فصل دوم : آرایه ها و ساختارها

اهداف

در این فصل دانشجو با کاربرد موارد زیر آشنا خواهد شد:

- آرایه
- لیست
- چند جمله‌ای
- ماتریس خلوت
- رشته

آرایه

آرایه مجموعه ای از زوج ها ، شامل اندیس و مقدار است (<index>,<value>) . به ازای هر اندیس یک مقدار مربوط به آن اندیس وجود دارد که به زبان ریاضی آنرا تناظر یا انگاشت می‌نامند.

در رابطه با آرایه به دو عمل اساسی نیاز است :

- بازیابی
- ذخیره سازی مقادیر

آرایه (Array)

- آرایه نوعی ساختمان داده است که عناصر آن هم نوع بوده و هر یک از عناصر با یک اندیس و بصورت مستقیم قابل دستیابی میباشد.
- آرایه میتواند یک بعدی و یا چند بعدی باشد
- آرایه های دو بعدی را با نام ماتریس میشناسیم

اندیس	0	1	2	3
مقدار	12	8	6	7

۱-۲ آرایه یک بعدی

آرایه در زبان C به صورت مثال در زیر آمده است :

```
int list [5]
```

نکته

در زبان C تمام آرایه ها از اندیس ۰ شروع می شوند.

آرایه ای با n عنصر با اندیسهای 0 تا n-1 میتوان به درایه‌های آن دسترسی پیدا کرد

۱-۲ آرایه یک بعدی

اگر تعریف آرایه به صورت زیر باشد:

$$A: Array\ [L_1 .. U_1]\ of\ Type$$

اگر هر نوع Type به اندازه n بایت فضا اشغال کند

آدرس خانه شروع α باشد

تعداد عناصر آرایه $= U_1 - L_1 + 1$

فضای اشغال شده توسط آرایه (فضای مورد نیاز) $= (U_1 - L_1 + 1) \times n$

آدرس خانه $A[i] = (i - L_1) \times n + \alpha$

۱-۲ جستجوی خطی در آرایه

برنامه ۱-۲

```
int search (A[n] , x)
{
  int i = 1 ;
  while (i <= n && A[i] != x){
     i + + ;
     if (i > n )
          return -1 // داده پیدا نشده
     else
          return i      // داده در محل اندیس آرایه است
  }
}
```

۲-۱ جستجوی دودویی(Binary) در آرایه مرتب

برنامه ۲-۲

```
int bsearch (A[n] , int x , int L , int U)
{
int i ;
while{
  i =(L+U)/2;
  if ( x < A[i] )
      U = i – 1;
  else if ( x > A[i] )
      L = i + 1;
  else
      return i                    //داده در اندیس i است
  }
return -1                         // داده پیدا نشده است
}
```

۱-۲ آرایه چند بعدی

- آرایه های دوبعدی یا ماتریس ها به دو روش در حافظه ذخیره می شوند:

$$\begin{bmatrix} 2 & 5 \\ 1 & 6 \\ 3 & 4 \end{bmatrix}_{3*2}$$

- سطری(Row Major)

۰	۱	۲	۳	۴	۵
2	5	1	6	3	4

- ستونی(Column Major)

۰	۱	۲	۳	۴	۵
2	1	3	5	6	4

۱-۲ آرایه چند بعدی

اگر تعریف آرایه به صورت زیر باشد:

A: Array [L_1 .. U_1, L_2 .. U_2] of Type

تعداد عناصر آرایه = $(U_1 - L_1 + 1) \times (U_2 - L_2 + 1)$

فضای اشغال شده توسط آرایه = $(U_1 - L_1 + 1) \times (U_2 - L_2 + 1) \times n$

آدرس خانه A[i,j] در روش سطری =

$$[(i - L_1) \times (U_2 - L_2 + 1) + (j - L_2)] \times n + \alpha$$

آدرس خانه A[i,j] در روش ستونی =

$$[(j - L_2) \times (U_1 - L_1 + 1) + (i - L_1)] \times n + \alpha$$

۲-۱ جمع ماتریس ها

- در جمع دو ماتریس ، حتماً باید یک ماتریس r×c با یک ماتریس r×c جمع شده و نتیجه نیزیک ماتریس r×c خواهد شد . در این عملیات عناصر دو آرایه نظیر به نظیر با یکدیگر جمع خواهند شد.

برنامه ۳-۲

```
for(i=0;i<r;++i)
  for(j=0;j<c;++j)
  {
     sum[i][j]=a[i][j]+b[i][j];
  }
```

۲-۱ ضرب ماتریس ها

- در ضرب باید بعد وسط دو ماتریس یکسان باشد، یک ماتریس r×m در یک ماتریس m×c ضرب شده و نتیجه نیزیک ماتریس r×c خواهد شد .

برنامه ۲-۴

```
for(i = 0; i < r1; ++i)
  for(j = 0; j < c2; ++j)
     {
     mult[i][j]=0;
     for(k = 0; k < c1; ++k)
     {
          mult[i][j] += a[i][k] * b[k][j];
     }
}
```

۱-۲ ماتریس خلوت(Sparse)

در علوم کامپیوتر متداول ترین نمایش برای ماتریس آرایه دوبعدی است که به صورت a[MAX_ROW][MAX_COLS] نمایش داده می شود. هر عنصر ماتریس به صورت a[i][j] نمایش داده می شود. ماتریسی که عناصر صفر آن زیاد باشد ماتریس اسپارس نامیده می شود.

حداقل اعمال ممکن شامل **ایجاد، جمع ، ضرب و ترانهاده ماتریس** میباشد.

مثالی از یک ماتریس خلوت

0	0	0	0	9
0	5	8	0	0
0	0	0	0	0

ماتریس خلوت

با توجه به ویژگیهای این ماتریس هر عضو را می توان بصورت منحصر بفردی با سه تایی <row,col,value> مشخص نمود.

سه تایی های بدست آمده بر اساس سطرها مرتب هستند و سپس عناصری که در یک سطر قرار دارند به ترتیب شماره ستون مرتب میشوند.

کلاس مربوط ماتریس‌های خلوت

```
class SpMtx;    //Sparse Matrix
class
class MTerm
{
friend class SpMtx;
private:
            int row,col,value;
};
```

```
class SpMtx
{
private:
            int Rows,Cols,Terms;
            MTerm smArr[MaxTerms];
};
```

0	0	0	0	9
0	5	8	0	0
0	0	0	0	0

⇔

Row	Col	Value
0	4	9
1	1	5
1	2	8

ترانهاده یک ماتریس

برای پیدا نمودن ترانهاده یک ماتریس باید جای سطرها و ستون ها را عوض کرد بدین مفهوم که هر عنصر *a[i][j]* در ماتریس اولیه به عنصر*b[j][i]* در ماتریس ترانهاده تبدیل می شود.

✓ الگوریتم زیر برای پیدا کردن ترانهاده یک ماتریس، الگوریتم مناسبی است :

for all element in column j

place element < i ،j ،value> in

element < j ،i ،value>

۲-۱ ترانهاده ماتریس ها

- برای محاسبه ترانهاده یک ماتریس، جای سطرها و ستون ها عوض می شوند.

برنامه ۲-۵

```
for(i = 0; i < r; ++i)
  for(j = 0; j < c; ++j)
  {
     trans[j][i]=a[i][j];
  }
```

```
SpMtx SpMtx::Transpose()
{
SpMtx b;
b.Rows = Cols;
b.Cols = Rows;
b.Terms = Terms;
if(Terms<=0)
   return b;
int CurB=0;
for(int c=0 ; c<Cols ; c++)
   for(int i=0 ; i<Terms ; i++)
      if(smArr[i].Col == c)
      {
        b.smArr[CurB].row=c;
        b.smArr[CurB].col=smArr[i].row;
        b.smArr[CurB].value=smArr[i].value;
        CurB++;
      }
return b;
}
```

ترانهاده یک ماتریس

الگوریتم بیان شده نشان می دهد که باید تمام عناصر در ستون ۰ را پیدا و آنها را در سطر ۰ ذخیره کرد همچنین تمام عناصر ستون ۱ را پیدا و در سطر ۱ قرار داد و همین فرآیند را ادامه داد. از آنجا که ماتریس اولیه سطری بوده لذا ستون های داخل هر سطر از ماتریس ترانهاده نیز به صورت صعودی مرتب می شود.

۱-۲ نمایش چندجمله ای ها

- ما نیازمند ساخت یک نوع داده ای تجریدی برای نمایش و پردازش چندجمله‌ای نمادین هستیم.
- منظور از نمادین لیستی از ضرایب و توانهاست که با هم چند جمله ای را تشکیل میدهند.

$$A(x) = 3x^2 + 2x + 4$$

$$B(x) = x^4 + 10x^3 + 3x^2 + 1$$

چند جمله‌ای ها (ادامه...)

- بزرگترین توان x در چند جمله ای را درجه چند جمله ای میگویند.
- P_A درجه چند جمله ای A(x)

$$A(x) = \sum_{i=0}^{P_A} a_i x^i$$

$$B(x) = \sum_{i=0}^{P_B} b_i x^i$$

$$A(x) + B(x) = \sum_{i=0}^{\max(P_A, P_B)} (a_i + b_i) x^i$$

$$A(x).B(x) = \sum_{i=0}^{P_A} \left(a_i x^i . \sum_{j=0}^{P_B} b_j x^j \right)$$

پیاده سازی چند جمله ایها

- استفاده از آرایه
  - در این روش اندیس آرایه معرف توان X و مقدار ذخیره شده معرف ضریب میباشد.
  - مثال:

0	1	2	3
12	8	0	7

$\Leftrightarrow \quad 7x^3+8x+12$

۱-۲ نمایش چندجمله ای ها

$$P(x) = a_n x^n + a_{n-1} x^{n-1} + \ldots + a_2 x^2 + a_1 x + a_0$$

- می توان ضرایب یک چند جمله ای را در یک ماتریس یک بعدی ذخیره کرد

a_0	a_1	...	a_{n-2}	a_{n-1}	a_n

- مشکل این روش این است که اگر ضرایب تعداد زیادی از جملات صفر باشد(جمله وجود نداشته باشد)، فضای زیادی به هدر می رود.
- راه حل: ذخیره در ماتریس دو بعدی(سطر توان و سطر ضرایب)

$$P(x) = 7x^{100} - 2x^2 + 4$$

4	-2	7
0	2	100

چند جمله ای های تنک (خلوت)

- در این چندجمله‌ای ها ضرایب صفر زیاد هستند مثلاً $x^{1000} + 1$ دارای 999 صفر است.
- برای ذخیره چند جمله‌ای های خلوت از ساختار زیر استفاده میکنیم
- عناصر بر اساس قوای صعودی و یا نزولی مرتب میشوند

0	300	
8	7	

$\Leftrightarrow \quad 7x^{300}+8$

کلاس مربوط به چند جمله‌ای

- بطور کلی اگر به صورت کلاس بنویسیم کلاس چند جمله ای به شکل زیر خواهد بود.

پیاده سازی ۱

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

```
////////////////////////////////////////////////////////////////////
```

پیاده سازی ۲

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

کلاس مربوط به چند جمله‌ای‌های خلوت

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

```
class Polynomial
{
private:
        int terms;
        Term termArr[MaxTerms];
public:
        Polynomial()
        {
                terms = 0;
                termArr[0]=0;
        }
};
```

جمع دو چند جمله ای خلوت

```
Polynomial Polynomial::Add(const Polynomial& B)
{
Polynomial c;
int a=0 , b=0;
float temp;
while(a<terms && b<B.terms)
{
  switch (compare(termArr[a].exp,B.termArr[b].exp))
  {
    case '=':
                temp = termArr[a].coef+B.termArr[b].coef;
                if(temp)
                  c.NewTerm(temp, termArr[a].exp);
                a++;  b++;
                break;
    case '<':
                c.NewTerm(B.termArr[b].coef, B.termArr[b].exp);
                b++;
                break;
    case '>':
                c.NewTerm(termArr[a].coef, termArr[a].exp);
                a++;
                break;

  }
}
for( ; a<terms; a++)
                c.NewTerm(termArr[a].coef, termArr[a].exp);
for( ; b<B.terms; b++)
                 c.NewTerm(B.termArr[b].coef, B.termArr[b].exp);
return c;
}
```

```
void Polynomial::NewTerm(float c , int e)
{
   if(terms >= MaxTerms)
   {
      cerr<<"Too many terms in polynomial\n";
      return;
   }
   termArr[terms].coef = c;
   termArr[terms].exp = e;
   terms++;
}
```

نوع داده مجرد رشته ای (STRING ADT)

از دیدگاه *ADT* یک رشته به صورت $S = s_0,\ldots,s_{n-1}$ تعریف می گردد به نحوی که s_i کاراکترهای اخذ شده از مجموعه کاراکترهای زبان برنامه نویسی می باشد. اگر *n=0* باشد، S یک رشته تهی می باشد.

در زبان C ، رشته ها به صورت آرایه های کاراکتری که به کاراکترتهی '\0' ختم می شوند ،آرایه می گردد.

نوع داده مجرد رشته ای (STRING ADT)

عملکردهای مناسب و مفیدی وجود دارد که می توان برای رشته ها تعریف کرد مانند :

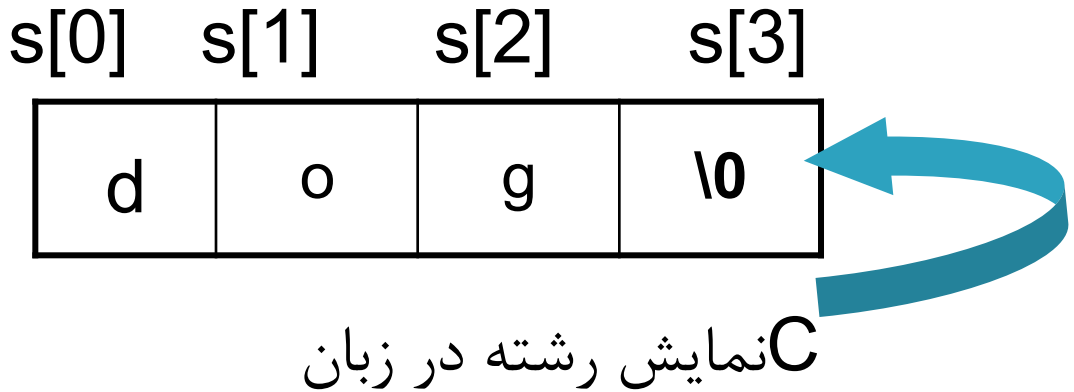
- ایجاد یک رشته تهی جدید
- خواندن یا نوشتن یک رشته
- ضمیمه کردن دو رشته به یکدیگر *(concatenation)*
- کپی کردن یک رشته
- مقایسه رشته ها
- درج کردن یک زیر رشته به داخل رشته
- برداشتن یک زیر رشته از یک رشته مشخص
- پیدا کردن یک الگو(pattern) یا عبارت در یک رشته

مختص ADT جدید

نوع داده مجرد رشته ای (STRING ADT)

نحوه ذخیره سازی در حافظه :

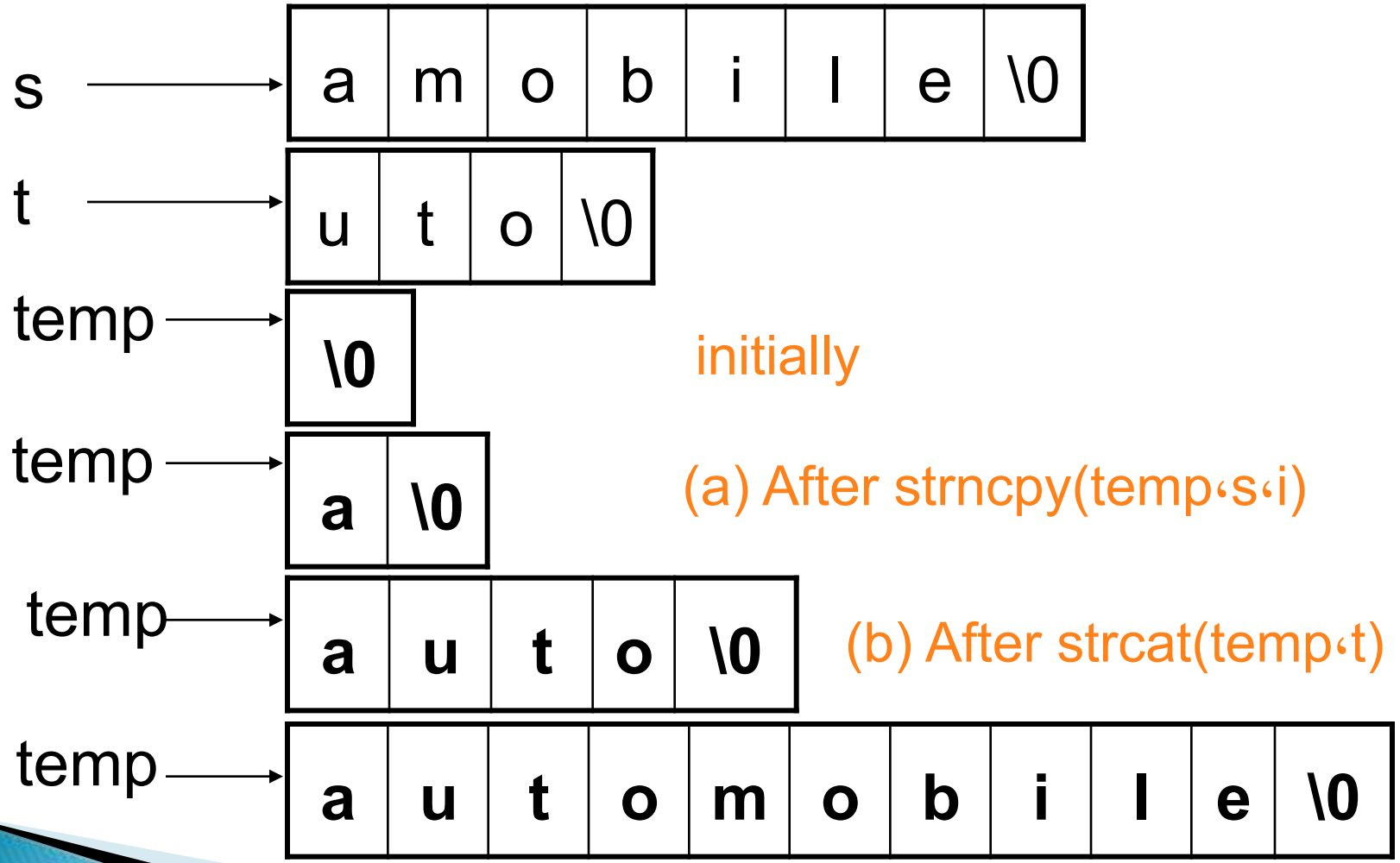
```
char  s[] = "dog”;
```



نمایش رشته در زبانC

مثال : درج رشته

می خواهیم رشته t را در موقعیت i=1 رشته s جای دهیم :



تطابق الگو(Pattern Matching)

فرض کنید که دو رشته داریم ، string وpat، به نحوی pat یک الگو یا pattern بوده و باید در string پیدا شود. ساده ترین راه برای تعیین اینکه آیا pat در رشته وجود دارد یا خیر، استفاده از تابع کتابخانه ای strstr می باشد.

با وجود اینکه strstr برای تطابق عبارت مناسب به نظر می رسد ، دو دلیل عمده برای نوشتن تابع تطابق الگو وجود دارد :

- تابع strstr برای *ANSI C.* جدید بوده و ممکن است که در کامپایلر موجود نباشد.
- چندین روش برای پیاده سازی تابع تطابق الگو وجود دارد.

تطابق الگو

نکته

غیر موثرترین روش ، تست متوالی هر کاراکتر رشته تا زمان پیدا شدن الگو و یا رسیدن به انتهای رشته ، می باشد. اگر substr در str نباشد، این روش دارای زمان محاسباتی $0(n \times m)$ خواهد بود که در آن n طول substr و m طول str می باشد.

برنامه ۶-۲

```
for(i=0;str[i]!='\0';i++)
{
    j=0;
    if(str[i]==substr[j])
    {
        temp=i+1;
        while(str[i]==substr[j])
        {
            i++;
            j++;
        }
         if(substr[j]=='\0')
        {
            cout<<"The substring is at "<<temp<<"\n";
            break;
        }
        else
        {
            i=temp;
            temp=0;
        }
    }
}
```

تمرین ۳

- برنامه ای برای جستجو در رشته با مرتبه اجرایی O(n+m) بنویسید.

(زمان تحویل هفته بعد)