PROGRAMARE ȘI STRUCTURI DE DATE

CURS 8

Lect. dr. Oneț-Marian Zsuzsanna

Facultatea de Matematică și Informatică UBB
în colaborare cu NTT Data

Cuprins

Vectorul

- *Vectorul* este o structură folosită foarte des când avem de stocat mai multe elemente de același tip.
- Să vedem care sunt avantajele și dezavantajele vectorului.

Vectorul - Avantaje

- Care credeți că este principalul avantaj al unui vector?

Vectorul - Avantaje

- Care credeți că este principalul avantaj al unui vector?
- Când se definește un vector, trebuie să spunem ce tip vor avea elementele vectorului și câte elemente vrem să aibă vectorul.

Vectorul - Avantaje

- Care credeți că este principalul avantaj al unui vector?
- Când se definește un vector, trebuie să spunem ce tip vor avea elementele vectorului și câte elemente vrem să aibă vectorul.
- Pe baza acestor informații se rezervă spațiu de memorare pentru vector. De exemplu, dacă limbajul de programare folosește 4 bytes pentru a stoca un număr întreg, și noi definim un vector de numere întregi de 10 elemente, atunci $4 * 10 = 40$ de bytes consecutivi vor fi rezervați pentru vectorul nostru.

Vectorul - Avantaje

- Care credeți că este principalul avantaj al unui vector?
- Când se definește un vector, trebuie să spunem ce tip vor avea elementele vectorului și câte elemente vrem să aibă vectorul.
- Pe baza acestor informații se rezervă spațiu de memorare pentru vector. De exemplu, dacă limbajul de programare folosește 4 bytes pentru a stoca un număr întreg, și noi definim un vector de numere întregi de 10 elemente, atunci $4 * 10 = 40$ de bytes consecutivi vor fi rezervați pentru vectorul nostru.
- *Valoarea* vectorului (adică ceea ce este reținut intern) este de fapt adresa de memorie unde începe vectorul.

Vectorul - Avantaje

- Având aceste date (zona de memorie unde începe vectorul și dimensiunea unui element), *adresa de memorie* pentru fiecare element din vector poate fi calculată foarte ușor.

Vectorul - Avantaje

- Având aceste date (zona de memorie unde începe vectorul și dimensiunea unui element), *adresa de memorie* pentru fiecare element din vector poate fi calculată foarte ușor.
- Presupunând că avem un vector de 10 elemente, care începe la adresa de memorie 168 (pentru simplitate folosim numere întregi în baza 10 pentru a reprezenta adrese de memorie), și conține numere întregi, iar fiecare număr întreg ocupă 4 bytes:
  - Primul element începe la adresa 168 (elementul pe poziția 0)
  - Al 2-lea element începe la adresa 172 (elementul pe poziția 1)
  - Al 3-lea element începe la adresa 176 (elementul pe poziția 2)
  - ...
  - Al i-lea element începe la adresa 168 + 4 * (i-1) (elementul pe poziția i-1)

Vectorul - Avantaje

- Având aceste date (zona de memorie unde începe vectorul și dimensiunea unui element), *adresa de memorie* pentru fiecare element din vector poate fi calculată foarte ușor.
- Presupunând că avem un vector de 10 elemente, care începe la adresa de memorie 168 (pentru simplitate folosim numere întregi în baza 10 pentru a reprezenta adrese de memorie), și conține numere întregi, iar fiecare număr întreg ocupă 4 bytes:
  - Primul element începe la adresa 168 (elementul pe poziția 0)
  - Al 2-lea element începe la adresa 172 (elementul pe poziția 1)
  - Al 3-lea element începe la adresa 176 (elementul pe poziția 2)
  - ...
  - Al i-lea element începe la adresa 168 + 4 * (i-1) (elementul pe poziția i-1)
- Practic, când noi scriem vector[i], acest lucru se transformă în ceva de genul: *accesează elementul care se află pe poziția adresă_vector + i * mărimea_unui_element*

Vectorul - Avantaje

- A calcula adresa unui element (folosind formula de pe pagina anterioară) necesită timp constant (o singură operație), a accesa ce se află la o anumită adresă (acest lucru se întâmplă implicit când folosim o variabilă) tot timp constant necesită.

- Avantajul principal al unui vector este faptul că oricare element poate fi accesat în timp constant ($\Theta(1)$).

Vectorul - Dezavantaje

- Care credeți că este dezavantajul principal al unui vector?

Vectorul - Dezavantaje

- Care credeți că este dezavantajul principal al unui vector?
- Când vrem să definim un vector trebuie să specificăm neapărat câte elemente va avea vectorul. Odată ce am definit un vector de n elemente, ulterior nu ne putem răzgândi, că de fapt vrem $2 * n$. Dacă nu știm exact câte elemente va avea vectorul, trebuie să găsim o limită superioară, pentru că odată definit, vectorul nu poate fi făcut mai mare.

Vectorul - Dezavantaje

- Care credeți că este dezavantajul principal al unui vector?
- Când vrem să definim un vector trebuie să specificăm neapărat câte elemente va avea vectorul. Odată ce am definit un vector de n elemente, ulterior nu ne putem răzgândi, că de fapt vrem $2 * n$. Dacă nu știm exact câte elemente va avea vectorul, trebuie să găsim o limită superioară, pentru că odată definit, vectorul nu poate fi făcut mai mare.
- Pe de altă parte, nici să ocup mult spațiu degeaba, pe motiv că *să fim siguri că e suficient de mare* nu e în regulă.

Vectorul - Dezavantaje

- Când avem nevoie de un vector, trebuie să stabilim numărul de elemente (dimensiunea vectorului) căutând un echilibru între a avea suficient spațiu să memorăm toate elementele de care avem nevoie, dar să nici nu ocupăm prea mult spațiu degeaba. Acest aspect este dezavantajul principal pentru un vector.

Vectorul dinamic

- Soluția pentru a evita dezavantajul principal (măcar parțial) este să folosim un *vector dinamic*.
- *Vectorul dinamic* este un vector a cărui dimensiune poate fi modificată.

Vectorul dinamic

- Soluția pentru a evita dezavantajul principal (măcar parțial) este să folosim un *vector dinamic*.
- *Vectorul dinamic* este un vector a cărui dimensiune poate fi modificată.
- Ideea de bază este simplă:
  - Definim un vector de n elemente.
  - Dacă la un moment dat vectorul este plin (toate pozițiile sunt ocupate), dar mai trebuie să adăugăm elemente, vom defini un alt vector, mai mare, și vom copia elementele din acest vector în cel nou. După ce am copiat elementele, vectorul nou devine vectorul nostru.

Vecotrl dinamic II

- Pentru a reprezenta un vector dinamic avem nevoie de 3 informații:
  - numărul maxim de elemente care încap în vector (capacitatea)
  - numărul de elemente care sunt stocate în vector
  - vectorul efectiv

Vecotrl dinamic II

- Pentru a reprezenta un vector dinamic avem nevoie de 3 informații:
  - numărul maxim de elemente care încap în vector (capacitatea)
  - numărul de elemente care sunt stocate în vector
  - vectorul efectiv
- Pentru că avem nevoie de mai multe informații, care tot timpul trebuie să fie folosite împreună, vom defini un tip nou de date, pentru a reprezenta un vector dinamic:

VectorDinamic:
cap: întreg
len: întreg
elemente: TElem[]

Vector Dinamic III

- *Vectorul Dinamic* este o structură de date, îl vom folosi ca să implementăm diferite containere.
- Dar, putem vorbi de operații și de interfață și pentru *Vectorul Dinamic*.

Vector dinamic - Operații I

- Ce operații ar trebui/ar putea să aibă un *vector dinamic*?

Vector dinamic - Operații II

- creeaza (cp)
  - **descriere:** creează un vector dinamic nou, care are capacitatea *cp* (constructorul clasei Vector Dinamic)
  - **pre:** *cp* - număr natural
  - **post:** Un vector dinamic a fost creat cu lungime 0 și capacitate *cp*
  - **aruncă**: excepție, dacă *cp* este negativ

Vector dinamic - Operații III

- vect.dim ()
  - **descriere:** returnează numărul de elemente din *vect*
  - **pre:** vect - VectorDinamic
  - **post:** **returnează** vect.len (numărul de elemente din vect)

Vector dinamic - Operații IV

- vect.element (i)
  - **descriere:** returnează elementul de pe poziția *i*
  - **pre:** vect - VectorDinamic, i - număr natural, $0 <= i < vect.len$
  - **post: returnează** vect.elemente[i]
  - **aruncă**: excepție, dacă *i* nu este poziție validă

Vector dinamic - Operații V

- vect.modifica (i, e)
  - **descriere:** schimbă elementul de pe poziția *i* cu valoarea *e*
  - **pre:** vect - VectorDinamic, i - număr natural, $0 <= i < vect.len$, e - TElem
  - **post:** vect.elemente[i] = e
  - **aruncă:** excepție, dacă *i* nu este poziție validă

Vector dinamic - Operații VI

- vect.adaugaSfarsit (e)
  - **descriere:** adaugă elementul *e* la finalul vectorului dinamic. Dacă este necesar, crește capacitatea vectorului.
  - **pre:** vect - VectorDinamic, e - TElem
  - **post:** vect.elemente[vect.len] = e (elementul a fost adăugat la finalul vectorului), vect.len = vect.len + 1

Vector dinamic - Operații VII

- vect.adaugaPozitie (i, e)
  - **descriere:** adaugă elementul *e* pe poziția *i*. Dacă este necesar, crește capacitatea vectorului.
  - **pre:** vect - VectorDinamic, i - număr natural, $0 <= i <= vect.len$, e - TElem
  - **post:** $vect.elemente = vect.elemente[0...i-1] + e + vect.elemente[i...vect.len-1]$, vect.len = vect.len + 1
  - **aruncă:** excepție, dacă *i* nu este poziție validă

Vector dinamic - Operații VIII

- vect.stergeSfarsit ()
  - **descriere:** șterge și returnează ultimul element din vector dinamic
  - **pre:** vect - VectorDinamic
  - **post: returnează** vect.elemente[vect.len-1]; vect.len = vect.len -1
  - **aruncă:** excepție, dacă *vect* este vid

Vector dinamic - Operații IX

- vect.stergePozitie (i)
  - **descriere:** șterge și returnează elementul de pe poziția i
  - **pre:** vect - VectorDinamic, i - număr natural, $0 <= i < vect.len$
  - **post: returnează** vect.elemente[i], vect.len = vect.len -1, $vect.elemente = vect.elemente[0...i-1] + vect.elemente[i+1...vect.len-1]$
  - **aruncă:** excepție, dacă *i* nu este poziție validă

Vector dinamic - Operații X

- vect.cauta(elem)
  - **descriere:** verifică dacă un element apare sau nu în vector
  - **pre:** vect - VectorDinamic, elem - TElem
  - **post: returnează** adevărat, dacă elem apare în vect, altfel fals

Vector dinamic - Operații XI

- vect.iterator(elem)
  - **descriere:** crează și returnează un iterator pentru vectorul dinamic
  - **pre:** vect - VectorDinamic
  - **post:** **returnează** un iterator pentru vectorul dinamic

Vector dinamic - Operații XII

- vect.stergeElement(elem)
  - **descriere:** șterge un element dat din vectorul dinamic
  - **pre:** vect - VectorDinamic, elem - TElem
  - **post:** prima apariție a elementului *elem* este ștearsă. Elementele care erau după prima apariție a lui *elem*, sunt mutate cu o poziție în față.

Vector Dinamic

- Operațiile pentr un vector dinamic seamană mult cu operațiile pentru TAD Listă. Motivul este că vectorul dinamic este o structură de date care poate fi folosită să implementăm o listă.
- Dar vectorul dinamic poate fi folosită și pentru a implementa alte containere (orice container discutat poate fi implementat pe un Vector Dinamic)
- În continuare vom vedea implementarea pentru 2 operații pentru Vector Dinamic: creează, adaugăSfârșit și vom vedea și iteratorul.

Vector Dinamic - operația creează

- Ce ar trebui să facă operația *creează*?

Vector Dinamic - operația creează

- Ce ar trebui să facă operația *creează*?

subalgorithm creeaza (cp: întreg) **este:**
//this se referă la tipul VectorDinamic, are cele 3 câmpuri prezentate mai sus
  dacă cp $\leq$ 0 **atunci**
    @aruncă o excepție
  sf_dacă
  this.cp = cp
  this.len = 0
  this.elemente = @un vector cu cp elemente ce tip TElem
sf_subalgoritm

- Cât este complexitatea subalgoritmului *creează*?

Vector Dinamic - operația creează

- Ce ar trebui să facă operația *creează*?

subalgorithm creeaza (cp: întreg) **este:**
//this se referă la tipul VectorDinamic, are cele 3 câmpuri prezentate mai sus
  dacă cp $\leq$ 0 **atunci**
    @aruncă o excepție
  sf_dacă
  this.cp = cp
  this.len = 0
  this.elemente = @un vector cu cp elemente ce tip TElem
sf_subalgoritm

- Cât este complexitatea subalgoritmului *creează*? $\Theta(1)$

Vector Dinamic - operația adaugăSfârșit

- Ce ar trebui să facă operația *adaugăSfârșit*?

Vector Dinamic - operația adaugăSfârșit

- Ce ar trebui să facă operația *adaugăSfârșit*?

```
subalgorithm adaugăSfârșit (e: TElem) este:
    dacă this.len == this.cap atunci
        //nu mai avem loc liber.
        vectNou = @un vector cu this.cp*2 elemente
        i: întreg
        pentru i = 0, this.len, 1 execută //copiem elementele existente
            vectNou[i] = this.elemente[i]
        sf_pentru
        this.elemente = vectNou //înlocuim vectorul
        this.cp = this.cp * 2
    sf_dacă
    this.elemente[this.len] = e
    this.len = this.len + 1
sf_subalgorithm
```

Vector Dinamic - operația adaugăSfârșit

- Cât este complexitatea subalgoritmului *adaugăSfârșit*?

Vector Dinamic - operația adaugăSfârșit

- Cât este complexitatea subalgoritmului *adaugăSfârșit*?
- În caz favorabil (când vectorul nu e plin), complexitatea este $\Theta(1)$
- În caz defavorabil (vectorul e plin și trebuie să creăm unul mai mare), complexitatea este $\Theta(n)$
- În caz mediu, complexitatea este $\Theta(1)$ amortizat
  - Cazul defavorabil se întâmplă rar (nu putem avea caz defavorabil de 2 ori consecutiv), de aceea, complexitatea mare pentru cazul defavorabil *se împarte* pentru cazuri favorabile, și în medie avem $\Theta(1)$.
- Complexitatea totală este $O(n)$
- **Obs:** Nu e obligatoriu să înmulțim capacitatea cu 2, putem înmulți cu 1.5, sau cu 1.3, sau 3, etc. Ceea ce este important e să înmulțim capacitatea cu o valoare, nu să adumăm o valoare (de ex: this.cp = this.cp + 5). Dacă facem adunare, nu mai avem caz defavorabil rar.

Vector Dinamic - iterator

- Până acum am discutat despre iterator la containere la modul abstract.
- Am spus că un iterator are un *element curent* din container, și trebuie să poată să *treacă* la elementul următor.
- Acum că avem o structură de date (vectorul dinamic) putem discuta despre implementarea iteratorului.
- Dacă reținem elementele într-un vector dinamic, care este cel mai simplu mod de a reține un *element curent*?

Vector Dinamic - iterator

- Până acum am discutat despre iterator la containere la modul abstract.
- Am spus că un iterator are un *element curent* din container, și trebuie să poată să *treacă* la elementul următor.
- Acum că avem o structură de date (vectorul dinamic) putem discuta despre implementarea iteratorului.
- Dacă reținem elementele într-un vector dinamic, care este cel mai simplu mod de a reține un *element curent*?
- Varianta cea mai simplă de a reține un element curent într-un vector dinamic este să reținem poziția elementului, deci elementul curent din iteratorul pentru vector dinamic, va fi o poziție.

Vector Dinamic - iterator

- Ce operații avea iteratorul?

Vector Dinamic - iterator

- Ce operații avea iteratorul?
  - creează - creează un iterator
  - element - returnează elementul curent
  - următor - trece la următorul element curent
  - valid - verifică dacă elementul curent din iterator este valid

Vector Dinamic - iterator

- Ce operații avea iteratorul?
  - creează - creează un iterator
  - element - returnează elementul curent
  - următor - trece la următorul element curent
  - valid - verifică dacă elementul curent din iterator este valid
- Ce câmpuri ar trebui să aibă iteratorul pe Vector Dinamic?

IteratorVD:
vect: VectorDinamic
curent: întreg

- E important ca să reținem în iterator și vectorul dinamic, pentru că acolo sunt elementele. Tot timpul iteratorul reține și structura/containerul peste care iterează.

Vector Dinamic - iterator - creează

- Ce ar trebui să facă operația *creează*?

Vector Dinamic - iterator - creează

- Ce ar trebui să facă operația *creează*?

subalgoritm creează (vect: VectorDinamic) **este:**
    this.vect = vect
    this.curent = 0
sf_subalgoritm

- Cât este complexitatea subalgoritmului?

Vector Dinamic - iterator - creează

- Ce ar trebui să facă operația *creează*?

subalgoritm creează (vect: VectorDinamic) **este:**
  this.vect = vect
  this.curent = 0
sf_subalgoritm

- Cât este complexitatea subalgoritmului? $\Theta(1)$

Vector Dinamic - iterator - element

- Ce ar trebui să facă operația *element*?

Vector Dinamic - iterator - element

- Ce ar trebui să facă operația *element*?

funcție element () **este:**
  returnează this.vect.elemente[this.curent]
sf_funcție

- Cât este complexitatea funcției?

Vector Dinamic - iterator - element

- Ce ar trebui să facă operația *element*?

funcție element () **este:**
  returnează this.vect.elemente[this.curent]
sf_funcție

- Cât este complexitatea funcției? $\Theta(1)$

Vector Dinamic - iterator - următor

- Ce ar trebui să facă operația *următor*?

Vector Dinamic - iterator - următor

- Ce ar trebui să facă operația *următor*?

subalgoritm următor() **este:**
  this.curent = this.curent + 1
sf_subalgorithm

- Cât este complexitatea subalgoritmului?

Vector Dinamic - iterator - următor

- Ce ar trebui să facă operația *următor*?

subalgoritm următor() **este:**
  this.curent = this.curent + 1
sf_subalgorithm

- Cât este complexitatea subalgoritmului? $\Theta(1)$

Vector Dinamic - iterator - valid

- Ce ar trebui să facă operația *valid*?

Vector Dinamic - iterator - valid

- Ce ar trebui să facă operația *valid*?

funcție valid() **este:**
  dacă *this.curent* $<$ *this.vect.len* **atunci**
    returnează adevărat
  altfel
    returnează fals
  sf_dacă
sf_funcție

- Cât este complexitatea funcției?

Vector Dinamic - iterator - valid

- Ce ar trebui să facă operația *valid*?

funcție valid() **este:**
  dacă *this.curent* $<$ *this.vect.len* **atunci**
    returnează adevărat
  altfel
    returnează fals
  sf_dacă
sf_funcție

- Cât este complexitatea funcției? $\Theta(1)$

Vector Dinamic - iterator - parcurgere

- Cum parcurgem un vector dinamic folosind iteratorul?

Vector Dinamic - iterator - parcurgere

- Cum parcurgem un vector dinamic folosind iteratorul?

```
subalgoritm parcurgere (vect:VectorDinamic) este:
    it: IteratorVD
    it = vect.iterator()
    câttimp it.valid() execută
        e: TElem
        e = it.element()
        @facem ceva cu e
        it.următor()
    sf_câttimp
sf_subalgoritm
```

- Cât este complexitatea parcurgerii?

Vector Dinamic - iterator - parcurgere

- Cum parcurgem un vector dinamic folosind iteratorul?

```
subalgoritm parcurgere (vect:VectorDinamic) este:
    it: IteratorVD
    it = vect.iterator()
    câttimp it.valid() execută
        e: TElem
        e = it.element()
        @facem ceva cu e
        it.următor()
    sf_câttimp
sf_subalgoritm
```

- Cât este complexitatea parcurgerii? $\Theta(n)$

Vector Dinamic - iterator

- În general când implementăm un iterator, încercăm să implementăm fiecare operație în așa fel încât să aibă complexitate $\Theta(1)$, ceea ce înseamnă că o parcurgere cu iterator va avea complexitate $\Theta(n)$.