = 12
succes

INSERT id = 11
succes


>mem
Amprenta de memorie: 3 alocari (216 bytes)

>select
11,Jasmin,Leachman,jleachmana@berkeley.edu,F,235.250000,173.16.160.41,CN
12,Ira,Beechcraft,ibeechcraftb@answers.com,F,4966.550000,135.104.120.68,BD
13,Britni,Mapam,bmapamc@gravatar.com,F,1056.860000,254.85.200.12,CN
14,Ariana,Klees,akleesd@google.es,F,1228.510000,10.179.129.143,KZ
1,Genevra,Merriman,gmerriman0@dmoz.org,F,3409.550000,106.195.34.151,PL
2,Kellie,Colles,kcolles1@reuters.com,F,1209.730000,99.153.26.131,TN
3,Brooke,Causton,bcauston2@scientificamerican.com,F,3998.020000,13.32.196.48,BR
4,Rees,Durston,rdurston3@mayoclinic.com,M,1298.700000,67.126.118.194,PL
5,Jerad,Winpenny,jwinpenny4@360.cn,M,1164.350000,21.143.138.69,MP

>mem
Amprenta de memorie: 3 alocari (216 bytes)
```

Figura 6

Coadă implementată prin listă simplu înlănțuită.

Alocarea inițială de memorie s-a dublat cu scopul de a păstra al doilea pointer la coada listei. Parcurgerea cozii tot implică un număr adițional de alocări.

```
>mem
Amprenta de memorie: 1 alocari (16 bytes)

>insert
Nr. de elemente de inserat: 5
INSERT id = 1
succes

INSERT id = 2
succes

INSERT id = 3
succes

INSERT id = 4
succes

INSERT id = 5
succes
```

```
>mem
Amprenta de memorie: 6 alocari (96 bytes)

>find
Obtinerea elementului:
1. Primul element (cap);
2. Ultimul element (coada).
>1
1,Genevra,Merriman,gmerriman0@dmoz.org,F,3409.550000,106.195.34.151,PL

>find
Obtinerea elementului:
1. Primul element (cap);
2. Ultimul element (coada).
>2
5,Jerad,Winpenny,jwinpenny4@360.cn,M,1164.350000,21.143.138.69,MP

>erase

>find
Obtinerea elementului:
1. Primul element (cap);
2. Ultimul element (coada).
>1
2,Kellie,Colles,kcolles1@reuters.com,F,1209.730000,99.153.26.131,TN

>mem
Amprenta de memorie: 6 alocari (96 bytes)

>select
2,Kellie,Colles,kcolles1@reuters.com,F,1209.730000,99.153.26.131,TN
3,Brooke,Causton,bcauston2@scientificamerican.com,F,3998.020000,13.32.196.48,BR
4,Rees,Durston,rdurston3@mayoclinic.com,M,1298.700000,67.126.118.194,PL
5,Jerad,Winpenny,jwinpenny4@360.cn,M,1164.350000,21.143.138.69,MP

>mem
Amprenta de memorie: 10 alocari (160 bytes)
```

Figura 7

Coadă implementată prin buffer ciclic.

În comparație cu stiva implementată prin buffer, alocarea inițială a crescut cu 8 octeți. De asemenea, în deosebire de liste înlănțuite, implementarea unei structuri de date prin buffer reduce numărul de alocări (la zero în cazul parcurgerii), dar crește utilizarea memoriei în general.

```
>mem
Amprenta de memorie: 2 alocari (96 bytes)

>insert
Nr. de elemente de inserat: 8
INSERT id = 1
succes

INSERT id = 2
succes

INSERT id = 3
```

```
succes

INSERT id = 4
succes

INSERT id = 5
succes

INSERT id = 6
succes

INSERT id = 7
succes

INSERT id = 8
succes


>mem
Amprenta de memorie: 2 alocari (96 bytes)

>insert
Nr. de elemente de inserat: 1
INSERT id = 9
succes


>mem
Amprenta de memorie: 3 alocari (224 bytes)

>find
Obtinerea elementului:
1. Primul element (cap);
2. Ultimul element (coada).
>1
1,Genevra,Merriman,gmerriman0@dmoz.org,F,3409.550000,106.195.34.151,PL

>find
Obtinerea elementului:
1. Primul element (cap);
2. Ultimul element (coada).
>2
9,Damara,Entreis,dentreis8@wikimedia.org,F,533.510000,225.68.220.80,PH

>select
1,Genevra,Merriman,gmerriman0@dmoz.org,F,3409.550000,106.195.34.151,PL
2,Kellie,Colles,kcolles1@reuters.com,F,1209.730000,99.153.26.131,TN
3,Brooke,Causton,bcauston2@scientificamerican.com,F,3998.020000,13.32.196.48,BR
4,Rees,Durston,rdurston3@mayoclinic.com,M,1298.700000,67.126.118.194,PL
5,Jerad,Winpenny,jwinpenny4@360.cn,M,1164.350000,21.143.138.69,MP
6,Retha,Broome,rbroome5@ox.ac.uk,F,1264.290000,156.53.2.206,CZ
7,Ross,Blewis,rblewis6@scribd.com,M,112.070000,158.139.134.165,CL
8,Lindy,Alberts,lalberts7@unicef.org,F,4263.470000,35.63.99.129,CN
9,Damara,Entreis,dentreis8@wikimedia.org,F,533.510000,225.68.220.80,PH

>mem
Amprenta de memorie: 3 alocari (224 bytes)
```

Figura 8

Arbore binar de căutare.

Considerăm următoarea arbore:

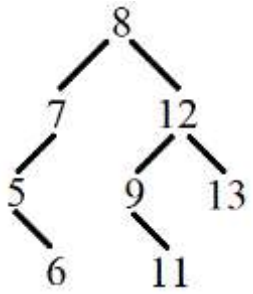

Figura 9

Consumul de memorie este identic față de listă dublu înlănțuită. Din cele trei tipuri de parcurgere a arborelui binar de căutare (preordine, inordine, postordine), parcurgerea inordine reprezintă un fragment crucial a metodei de sortare „Binary Tree Sort”.

```
>mem
Amprenta de memorie: 1 alocari (8 bytes)

>insert
Nr. de elemente de inserat: 8
INSERT id = 8
succes

INSERT id = 7
succes

INSERT id = 12
succes

INSERT id = 5
succes

INSERT id = 9
succes

INSERT id = 13
succes

INSERT id = 6
succes

INSERT id = 11
succes


>mem
Amprenta de memorie: 9 alocari (200 bytes)

>select
Ordinea parcurgerii:
1. Preordine;
2. Inordine;
3. Postordine.
>1
```

```
8,Lindy,Alberts,lalberts7@unicef.org,F,4263.470000,35.63.99.129,CN
7,Ross,Blewis,rblewis6@scribd.com,M,112.070000,158.139.134.165,CL
5,Jerad,Winpenny,jwinpenny4@360.cn,M,1164.350000,21.143.138.69,MP
6,Retha,Broome,rbroome5@ox.ac.uk,F,1264.290000,156.53.2.206,CZ
12,Ira,Beechcraft,ibeechcraftb@answers.com,F,4966.550000,135.104.120.68,BD
9,Damara,Entreis,dentreis8@wikimedia.org,F,533.510000,225.68.220.80,PH
11,Jasmin,Leachman,jleachmana@berkeley.edu,F,235.250000,173.16.160.41,CN
13,Britni,Mapam,bmapamc@gravatar.com,F,1056.860000,254.85.200.12,CN

>select
Ordinea parcurgerii:
1. Preordine;
2. Inordine;
3. Postordine.
>2
5,Jerad,Winpenny,jwinpenny4@360.cn,M,1164.350000,21.143.138.69,MP
6,Retha,Broome,rbroome5@ox.ac.uk,F,1264.290000,156.53.2.206,CZ
7,Ross,Blewis,rblewis6@scribd.com,M,112.070000,158.139.134.165,CL
8,Lindy,Alberts,lalberts7@unicef.org,F,4263.470000,35.63.99.129,CN
9,Damara,Entreis,dentreis8@wikimedia.org,F,533.510000,225.68.220.80,PH
11,Jasmin,Leachman,jleachmana@berkeley.edu,F,235.250000,173.16.160.41,CN
12,Ira,Beechcraft,ibeechcraftb@answers.com,F,4966.550000,135.104.120.68,BD
13,Britni,Mapam,bmapamc@gravatar.com,F,1056.860000,254.85.200.12,CN

>select
Ordinea parcurgerii:
1. Preordine;
2. Inordine;
3. Postordine.
>3
6,Retha,Broome,rbroome5@ox.ac.uk,F,1264.290000,156.53.2.206,CZ
5,Jerad,Winpenny,jwinpenny4@360.cn,M,1164.350000,21.143.138.69,MP
7,Ross,Blewis,rblewis6@scribd.com,M,112.070000,158.139.134.165,CL
11,Jasmin,Leachman,jleachmana@berkeley.edu,F,235.250000,173.16.160.41,CN
9,Damara,Entreis,dentreis8@wikimedia.org,F,533.510000,225.68.220.80,PH
13,Britni,Mapam,bmapamc@gravatar.com,F,1056.860000,254.85.200.12,CN
12,Ira,Beechcraft,ibeechcraftb@answers.com,F,4966.550000,135.104.120.68,BD
8,Lindy,Alberts,lalberts7@unicef.org,F,4263.470000,35.63.99.129,CN

>erase
WHERE id = 8

>select
Ordinea parcurgerii:
1. Preordine;
2. Inordine;
3. Postordine.
>1
9,Damara,Entreis,dentreis8@wikimedia.org,F,533.510000,225.68.220.80,PH
7,Ross,Blewis,rblewis6@scribd.com,M,112.070000,158.139.134.165,CL
5,Jerad,Winpenny,jwinpenny4@360.cn,M,1164.350000,21.143.138.69,MP
6,Retha,Broome,rbroome5@ox.ac.uk,F,1264.290000,156.53.2.206,CZ
12,Ira,Beechcraft,ibeechcraftb@answers.com,F,4966.550000,135.104.120.68,BD
11,Jasmin,Leachman,jleachmana@berkeley.edu,F,235.250000,173.16.160.41,CN
13,Britni,Mapam,bmapamc@gravatar.com,F,1056.860000,254.85.200.12,CN

>select
Ordinea parcurgerii:
1. Preordine;
2. Inordine;
3. Postordine.
>2
```

```
5,Jerad,Winpenny,jwinpenny4@360.cn,M,1164.350000,21.143.138.69,MP
6,Retha,Broome,rbroome5@ox.ac.uk,F,1264.290000,156.53.2.206,CZ
7,Ross,Blewis,rblewis6@scribd.com,M,112.070000,158.139.134.165,CL
9,Damara,Entreis,dentreis8@wikimedia.org,F,533.510000,225.68.220.80,PH
11,Jasmin,Leachman,jleachmana@berkeley.edu,F,235.250000,173.16.160.41,CN
12,Ira,Beechcraft,ibeechcraftb@answers.com,F,4966.550000,135.104.120.68,BD
13,Britni,Mapam,bmapamc@gravatar.com,F,1056.860000,254.85.200.12,CN

>select
Ordinea parcurgerii:
1. Preordine;
2. Inordine;
3. Postordine.
>3
6,Retha,Broome,rbroome5@ox.ac.uk,F,1264.290000,156.53.2.206,CZ
5,Jerad,Winpenny,jwinpenny4@360.cn,M,1164.350000,21.143.138.69,MP
7,Ross,Blewis,rblewis6@scribd.com,M,112.070000,158.139.134.165,CL
11,Jasmin,Leachman,jleachmana@berkeley.edu,F,235.250000,173.16.160.41,CN
13,Britni,Mapam,bmapamc@gravatar.com,F,1056.860000,254.85.200.12,CN
12,Ira,Beechcraft,ibeechcraftb@answers.com,F,4966.550000,135.104.120.68,BD
9,Damara,Entreis,dentreis8@wikimedia.org,F,533.510000,225.68.220.80,PH

>mem
Amprenta de memorie: 9 alocari (200 bytes)

>find
WHERE id = 12
12,Ira,Beechcraft,ibeechcraftb@answers.com,F,4966.550000,135.104.120.68,BD
```

Figura 10

Concluzii

Conform experimentelor realizate, pot să concluzionez că:

- Listele simplu înlănțuite sunt utile pentru cazurile când adăugăm și ștergem frecvent elementele. Viteza de acces unui element este mai încet, în comparație cu bufferi;
- Listele dublu înlănțuite sunt utile pentru memorizarea elementelor care țin cont de predecesorii săi;
- Stivele/cozile implementate prin listă sunt utile în cazul când: elementele sunt adăugate și șterse în mod frecvent, viteza de acces nu este importantă, și că utilizarea memoriei este scăzută;
- Stivele/cozile implementate prin buffer sunt utile în cazul când: avem un număr de elemente, mai mic decât mărimea buffer-ului (operația de adăugare/ștergere se execută rapid). Dezavantajele: consumul de memorie este mai mult, și are loc redimensionarea buffer-ului dacă buffer-ul este prea mic pentru a stoca un nou element (complexitatea temporală liniară în cel mai rău caz – pentru adăugare).

Anexe

Se anexează fișierul .csv și codurile sursă elaborate în limbajul de programare C, în care sunt implementate structurile dinamice de date cerute în sarcină.