Лінійні списки.
Методи зберігання. Операції.

Лінійний список зазвичай визначається як абстрактний тип даних, що формалізує поняття впорядкованої колекції даних.
Вікіпедія

Лінійні списки

Лінійний список - скінченна послідовність однотипних елементів (вузлів).

Кількість елементів у послідовності - довжина списку може змінюватися.

Наприклад: ***F=<7, 2, 7, 12, 17>***.

Основні операції:

- пошук елемента з заданими властивостями;
- визначення і-того елемента;
- додавання елемента до, або після вказаного;
- вилучення певного елемента зі списку;
- впорядкування елементів списку.

Задача - забезпечити максимальну ефективність обробки.

Лінійні списки

Методи збереження списків:

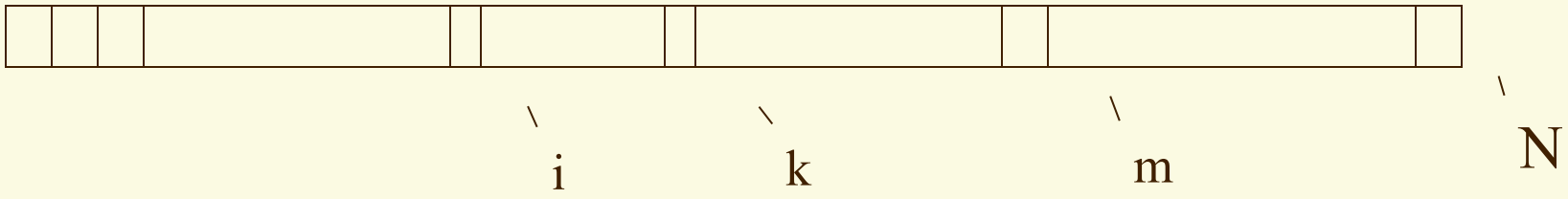
- послідовні (масив D з n елементів та змінна, що вказує довжину);
- зв`язані (структури зв`язані у ланцюг полями-покажчиками).

При обранні методу збереження слід враховувати які операції і з якою частотою будуть виконуватися над списком.

Послідовне зберігання списків

Наприклад:

```
const int N = 100;
int d[N];
int  m, i, k;
     …
d[i] = d[k];
     ...
```



Послідовне зберігання списків

- Пошук i-того елемента та його сусідів:
  d[i-1], d[i-2], d[i] (i>0; i<m) t=O(1).
- Вилучення елемента наступного за i-тим:
  for (j=i+1; j<m; j++) d[j-1] = d[j]; m--;
  t=O(m).
- Додавання елемента після i-того:
  for (j=m; j>0; j--) d[j] = d[j-1];
  d[i] = dnew; m++; t=O(m).
- Впорядкування $a_1,\ldots a_k \rightarrow a_1',\ldots,a_s', a_1, a_1'',\ldots,a_t''$
  ($a_1',\ldots,a_s' < a_1$ та $a_1 \leq a_1'',\ldots,a_t''$) t=??? .

Часткове впорядкування

// $a_1,\ldots a_k \rightarrow a_1',\ldots,a_s', a_1, a_1'',\ldots,a_t''$ $(a_1',\ldots,a_s' < a_1$ та $a_1 \le a_1'',\ldots,a_t'')$

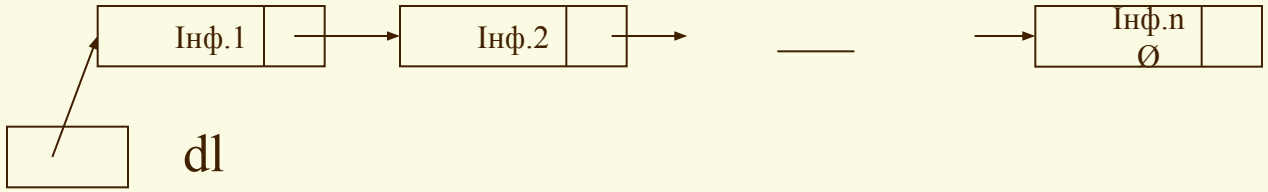
```
void ptar(){
 int t=0, dv;
 for (int i=1; i<m; i++){
   if (d[i]<d[t]){
      dv = d[i];
      for (int j=i; j>0; j--) d[j] = d[j-1];
      t++; d[0] = dv; }
}
//
```

$t=O(m^2)$

Види послідовного зберігання лінійних списків

- Також для роботи зі списком використовують послідовне зберігання, але без прив`язки до першого елементу масиву (початок списку не обов`язково у першому елементі масиву).
- Також використовують послідовне зберігання, коли в одному масиві розміщується кілька списків (кожний фіксується своїм набором параметрів). При роботі може виникати необхідність у перерозподілі вільної пам`яті масиву між кількома списками.

Зв`язане зберігання списків



Наприклад:

```
typedef struct Node {int dat;
                     Node *next;} Listn, *Listp;
Listp dl; //покажчик першого елемента
```

Доступ:

```
Listp p;
p = dl;  p->dat = …;   p->next = …;
```

Пошук *i*-го елемента

```
//повертає покажчик на і-тий елемент,
//або NULL, якщо відсутній
Listp lfnd(Listp dl, int i){
  for (int j=1; j<i && dl; j++) dl = dl->next;
  return dl;
}



//                                                    t=O(L)
//де L – кількість елементів (довжина списку)
```

Пошук сусідів елемента з покажчиком *p*

```
//виведення сусідів вузла з покажчиком p
void lprnt(Listp dl, Listp p){
  if (!p || dl==p || !(p->next)){
    cout << "no neighbouring" << endl; return;}
  while (dl && (dl->next != p)) dl = dl->next;
  if (dl)
     cout << dl->dat << ' ' << (p->next)->dat << endl;
  else cout << "no neighbouring" << endl;
} //можна казати й про відсутність лівого або правого
//                                                  t=O(L)
```

Вилучення елемента наступного за вузлом з покажчиком *p*

```
void ldel(Listp p){
  Listp r;
  if (!(r = p->next)){
    cout << "no next element" << endl; return;}
  p->next = r->next; delete r;
}



//                                  t=O(1)
```

Додавання елемента за вузлом з покажчиком *p*

```
void linst(Listp p, int dv){
  Listp r;
  r = p->next;
  p->next = new Listn;
  (p->next)->dat = dv;
  (p->next)->next = r;
}
//                                  t=O(1)
```

//??? Додавання елемента перед вузлом з покажчиком ***p***

Часткове впорядкування

```
Listp lptar(Listp dl){
  Listp p=dl, r;
  int dv=dl->dat;
  while (r = p->next)
    if (r->dat < dv){
      p->next = r->next;
      r->next = dl; dl = r;
    } else p = r;
  return dl;
}
//
```

t=O(L)

Приклад

На вході послідовність цілих чисел з інтервалу 1 - 999. Написати функцію для введення цієї послідовності й формування впорядкованого списку у зв`язаному зберіганні.

При побудові впорядкованого списку, будемо вводити черговий елемент послідовності у відповідне місце списку. Для уніфікації обробки ситуацій та спрощення перевірок можна організувати початковий список вигляду F= <0, 1000>.

Приклад

```
Listp larrange(){
  int in;
  Listp t, r = new Listn;
  r->dat = 0; r->next = new Listn;
  (r->next)->dat = 1000; (r->next)->next = NULL;
  cin >> in;
  while (in>0 && in<1000) {Listp p;
    for (p=r; (p->next)->val < in; p=p->next);
    t = p->next; p->next = new Listn;
    (p->next)->dat = in; (p->next)->next = t;
    cin >> in;} return r;
} // ??? складність
```

Види зв`язного зберігання лінійних списків

- “Звичайний” однозв`язний лінійний список;
- Двузв`язний лінійний список;
- Циклічний список;
- Список “з головою”.
- Комбінації розглянутих видів.

Крім того:

- інформація може бути розташована у елементах списку;
- елементи списку містять покажчики на інформаційні об`єкти.

Зауваження

- При розгляді послідовних способів обмежились представленням кожного списку у окремому масиві з фіксованим розташуванням вузлів починаючи з першого елемента масиву.
- При розгляді зв`язних способів представлення обмежились – лінійними однозв`язними списками з інформацією безпосередньо у елементах.
- Аналогічним чином можна реалізувати дії зі списками при інших послідовних та зв`язних способах представлення.

Підсумки

- Термін список розглядали як математичну (алгоритмічну) структуру даних, для якої можливі різні програмні представлення.
- При розгляді основних операцій зі списками з урахуванням способу представлення аналізували часову складність операції.
- Ємкісна складність при послідовних представленнях визначається розміром масиву, при зв`язних – довжиною списку.
- Складність операцій залежить від обраного способу представлення.

Поради

- При обранні методу збереження для списку слід враховувати які операції і з якою частотою будуть виконуватися над списком.
- Не зловживати рекурсією.
- Не зловживати “трюкачеством”.
- Розумним чином використовувати такі можливості програмування як структурованість та модульність.

Задачі

- Оформити бібліотеки для виконання основних операцій обробки списків:
  - послідовне зберігання списків;
  - зв`язне зберігання списків.
- Послідовність F цілих чисел $a_1 \le a_2 \le \ldots \le a_k$ та стек V вільних ділянок пам`яті реалізовані як списки у зв`язаному зберіганні з покажчиками af та av на їх початки. Написати функцію, яка для кожного i, $1 \le i \le k-1$, де $a_i = a_{i+1}$, вилучає a_{i+1} зі списку F і приєднує звільнену ділянку пам`яті до стека V.

Задачі

- У файлі задана послідовність цілих додатних чисел. Написати програму для запам`ятовування послідовності у вигляді зв`язаного списку W, друкування її у зростаючому порядку шляхом визначення найменшого елемента в W та його вилучення після друкування до вичерпання списку W.
- Написати функцію для впорядкування списку при:
  - послідовному зберіганні;
  - зв`язному зберіганні.

Задачі

- Розглянути роботу зі списком (основні операції), для якого застосовується послідовне зберігання, але без прив`язки до першого елемента масиву (початок списку не обов`язково у першому елементі масиву).
- Розглянути роботу зі списками (основні операції) при послідовному зберігання, коли в одному масиві розміщується кілька списків (кожний фіксується своїми параметрами). При роботі може виникати необхідність у перерозподілі вільної пам`яті масиву між списками.