فصل دوم:

آرایه ها و ساختارها

- *آشنایی با آرایه*
- *آشنایی با لیست*
- *پیاده سازی چند جمله‌ای ها*
- *آشنایی با ماتریس خلوت*
- *آشنایی با رشته*

آرایه (Array)

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

- آرایه میتواند یک بعدی و یا چند بعدی باشد
- آرایه های دو بعدی را با نام ماتریس می شناسیم

آرایه (Array)

اندیس	0	1	2	3
مقدار	12	8	6	7

آرایه یک بعدی

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

int list [5]

- در زبان C تمام آرایه ها از اندیس صفر شروع می شوند.
- آرایه ای با n عنصر با اندیسهای 0 تا n-1 می توان به درایه‌های آن دسترسی پیدا کرد

آرایه یک بعدی

اگر تعریف آرایه به صورت زیر باشد:

A: Array [L1 .. U1] of Type

اگر هر نوع Type به اندازه n بایت فضا اشغال کندو آدرس خانه شروع α باشد

تعداد عناصر آرایه $= U_1 - L_1 + 1$

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

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

جستجوی خطی در آرایه

```
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
   }
}
```

جستجوی دودویی(Binary) در آرایه مرتب

```
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
}
```

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

آرایه چند بعدی

اگر تعریف آرایه به صورت زیر باشد:

A: Array [L1 .. U1, L2 .. U2] 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 =

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

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.

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

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 .

```
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];
    }
}
```

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 خلـوت نامیده می شود.

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

ماتریس خلوت– مثال

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

ماتریس خلوت– مثال

با توجه به ویژگی های این ماتریس هر عضو را می توان بصورت منحصر به فردی 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 ستون مرتب می شوند.

ماتریس خلوت– مثال

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 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 عنصرb[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>

ترانهاده ماتریس

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

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

نمایش چندجمله ای ها

- ما نیازمند ساخت یک نوع داده انتزاعی برای نمایش و پردازش چندجمله‌ای نمادین هستیم.
- منظور از نمادین، لیستی از ضرایب و توان هاست که با هم چند جمله ای را تشکیل می دهند.

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

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

پیاده سازی چندجمله ای ها

- **استفاده از آرایه**
  - *در این روش اندیس آرایه معرف توان x و مقدار ذخیره شده معرف ضریب می باشد.*
  - *مثال:*

0	1	2	3
12	8	0	7

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

نمایش چندجمله ای ها

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

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

a_0	a_1	⋯	a_{n-2}	a_{n-1}	a_n

نمایش چندجمله ای ها

- مشکل این روش این است که اگر ضرایب تعداد زیادی از جملات صفر باشد(جمله وجود نداشته باشد)، فضـای زیـادی به هـدر 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 خلوت

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

0	300
8	7

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

نوع داده انتزاعی رشته (STRING)

- 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 کاراکتر تهی "\0" ختم می شوند،پیاده سازی می شوند.

نوع داده انتزاعی رشته (STRING)

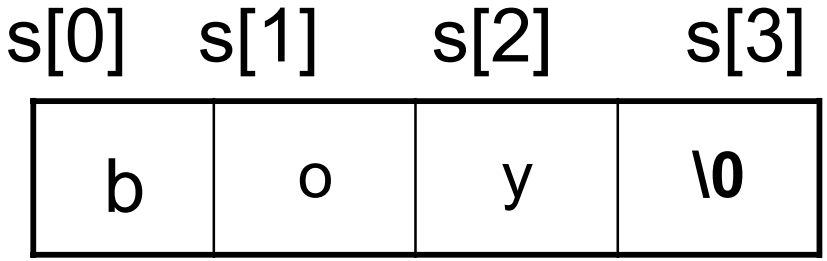
عملکردهای مناسب و مفیدی می توان برای رشته ها تعریف کرد مانند :

- 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

نوع داده انتزاعی رشته (STRING)

نحوه ذخیره سازی در حافظه :

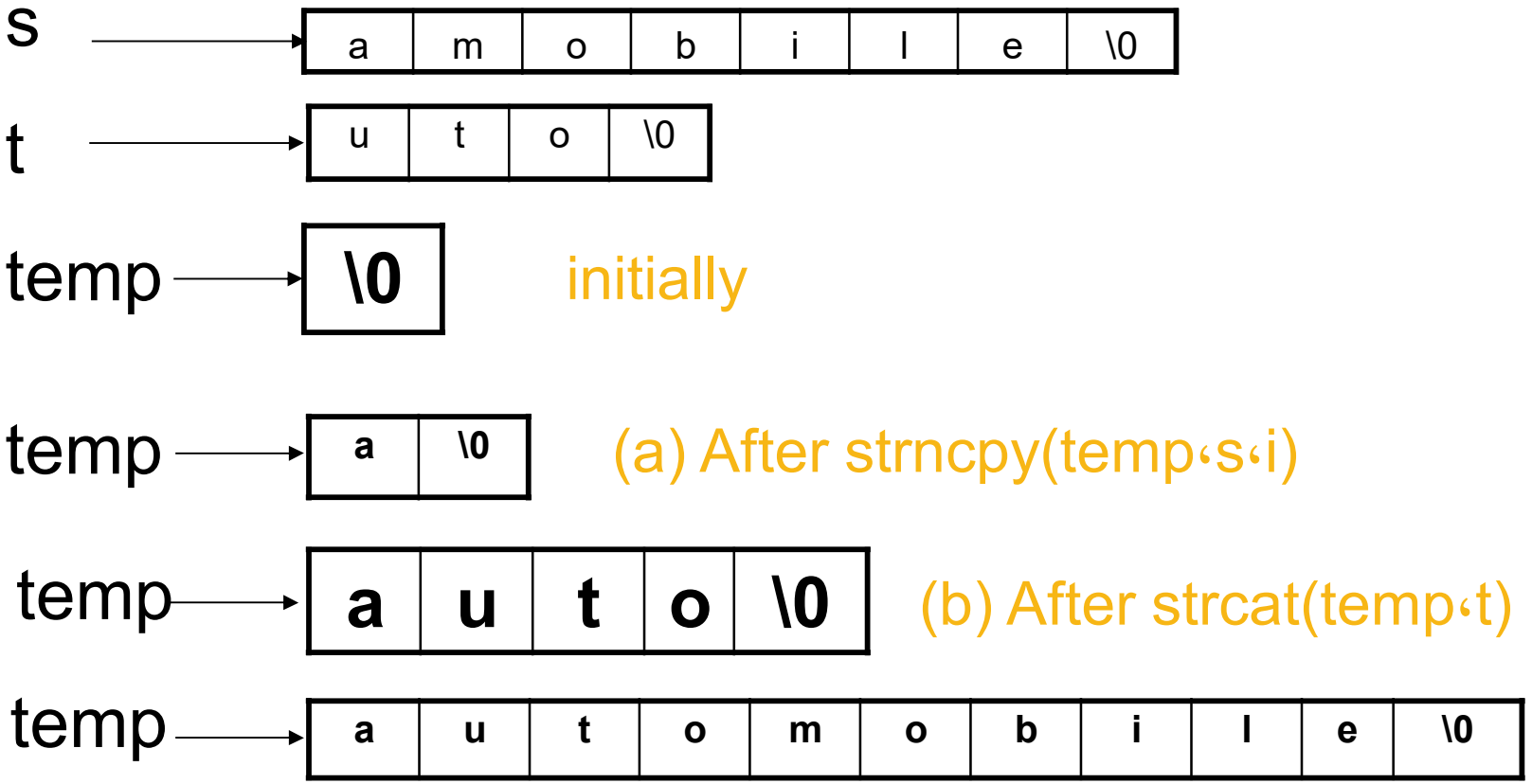
```
char  s[] = “boy”;
```



نمایش رشته در زبان C

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 string و pat داریم، به نحوی pat یک الگـــــو 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(Pattern Matching)

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 $\theta(n\times m)$ خواهد بود که در آن n طول substr و m طول strمی باشد.

تطابق الگو(Pattern Matching)

```
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

برنامه ای برای جستجو در رشته با مرتبه اجرایی $\theta(n+m)$ بنویسید.

(زمان تحویل هفته آینده)