Structuri de date

Laboratorul 3

Mihai Nan
mihai.nan.cti@gmail.com
Grupa 314aCC

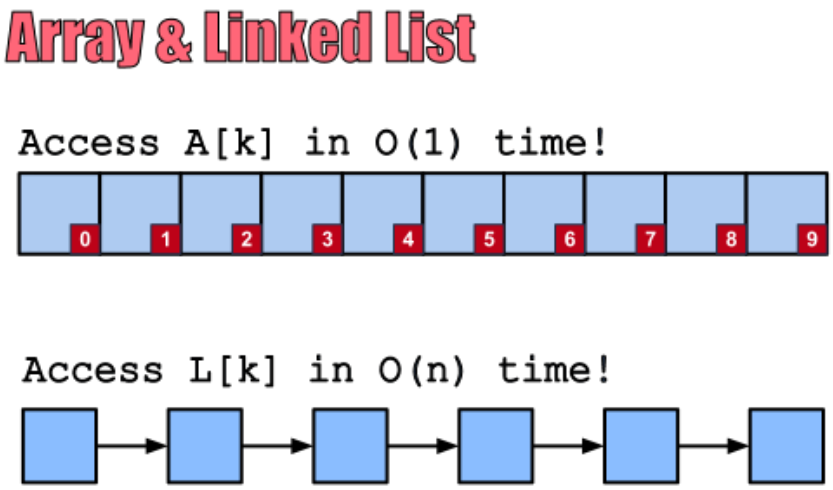
Facultatea de Automatică şi Calculatoare
Universitatea Politehnica din Bucureşti
Anul universitar 2017 - 2018

1 Liste simplu inlantuite

1.1 Introducere

Listele simplu inlantuite sunt structuri de date dinamice omogene. Spre deosebire de masive, listele nu sunt alocate ca blocuri de memorie, ci ca elemente separate de memorie. Fiecare nod al listei contine, in afara de informatia utila, adresa urmatorului element. Aceasta organizare permite numai acces secvential la elementele listei.

Pentru accesarea listei, trebuie cunoscuta adresa primului element (numit cap de lista), elementele urmatoare fiind accesate parcurgand lista.



O ***lista liniara simplu inlantuita*** este caracterizata prin faptul ca relatia de ordine definita pe multimea elementelor este unica si totala. Ordinea elementelor pentru o astfel de lista este specificata explicit, printr-un camp de informatie care este parte componenta a fiecarui element si indica elementul urmator, conform cu relatia de ordine definita pe multimea elementelor listei.

1.2 Implementarea structurii de date

Daca implementarea structurii se face prin tehnici de alocare dinamica, se obtine o ***lista inlantuita alocata dinamic***. Pentru rolul pe care il joaca informatiile de legatura in structurile inlantuite, cel mai potrivit este tipul ***pointer***. Tipul campului care realizeaza legatura va fi ***pointer la element de lista***.

Structura de date este recursiva, deoarece campul informatiei de legatura este de tip pointer la tipul nodului. Astfel, listele simplu inlantuite sunt structuri obtinute prin concatenarea de noduri de tipul descris in declararea structurii prin intermediul campului de legatura.

Observatie

Pentru simplitate si omogenitate, exercitiile din cadrul acestui laborator se vor efectua uzitand reprezentarea, definita in blocul de cod de mai jos, pentru o lista liniara simplu inlantuita alocata dinamic.

Cod sursa C

```c
typedef struct list {
  int value;
  struct list* next;
}*List;
```

1.3 Operatii cu liste

1.3.1 Initializarea listei

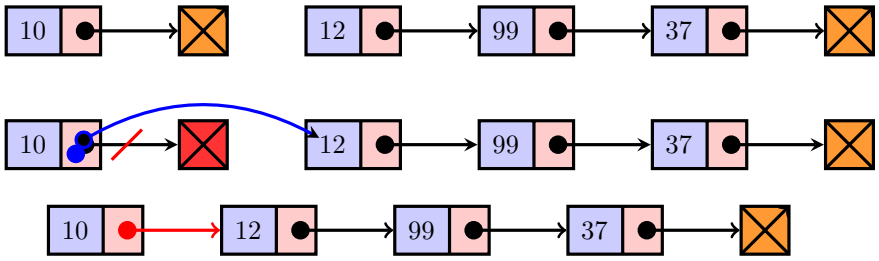
Operatia de initializare a unei liste stabileste adresa de inceput a listei, cu alte cuvinte, construieste o lista cu un singur element. Crearea unui nou element de lista necesita alocarea de memorie, prin functia ***malloc***. Verificarea rezultatului cererii de alocare (***NULL*** - daca alocarea este imposibila) se poate face printr-o instructiune ***if***. Dupa ce i s-a alocat memorie, se retine valoarea si se stabileste adresa elementului urmator ca ***NULL*** – pentru a marca sfarsitul listei.

1.3.2 Inserarea unui element

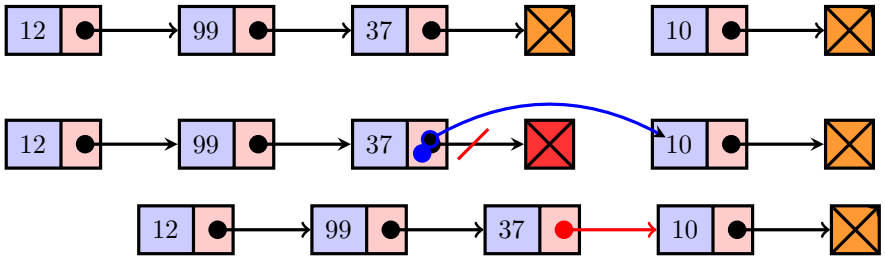
Inserarea unui element se poate face la inceputul, la sfarsitul listei sau in interiorul listei.

Adaugarea la inceputul listei

Avem de analizat doua cazuri: daca lista este vida sau daca lista contine elemente. In ambele cazuri, uzitam functia de initializare a listei pentru a defini un nou nod pe care sa il alipim listei.



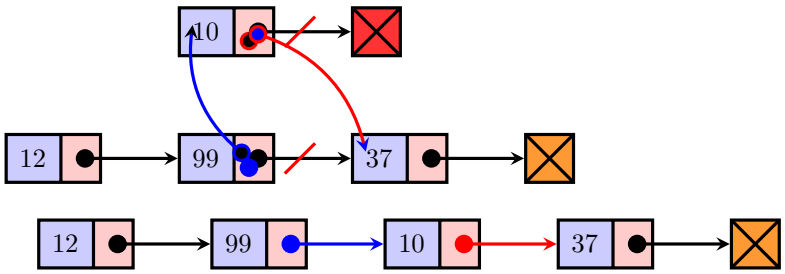
Adaugarea la sfarsitul listei



De asemenea, si aici avem de analizat cele doua cazuri: lista vida sau lista care contine elemente. Daca lista nu este vida, adaugam nodul la sfarsitul listei. Alocam memorie pentru noul nod si scriem informatia pe care dorim sa o adaugam in nod tot cu ajutorul functiei pentru initializare. Daca lista nu este vida, trebuie sa legam ultimul nod al listei de noul nod, devenind ultimul nod al listei, iar, in caz contrar, doar sa returnam noul nod initializat. Pentru a putea parcurge lista, trebuie sa folosim o varibaila auxiliara ***temp*** tot de tip ***List***, precum si o structura repetitiva – ***while***.

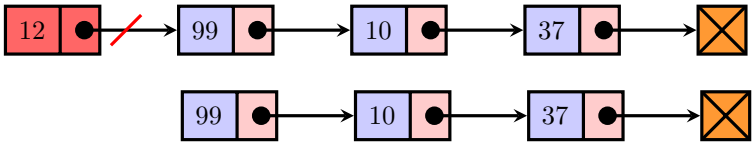
Adaugarea in interiorul listei

Pentru inserarea unui nod dupa un nod existent in lista trebuie sa alocam memorie pentru noul nod si sa legam acest nod de predecesorul sau si de succesorul sau. Daca nodul dupa care am dori sa adaugam informatia este ultimul nod al listei, aplicam fix rationamentul de la inserarea la sfarsitul listei.



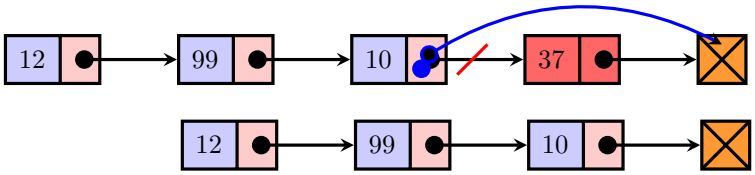
1.3.3 Stergerea unui element

Ca si adaugarea, stergerea unui element dintr-o lista este de mai multe feluri. Prin operatia de stergere se intelege scoaterea unui element din inlantuire. Elementul care a fost izolat de lista trebuie sa fie procesat in continuare, cel putin pentru a fi eliberata zona de memorie pe care o ocupa, de aceea adresa lui trebuie salvata.
Stergerea de la inceputul listei



Sunt cateva cazuri care trebuie sa le luam in cosiderare: lista este vida sau lista are un singur element. Oricum ar fi, algoritmul general este urmatorul: salvez primul nod intr-un nod auxiliar ***temp***, succesorul primului nod (*al doilea nod din lista initiala*) devine primul nod si eliberez memoria ocupata de obiectul salvat in ***temp***.

Stergerea de la finalul listei



Pentru stergerea utlimului element din lista trebuie sa cautam predecesorul sau – care va deveni ultimul nod din lista – si sa eliberam memoria alocata pentru ultimul nod din lista.

2 Probleme de laborator

Observatie

In arhiva laboratorului, gasiti un schelet de cod de la care puteti porni implementarea functiilor propuse, avand posibilitatea de a le testa functionalitatea.
Punctajul maxim pentru acest laborator este 10 si se va acorda doar daca rezolvarea este insotita si de explicatii.

2.1 Probleme standard

Problema 1 - 0 puncte

Pornind de la definitia structurii de date ***List*** care reprezinta un element al unei liste simplu inlantuite ce va retine valori de tip ***int***, implementati operatia de initializare a unei liste. Aceasta operatie stabileste adresa de inceput a listei, aloca memoria necesara pentru retinerea unui element, se adauga valoarea elementului ***value*** in lista si se stabileste adresa elementului urmator ca fiind ***NULL*** - pentru a marca sfarsitul listei.

```
List initList(int value);
```

Problema 2 - 0 puncte

Implementati o functie care afiseaza o lista liniara simplu inlantuita, primita ca argument.

Observatie

In implementarea functiilor pentru rezolvarea urmatoarelor 3 probleme, analizati separat cazul listei vide!

Problema 3 - 0.5 puncte

Implementati operatia de adaugare a unui element la inceputul listei liniare simplu inlantuite.

```
List addFirst(List l, int value);
```

Problema 4 - 1 punct

Implementati operatia de adaugare a unui element la sfarsitul listei liniare simplu inlantuite.

```
List addLast(List l, int value);
```

Problema 5 - 1.5 puncte

Definiti o functie care realizeaza operatia de adaugare a unui nod in lista, pozitionandu-l dupa primul nod care contine valoarea **x**. Se garanteaza ca exista in lista cel putin un nod cu valoarea **x**.

```
List addItem(List l, int x, int value);
```

Problema 6 - 2 puncte

Sa se scrie o functie care poate fi uzitata pentru a sterge dintr-o lista liniara simplu inlantuita nodul care contine valoarea ***value***.

```
List deleteItem(List l, int value);
```

⚠ IMPORTANT !

⚠ Analizati toate cazurile posibile, raportandu-va la pozitia pe care o poate avea un nod intr-o lista si luand in considerare cazul listei vide!

Problema 7 - 1 punct

Definiti o functie care are ca unic argument o lista liniara simplu inlantuita si implementeaza operatia de stergere a listei. Cu alte cuvinte, functia va avea ca rezultat lista vida si va sterge fiecare nod din lista, dealocand memoria necesara reprezentarii sale.

Problema 8 - 2 puncte

Sa se implementeze o functie care primeste doua liste simplu inlantuite ordonate crescator si realizeaza reuniunea lor. In lista rezultata nu vor exista elemente duplicate.

```
List reunion(List list1, List list2);
```

Problema 9 - 2 puncte

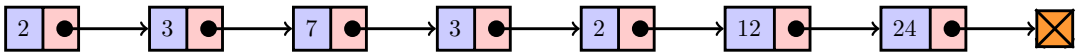
Scrieti un program care sa determine elementul din mijlocul unei liste simplu inlantuite, fara a folosi un contor si fara a cunoaste dimensiune listei.

2.2 Problema bonus

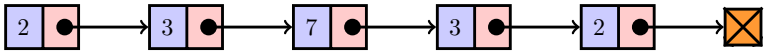
Problema 10 - 2 puncte

Danduse o lista liniara simplu inlantuita ce contine numere intregi, determinati sublista palindrom maximala, ca si numar de elemente.

Exemplu



Rezultatul



3 Interviu

Observatie

Aceasta sectiune este una optionala si incearca sa va familiarizeze cu o serie de intrebari ce pot fi adresate in cadrul unui interviu tehnic. De asemenea, aceasta sectiune poate fi utila si in pregatirea pentru examenul final de la aceasta disciplina.

Intrebari interviu

1. Cum copiezi o lista simplu inlantuita daca, pe langa pointerul la urmatorul nod (***next***), fiecare nod mai are un pointer si la nodul care urmeaza dupa nodul urmator (***next->next***)?
2. Enumerati cateva avantaje ale folosirii listelor simplu inlantuite alocate dinamic in detrimentul vectorilor.
3. Cum se poate reprezenta o matrice folosind numai liste liniare simplu inlantuite?
4. Cum se poate eficientiza operatia de adaugare a unui element la sfarsitul unei liste liniare simplu inlantuite? Implementati o functie care sa realizeze aceasta operatie, tinand cont de optimizarea propusa.
5. Se poate realiza operatia de inversare a unei liste simplu inlantuite fara a folosi memorie auxiliara? Daca acest lucru este posibil, oferiti o implementare a acestei operatii.
6. Se ofera un text care contine mai multe paragrafe care au numar variat de linii. Cum se poate retine acest text in memorie folosind numai liste simplu inlantuite, astfel incat sa fie facila accesarea caracterului aflat pe pozitia **nrC** a linii cu numarul **nrL** din paragraful **nrP**?

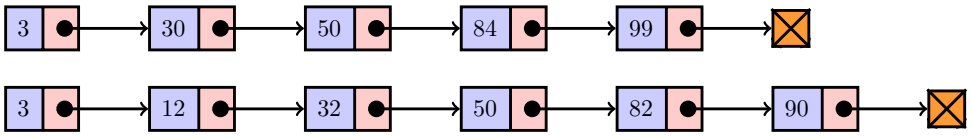
3.1 Probleme interviu

Problema 1: Implementati un program care detecteaza daca o lista simplu inlantuita contine o bucla.

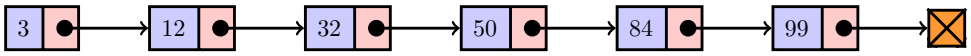
Problema 2: Se dau doua liste liniare simplu inlantuite ordonate crescator. Sa se implementeze un program care determina secventa de suma maxima care se poate forma din cele doua liste respectand urmatoarele liste: rezultatul poate contine elemente din ambele liste, cand construim rezultatul, putem sa trecem de la o lista la alta doar prin punctele de intersectie (noduri care contin valori egale). In implementare, vi se permite uzitarea unui spatiu auxiliar cuprins in O(1).

Exemplu

input



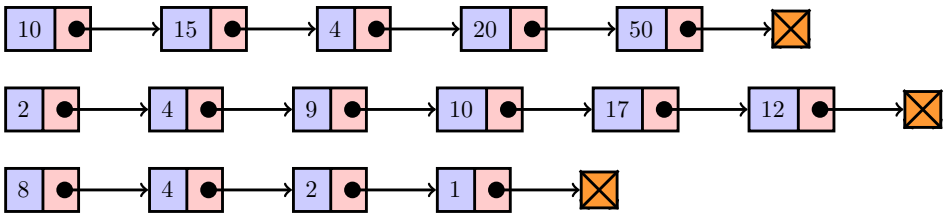
output



Problema 3: Se ofera trei liste simplu inlantuite. Sa se scrie un program care citeste cele 3 liste si un numar intreg de la tastatura si determina un triplet, un element din fiecare lista, a carui suma este egala cu numarul citit.

Exemplu

input



25

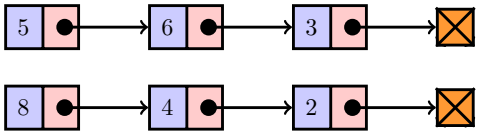
output

15 2 8

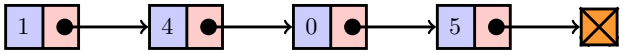
Problema 4: Danduse doua numere naturale reprezentate prin doua liste liniare simplu inlantuite, fiecare nod reprezentand o cifra, implementati o functie care sa realizeze operatia de adunare a celor doua numere naturale, rezultatul fiind reprezentat tot ca o lista liniara simplu inlantuita.

Exemplu

input



output



Feedback

Pentru imbunatatirea constanta a acestui laborator, va rog sa completati formularul de feedback disponibil aici.

De asemenea, va rog sa semnalati orice greseala / neclaritate depistata in laborator pentru a o corecta.

Va multumesc anticipat!