Laborator de Structuri de Date si Algoritmi – Lucrarea nr. 6

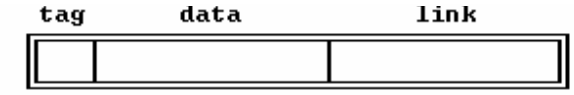
Liste generalizate

1. Reprezentarea in C++

2. Parcurgerea unei liste generalizate

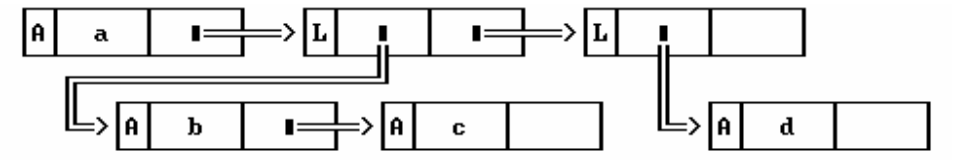
1. Reprezentarea in C++

Un element al unei liste generalizate poate fi un *atom* sau o *lista*. Reprezentarea trebuie sa asigure un mijloc de a discrimina intre cele doua situatii. Solutia directa este de a memora in fiecare nod tipul lui (Atom sau Lista). Astfel, un element va avea urmatoarea structura:

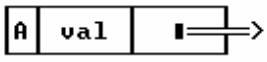


unde: tag - eticheta discriminatoare;
data - informatia utila: un Atom sau o Lista;
link - informatia de inlantuire.

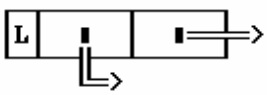
De exemplu, lista **(a, (b, c), (d))** va avea reprezentarea:



Cimpul *data* poate contine un Atom sau o lista. Deci pentru structura de date care memoreaza un element de lista exista doua variante:



Un element de lista generalizata care contine un Atom avind valoarea val. Cimpul "data" contine valoarea atomului.



Un element de lista generalizata care contine o Lista. Cimpul "data" contine pointer la primul element dintr-o lista.

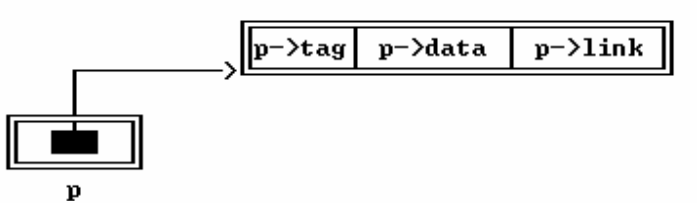
Laborator de Structuri de Date si Algoritmi – Lucrarea nr. 6

Acest lucru se exprima in C++ cu ajutorul unei uniuni (union):

```
typedef char Atom;
enum { ATOM, LISTA};
struct GElement;                    // declaratie simpla
typedef GElement* GLista;           // o lista generalizata este un
                                    // pointer la primul element

struct GElement{
     char tag;                      // eticheta
     union {                        // data poate fi
              Atom A;               // Atom
              GLista L;             // sau GLista
            } data;
     Gelement *link;                // legatura
     };
```

Daca ***p*** este un pointer care contine adresa unui GElement, atunci in memorie vom avea:



p->data.A - cimpul data considerat Atom
p->data.L - cimpul data considerat lista

2. Parcurgerea unei liste generalizate

Definitia unei liste generalizate este o definitie recursiva (O „lista” este o colectie ordonata de atomi si „liste”. In consecinta operatiile de prelucrare a listelor generalizate vor fi realizate prin proceduri recursive.

Iata procedura de ***afisare*** a unei liste generalizate:

```
void afisare(GLista L)
{
     GElement* p;
     p=L;
     while(p!=NULL) {
            if (p->tag==ATOM)
                 printf("%c",p->data.A);
            else afisare(p->data.L);
          p = p->link;
          }
}
```

Prelucrare element de lista

Laborator de Structuri de Date si Algoritmi – Lucrarea nr. 6

Se observa scheletul parcurgerii unei liste inlantuite simple, in care prelucrarea elementelor se face diferentiat in functie de tipul acestora: Atom sau Lista. Prelucrarea sublistelor se face prin apeluri recursive la aceeasi procedura.

Sa scriem o functie care calculeaza ***numarul de atomi*** aflati in total intr-o lista si in toate listele continute de aceasta.

Pornim de la urmatoarea definitie recursiva:

$$NrAtomi(L) = Na(L) + \Sigma\ NrAtomi(Li)$$

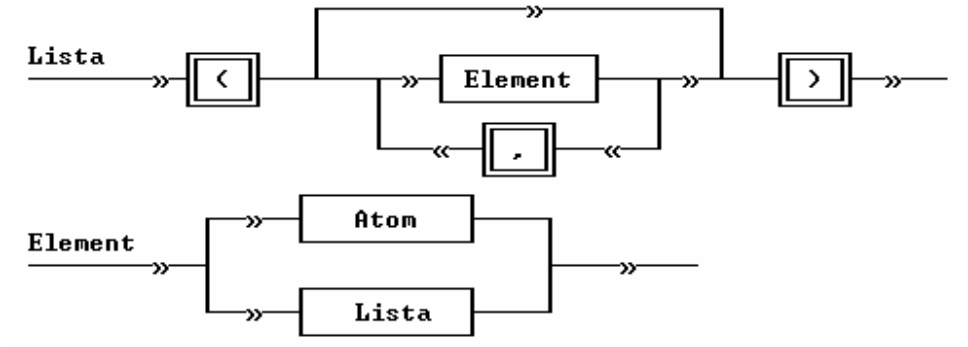
unde: - NrAtomi - numarul total de atomi din lista L;
- Na(L) - numarul elementelor listei L care sint atomi;
- Li - elementele listei L care sint subliste.

```
int NrAtomi(GLista L)
{
          GElement* p;
          int nr = 0;
          p = L;
          while p<>NULL do
                    {
                    if p->tag==ATOM then
                        nr++;
                    else nr+=NrAtomi(p->data.L);
                    p = p->link;
                    }
          return nr;
}
```

Prelucrare element de lista

Problema

Sa se scrie o functie care citeste o lista generalizata, cu atomi de tip caracter, continind o litera, specificata conform urmatoarei diagrame de sintaxa:



Exemple: (A,(B,C),(D)) - O lista formata dintr-un atom A, o lista cu doua elemente (B,C) si o lista cu un element (D) ;

Laborator de Structuri de Date si Algoritmi – Lucrarea nr. 6

((A,B,C),(),B) - O lista formata dintr-o lista cu trei elemente (A,B,C), o lista vida () si un atom B.

In ***Anexa*** laboratorului este prezentata rezolvarea acestei probleme.

TEMA

Sa se scrie un program care citeste o lista generalizata specificata conform indicatiilor de mai sus, folosind functia ***creareGLista*** din modulul GLISTA.CPP

- Sa se afiseze toti atomii din lista.
- Sa se determine atomul cu valoarea cea mai mica dintre atomii cuprinsi in lista citita si in sublistele ei.
- Sa se afiseze lista citita in formatul folosit la intrare (cu paranteze).
- Sa se afiseze numarul de atomi aflati in total in lista si in toate listele continute de aceasta..
- Sa se testeze daca doua liste sint egale. Doua sint egale daca elementele lor luate in ordine sint egale doua cite doua.
- Definim adincimea listei ca fiind numarul maxim de nivele de imbricarea (lista in lista).

De exemplu:

(A, B) - are adincimea 1;
(A, (B,C)) - are adincimea 2;
((B, (C,D)),E) - are adincimea 3;

Sa se afiseze adincimea listei citite.

TEMA (dificila)

Definim doua liste echivalente, doua liste care au aceeasi atomi si listele continute sint echivalente doua cite doua (nu conteaza ordinea). De exemplu:

(A, (B,C), (D)) ≡ ((D), A, (C,B))

Sa se scrie un program care sa determine daca doua liste sint echivalente.

Laborator de Structuri de Date si Algoritmi – Lucrarea nr. 6

ANEXA

Diagramele de sintaxa impuse sint indirect recursive (diagrama Lista este definita cu ajutorul diagramei Element, care la rindul ei este definita cu ajutorul diagramei Lista). Corespunzator fiecarei diagrame vom scrie o subrutina:

GElement* citesteGElement();
// Citeste de la intrare o formulare care corespunde diagramei "Element", creeaza un
// element de lista si il completeaza in functie de situatie, cu un Atom obtinut printr-un
// apel al functiei citesteAtom sau cu un pointer la o lista obtinut printr-un apel al
// functiei *citesteLista*.
// *citesteGElement* intoarce pointer la elementul creat.

GLista citesteGLista();
// Citeste o lista in care elementele sint citite si create de *citesteGElement*, si intoarce
// pointer la primul element. Subrutina verifica corectitudinea utilizarii parantezelor.

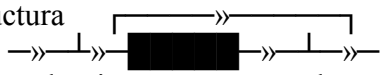
Observatii:

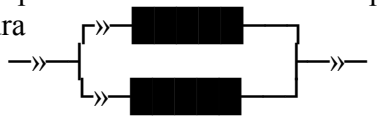
■ La intrarea in fiecare subrutina primul caracter din sirul de caractere care va fi prelucrat de subrutina respectiva este deja citit in variabila *crtChar*. Puteti da o motivatie pentru acest lucru?

■ Pentru uniformitate, la iesirea din fiecare subrutina *crtChar* contine valorea caracterului urmator.

■ In fluxul de control al subrutinelor:

- Un bloc de tip "notiune" (cu linie simpla) corespunde unui apel de subrutina.
- Un bloc de tip "simbol" (cu linie dubla) corespunde unui test asupra caracterului citit de la intrare.

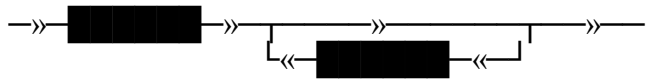
- O structura 
in diagrama de sintaxa, corespunde unei structuri de control de tip IF-THEN. Conditia se pune asupra caracterului cu care incepe formularea din cadrul blocului.

- O structura 
in diagrama de sintaxa, corespunde unei structuri de control de tip IF-THEN-ELSE.

- O structura 
in diagrama de sintaxa, corespunde unei structuri de control de tip WHILE-DO.

- O structura 

in diagrama de sintaxa, corespunde unei structuri de control de tip REPEAT-UNTIL. In cazul nostru primul element care se insereaza in lista trebuie tratat separat, de aceea am utilizat o scriere echivalenta cu

WHILE-DO: 

```
/*************************************************************
                    Fisierul GLISTA.H
*************************************************************/
   typedef char Atom;
   enum { ATOM, LISTA};
   struct GElement;          // declaratie simpla
   typedef GElement* GLista;   // o lista generalizata este un
                               // pointer la primul element

   struct GElement{
         char tag;          // eticheta
         union {            // data poate fi
               Atom A;      //      Atom
               GLista L;    //  sau GLista
               } data;
         Element* link;     // legatura
         };

GLista creareGLista();
   // citeste si returneaza o lista generalizata data de la
   // intrarea standard, respectind sintaxa specificata.

/*************************************************************
                    Fisierul GLISTA.CPP
*************************************************************/
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#include <ctype.h>

#include "GLISTA.H"

char crtChar;

void eroare()
{
   printf("Sirul de intrare este eronat!\n"
          "Apasati tasta o tasta ...");
   getch();
```

```
    exit(1);
}

void readchar()
{
    do crtChar=getchar();
    while(crtChar==' ');
}

char citesteAtom()
{
    char c;
    if(!isalpha(crtChar)) eroare();
    c = crtChar;
    readchar();
    return c;
}

GLista citesteGLista();

GElement* citesteGElement()
{
    GElement* p = new GElement;
    if(crtChar=='(') {
                        p->data.L = citesteGLista();
                        p->tag = LISTA;
                        }
    else {
            p->data.A = citesteAtom();
            p->tag = ATOM;
            }
   p->link = NULL;
   return p;
}

GLista citesteGLista()
{
    GElement* cap, *coada, *p;
    if( crtChar!='(' ) eroare();
    readchar();
    if( crtChar==')' )  cap = NULL;
    else {
            cap = citesteGElement();
            coada = cap;
            while( crtChar==',' ) {
                    readchar();
```

```
                    p = citesteGElement();
                    coada->link = p;
                    coada = p;
                    }
            if( crtChar!=')' ) eroare();
            }
    readchar();
    return cap;
}

GLista creareGLista()
{
    do readchar();
    while (crtChar=='\n');
    return (citesteGLista());
}
```