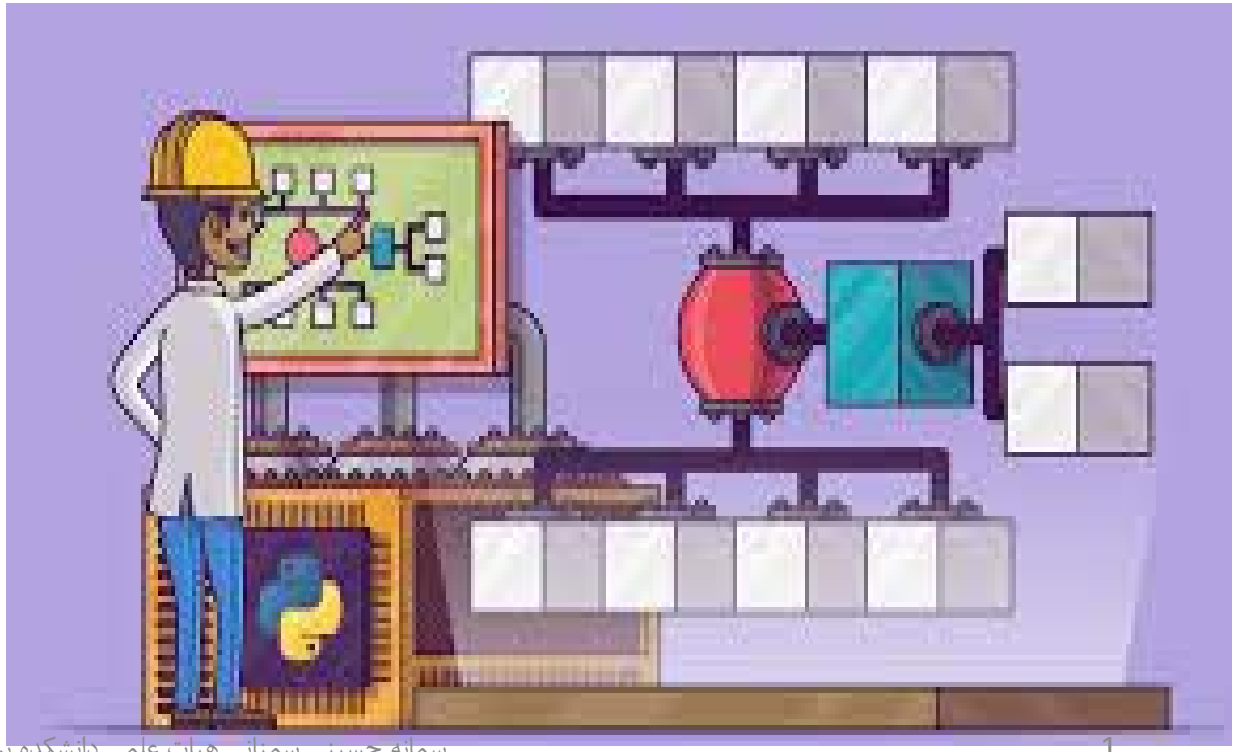
ساختمان داده ها

مدرس:
سمانه حسینی سمنانی

دانشگاه صنعتی اصفهان- دانشکده برق و کامپیوتر



آرایه ها

- لیست
- آرایه
- چند جمله‌ای
- ماتریس خلوت

Abstract Data Type

- ساختمان داده: یک روش خاص از ذخیره داده ها طوری که بتوان به صورت کارا از آن استفاده کرد:
  - زمان اجرا
  - حافظه
- ساختمان داده
  - طراحی
  - Abstract Data Type: نوع داده ای است که در ان مشخصات داده ها و اعمال بر روی آنها از بازنمایی و پیاده سازی داده جدا می شود
  - پیاده سازی
- C++
  - Class

لیست ها

- لیست مرتب شده: ordered list

Ordered (linear) list: ($item_1$, $item_2$, $item_3$, …, $item_n$)

- مثال
  - (Sunday, Monday, Tuesday, Wednesday, Thursday, Friday, Saturday)
  - (1941, 1942, 1943, 1944, 1945)
  - (a_1, a_2, a_3, …, a_{n-1}, a_n)

لیست ها

- طراحی
  1. پیدا کردن طول یک لیست
  2. خواندن اقلام داده یک لیست از چپ به راست یا بر عکس
  3. بازیابی i امین عنصر از یک لیست
  4. تعویض یک قلم اطلاعاتی در i امین موقعیت یک لیست
  5. درج یک قلم داده جدید در i امین موقعیت یک لیست
  6. حذف یک قلم اطلاعاتی از i امین موقعیت یک لیست
- پیاده سازی
  - نگاشت ترتیبی sequential mapping (1)~(4) **مثلا با استفاده از آرایه**
  - نگاشت غیر ترتیبی non-sequential mapping (5)~(6) **مثلا با استفاده از لیست پیوندی**

ADT آرایه

آرایه به عنوان یک نوع داده مجرد

- آرایه: مجموعه ای پشت سر هم از خانه های حافظه
- تعریف نامناسب: به روشنی جزئیات پیاده سازی را نشان می دهد
- آرایه: مجموعه ای از زوج ها به صورت $< index, term >$
- برای هر اندیس یک مقدار متناظر وجود دارد: Mapping
- اعمال:
  - دریافت
  - ذخیره

آرایه به عنوان یک نوع داده مجرد

```
class GeneralArray {
// A set of pairs <index, value> where for each value of index in Indexét
// there is a value of type float. IndexSet is a finite ordered set of one or more
// dimensions, for example, {0, ... , n – 1} for one dimension, {(0, 0)
// (0, 1), (0, 2), (1, 0), (1, 1), (1, 2), (2, 0), (2, 1), (2, 2)} for two dimensions. etc.
public:
     GeneralArray (int j, RangeList list, float initValue = defaultValue);
     //This constructor creates a j dimensional array of floats; the

     // range of the kth dimension is given bye the kth element of list. For each
     // index i in the index set, insert <i, initValue> into the array.

     float Retrieve (index i);
     // If i is in the index set of the array, return the float associated with i
     // in the array; otherwise throw an exception.

     void Store (index i, float x);
     // If i is in the index set of the array, replace the old value associated with i
     // by x; otherwise throw an exception.
}; //
```

تعداد ابعاد آرایه — طول هر بعد — نوع عناصرآرایه — مقادیر پیش فرض

تفاوت GeneralArray با آرایه ساده

- کلی تر از آرایه معمولی در c++
- در آرایه معمولی اندیس ها مجموعه اعداد صحیح پشت سر هم هستند ولی در GeneralArray اینطور نیست.
- چک اعتبار اندیس در آرایه معمولی انجام نمی شود

ADT چند جمله ای

نوع داده مجرد polynomial (چند جمله ای)

- ساخت یک نوع ADT برای نمایش و پردازش چندجمله‌ای

$$a(x) = 3x^2 + 2x - 4$$

$$b(x) = x^8 + 10x^5 - 3x^3 + 1$$

$$a(x) = \Sigma a_i x^i \,,\; b(x) = \Sigma b_i x^i$$

$$a(x) + b(x) = \Sigma(ai + bi)xi$$

$$a(x) \cdot b(x) = \Sigma(a_i x^i \cdot \Sigma(b_j x^j))$$

- به صورت مشابه می توان تفریق و تقسیم چند جمله ایها و بسیاری از عملیات دیگر را تعریف کرد.

نوع داده مجرد polynomial (چند جمله ای)

```
class Polynomial {
// p (x) = a0 x^e0 + ... + an x^en; a set of ordered pairs of < ei, ai >,
// where ai is a nonzero float coefficient and ei is non-negative integer exponent.
Public:
   Polynomial();
   // Construct the polynomial p(x) = 0

   Polynomial Add(Polynomial poly);
   // Return the sum of the polynomials *this and poly.

   Polynomial Mult(polynomial poly);
   // Return the product of the polynomials *this and poly.

   float Eval(float f);
   // Evaluate the polynomial *this at f and return the result.
};
```

پیاده سازی Polynomial ADT

- نمایش اول:
  - ضرایب به ترتیب نزول درجه در ارایه ذخیره شود
  - مزیت: انجام ساده عملیات جمع و تفریق و ضرب و ..
  - عیب: اگر $a.degree \ll MaxDegree$ ← هدر رفتن حافظه

a.degree=n,
$a.coef[i]=a_{n-i}$

```
class Polynomial
{
private:
    int degree;
    float Coef[MaxDegree+1];
};
```

پیاده سازی Polynomial ADT

- نمایش دوم:
  - آرایه coef را به گونه ای در نظر بگیریم که طول آن برابر a.degree+1 شود

```
class Polynomial
{
public:
        Polynomial(int deg)
        {degree = deg;
        Coef = new float[deg];}
private:
        int degree;
        float *Coef;
};
```

Drawback: $2X^{1000}+1$ in this representation has 2 nonzero term but needs 1002 space

پیاده سازی Polynomial ADT

- نمایش سوم:
- فقط ضرایب غیر صفر را به همراه توان آنها ذخیره کنیم.

```
class Polynomial
{
private:
        Term *termArray;
        int terms;   //number of non zero terms
        int capacity;   //size of termArray
....
};
```

```
class Term
{
friend class Polynomial;
private:
        float coef;   //Coefficient
        int exp;      //Exponent
};
```

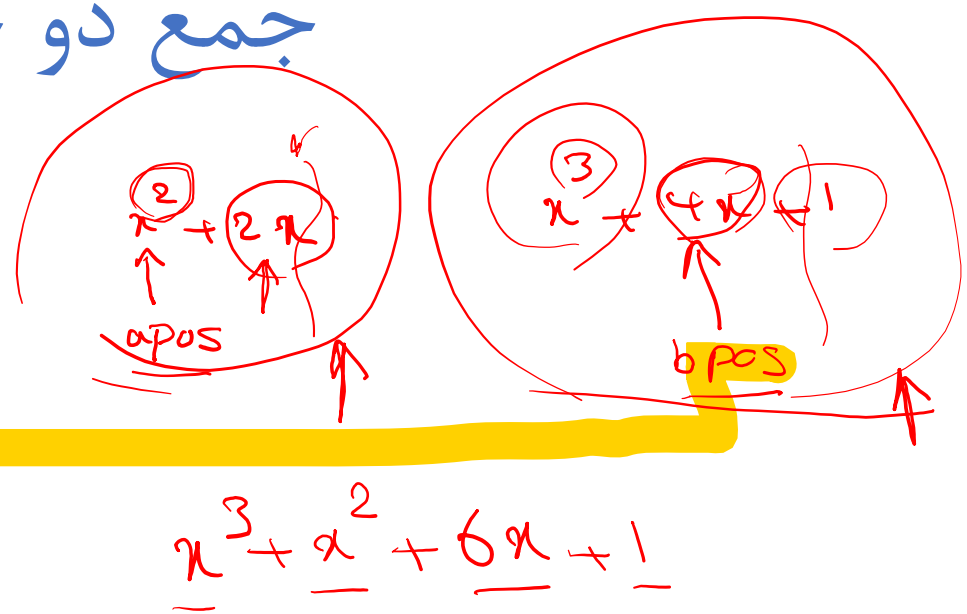
- Initial value: capacity=1, terms=0
- $2X^{1000}+1$ needs 6 space

پیاده سازی Polynomial ADT

- در چند جمله ای خلوت نمایش ۳ بهتر است
- اگر تمام جمله ها غیر صفر باشند نمایش ۳ دو برابر نمایش ۲ حافظه لازم دارد.

جمع دو چند جمله ای با نمایش ۳

```
1   Polynomial Polynomial::Add(Polynomial b)
2   { // Return the sum of of the polynomials *this and b.
3       Polynomial c;
4       int aPos = 0, bPos = 0;
5       while ((aPos < terms) && (bPos < b.terms))
6         if ((termArray [aPos].exp == b.termArray [bPos].exp) {
7                 float t = termArray [aPos].coef + b.termArray [b.Pos].coef;
8                 if (t) c.NewTerm (t, termArray [aPos].exp);
9                 aPos++; bPos++;
10        }
11        else if ((termArray [aPos].exp < b.termArray [bPos].exp) {
12                c.NewTerm (b.termArray [bPos].coef, b.termArray [bPos].exp);
13                bPos++;
14        }
15         else {
16                c.NewTerm (termArray [aPos].coef, termArray [aPos].exp);
17                aPos++;
18         }
19      // add in remaining terms of *this
20      for (; aPos < terms ; aPos++)
21          c.NewTerm (termArray [aPos].coef, termArray [aPos].exp);
22       // add in remaining terms of b (x)
23       for (; bPos < b.terms; b++);
24          c.NewTerm (b.termArray [bPos].coef, b.termArray [bPos].exp);
25       return c;
26  }
```



جمع دو چند جمله ای با نمایش ۳

```
void Polynomial::NewTerm(const float theCoeff, const int theExp)
{// Add a new term to the end of termArray.
  if (terms == capacity)
   {// double capacity of termArray
          capacity *=2;
          term *temp = new term [capacity];      //new array
          copy(termArray, termArray + terms, temp);
          delete [] termArray;                   // deallocate old memory
          termArray = temp;
   }
  termArray [term].coef = theCoeff;
  termArray [terms++].exp = theExp;
}
```


سمانه حسینی سمنانی
هیات علمی دانشکده برق و کامپیوتر- دانشگاه صنعتی اصفهان

تحلیل پیچیدگی جمع دو چند جمله ای با نمایش ۳

- m: تعداد جملات غیر صفر در آرایه اول
- n: تعداد جملات غیر صفر در آرایه دوم
- Complexity analysis:
  - $O(n+m+$ زمان صرف شده برای دو برابر کردن آرایه $)$
- Initial $c.capacity = 1$

سمانه حسینی سمنانی
هیات علمی دانشکده برق و کامپیوتر- دانشگاه صنعتی اصفهان

جمع دو چند جمله ای با نمایش ۳

$$c.capacity = 1$$
$$2$$
$$4$$
$$8$$
$$\vdots$$
$$2^k$$

$$O(\sum_{i=1}^{k} 2^i) = O(2^{k+1}) = O(2^k)$$

جمع دو چند جمله ای با نمایش ۳

$$O(\sum_{i=1}^{k} 2^i) = O(2^{k+1}) = O(2^k)$$

$$c.terms > 2^{k-1}$$

$$m + n \geq c.terms$$

$$\Longrightarrow$$

$$m + n > 2^{k-1}$$

$$\Longrightarrow$$

$$O(n + m + \text{زمان صرف شده برای دو برابر کردن آرایه}) = O(m + n)$$