**بنام خدا**

# ساختمان داده‌ها

**رحمت قاسمی**
ghasemi@iauneka.ac.ir

نکوهش مکن چرخ نیلوفری را  برون کن زسر باد خیره سری را
درخت تو گر بار دانش بگیرد  به زیر آوری چرخ نیلوفری را

## ساختمان داده چیست؟

ساختمان داده روشی برای ذخیره سازی و چینش داده ها و اطلاعات در حافظه کامپیوتر است به نحوی که به طور موثر تری بتوانیم عمل پردازش روی داده ها را انجام دهیم. یعنی از حافظه و پردازش گر به صورت بهینه استفاده شود. غالباً **انتخاب یک ساختمان داده موجب ایجاد الگوریتم های متناسب با آن خواهد شد که این دو در کنار هم موجب افزایش سرعت انجام یک وظیفه یا کاهش مصرف حافظه برای پردازش داده می شود. از ساختمان داده های بنیادین می توان به آرایه و لیست پیوندی اشاره کرد. مثالهای دیگری از ساختمان داده می تواند رشته، صف، پشته، درخت، گراف باشد.**

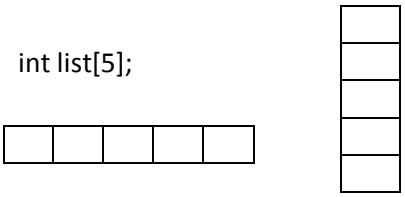
## آرایه:

تعریف: لیستی از عناصر هم نوع است که در خانه های پشت سر هم حافظه ذخیره شده اند. هرعنصرآرایه توسط نام آرایه و یک اندیس عددی قابل دسترسی است.

آرایه یکی از مهمترین ساختمان داده ها می باشد، که از آن می توان برای ایجاد ساختمان داده های دیگر مانند پشته، صف و درخت استفاده نمود. بنابراین مطالعه و بررسی آرایه و نحوه ذخیره سازی آن در حافظه مهم و اجتناب ناپذیر می باشد.

عملیات مختلفی روی آرایه می توان انجام داد که می تواند شامل ایجاد آرایه، درج در آرایه، حذف از آرایه و جستجو باشد.

int list[5];



اگر $a$آدرس اولین عنصریک آرایه یک بعدی باشد آنگاه آدرس i امین عنصردرآرایه یک بعدی ازفرمول زیرمحاسبه می شود:

آدرس i امین عنصر $= a + i * sizeof$ (نوع آرایه)

فرض $a = 210$ آدرس list[3] = 210 + 3 * *sizeof* (int)
= 210 + 3 * 2 = 216

آرایه چندبعدی : یک آرایه می تواندبه صورت چندبعدی تعریف شود دراین صورت تعدادعناصر آرایه ازفرمول زیرمحاسبه می شود.

$$\text{تعداد عناصر آرایه} = \prod_{i=0}^{n-1} u_i$$

که $u_i$ مشخص کننده بعد i ام آرایه می باشد.

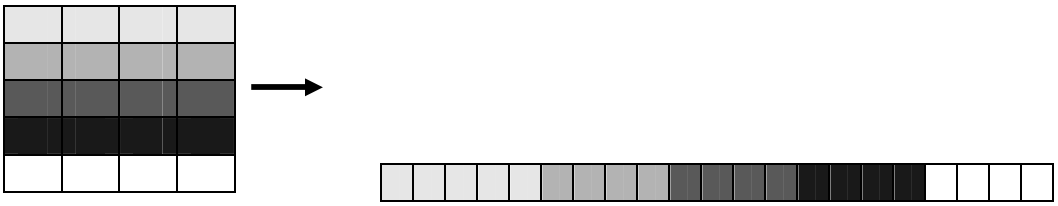
مثال : تعدادعناصر آرایه a[10][5][6] برابراست با10*5*6=300

همانطور که می دانیم حافظه اصلی کامپیوتر یک آرایه یک بعدی بسیار بزرگ است. اما مساله این است که چطور یک آرایه چند بعدی (مثلا یک آرایه دو بعدی یا سه بعدی) را در حافظـه ذخیـره کنیم. زبانهای برنامه سازی این کار را به دو روش انجام می دهند، که عبارتند از:

1- روش سطری

2- روش ستونی

فرض کنیم یک آرایه با 5 سطر و 4 ستون (مثلا int A[5][6]; ) را داشته باشیم.



اگردریک آرایه دوبعـدی بـه صـورت $a[u0][u1]$ آدرس خانـه a[0][0] برابربـا $a$ باشـدآنگاه آدرس a[i][0] عبارتست از :

آدرس $(a[i][0]) = a + i * u1$

آدرس $(a[i][j] = a + i * u1 + j$

درمثال مااگرآدرس خانه a[0][0] برابربا210 باشدآنگاه آدرس خانه a[2][3] رابدست آورید نوع آرایه راازنوع char فرض کنید.

Char a[5][4]

آدرس (a[2][3])=210+2*4+3=221

اگرنوع آرایه متفاوت باشد یعنی int یا float و ... باشد.

آدرس a[i][j]= $a$ +(i*$u_1$+j)*sizeof( نوع آرایه)

در آرایه سه بعدی:

آدرس a[i][j][k]= $a$ +(i*$u_1$+ $u_2$)+j $u_2$+k

اگر نوع آرایه متفاوت باشد:

آدرس a[i][j][k]= $a$ +(i*$u_1$+ $u_2$+k)*sizeof(نوع ارایه)

به طورکلی درآرایه چندبعدی : A[$u_o$] [$u_1$]...[$u_{n-1}$]

$$(A[i0][i1]...[in+1]) = a + (\sum_{j=0}^{n-1} i_j b_j) * sizeof\,(ArrayType)$$

$$b_j = \prod_{k=j+1}^{n-1} u_k \qquad 0 \le j < n-1$$

$$b_{n-1} = 1$$

مثال :درآرایه چهاربعدی اگر A[10][5][4][6] باشدازنوع int ,$a = 210$ موجوداست مطلوبست محاسبه آدرس A[5][2][3][4].(روش ذخیره سازی سطری است)

$$i0 = 5 \qquad u0 = 10 \qquad j = 0 \to b_0 = \prod_{k=j21}^{n-1} u_k = 5 \times 4 \times 6 = 120$$

$$i1 = 2 \qquad u1 = 5 \qquad j = 1 \to b_1 = \prod_{k=2}^{3} u_k = 4 \times 6 = 24$$

$$i2 = 3 \qquad u2 = 4 \qquad j = 2 \to b_2 = \prod_{k=3}^{3} u_k = 6$$

$$i3 = 4 \qquad u3 = 6 \qquad j = 3 \to b_3 = 1$$

| j | $u_j$ | $i_i$ | $b_j$ | $i_j$*$b_j$ |
|---|---|---|---|---|
| 0 | 10 | 5 | 5*4*6=120 | 5*120=600 |
| 1 | 5 | 2 | 4*6=24 | 2*24=48 |
| 2 | 4 | 3 | 6 | 3*6=18 |
| 3 | 6 | 4 | 1 | 4*1=4 |
| SUM | | | | 670 |

$$p = a + (\sum_{j=0}^{n-1} i_j b_j) * sizeof(\text{int})$$
$$\Rightarrow p = 210 + (5\times120 + 2\times24 + 3\times6 + 4\times1) * sizeof(\text{int})$$
$$p = 210 + (600 + 48 + 18 + 4)\times2 = 1550$$

تمرین: در مثال قبل آدرس عنصر در روش ستونی ذخیره سازی چند است؟

## ساختمان struct :

ساختمان یا مجموعه ای از اقلام داده ها می باشد، به هر قلم داده فیلد نیز می گوییم که به وسیله نام و نوع داده مشخص می شود. جهت تعریف ساختمان در زبان c از کلمه کلیدی struct استفاده می کنیم. مثلاساختمان یا Record مربوط به یک شخص می تواند به صورت زیرتعریف شود:

```
struct person {
     char  name [20];      //نام
     int  age;        // سن
     foat   salary ;       //حقوق
} person;
```

در تعریف ساختمان person سه فیلد وجود دارد، name که یک رشته کاراکتری به طول 20 بایت، age که یک عدد صحیح از نوع int است و salary که از نوع float است. اگر بخواهیم اطلاعات فردی یک نفر را ذخیره نماییم می توانیم از ساختار فوق استفاده کنیم و متغییری از نوع person تعریف کنیم.

یعنی:

```
struct person  person1;
```

یا می توانی برای ذخیره اطلاعات تعدای از افراد مثلا اطلاعات 20 نفر از یک آرایه استفاده کنیم و می نویسیم:

```
struct  person  persons[20];
```

در زبان c این امکان وجود دارد که با کلمه کلیدی **typedef یک نـوع داده جدیـد معرفـی کنیم، ینابراین مـی تـوانیم مـثلا یـک نـوع داده جدیـد مـثلا** TPerson از روی سـاختمان person تعریف کنیم.

```
typedef struct person  TPerson;
```

بنابراین میتوانیم تعریف های فوق را بازنویسی نماییم و بنویسیم:

```
TPerson person1;
TPerson persons[20];
```

*میزان حافظه مصرفی دریک ساختمان برابراست با مجموع حافظه مصرفی فیلدهای آن .

درمثال فوق ساختمان person 26 بایت حافظه رامصرف میکند( 20تابرای نـام ،2تـابرای سـن ، 4تابرای حقوق )

*ترتیب قرارگرفتن عناصرساختمان درحافظه درزبان c به صورت پشت سرهم می باشد.

عناصر ساختار میتوانند خود یک ساختار باشند، یا یک اشاره گر به یـک سـاختمان داده دیگـر یـا همان ساختار باشند. به ساختمانی که در خود یک فیلد اشاره گری از نـوع خـود سـاختار داشـته باشد، ساختار خود ارجاعی می گوییم. از این نوع ساختار ها برای ایجاد ساختمان داده های پویا مانند لیست پیوندی، درخت و ... استفاده می شود.

در زیر مجددا تعریف ساختمان person را مشاهدی می کنید، که در آن از یک فیلد اشاره گری به نام next استفاده شده است.

```
struct  person {
     char  name [20];      //نـام
     int  age;        // سن
     foat   salary ;       //حقـوق
     struct person * next;
};
```

تعداد فیلد های اشاره گر می تواند بیشتر هم باشد که بستگی به نوع ساختمان داده تعریف شده دارد. مثلا در لیست دو پیوندی، در هر گره لیست از دو اشاره گر برای ذخیره آدرس گره بعدی و گره قبلی استفاده شده است.

union : محلی ازحافظه است که توسط دویا چندمتغیر به طوراشتراکی مورداستفاده قرارمیگیرد.

```
union {
      int i;
      char ch;
      float y;
};
```

*میزان حافظه مصرفی union برابربامیزان حافظه مصرفی بزرگترین فیلدآن است .

درمثال فوق union 4 بایت حافظه مصرف کرده است .

مثال :استفاده از union

```
struct  shape{
      int type;
      int  x;
      int  y;
      union u {
      int radius;
      int lenth;
      }   u;
};

struct shape  s1,s2;

s1.type=0;
s1.x=10;s1.y=20;
s1.u.radius=100;

s2.type=1;
s2.x=15;
s2.y=18;
s2.u.lenth=50;
```

## تحلیل نحوه اجرای یک برنامه

ازدوجنبه برنامه هارامورد‌تحلیل قرارمیدهیم :

1- ازلحاظ میزان مصرف حافظه

2- ازلحاظ پیچیدگی زمان (یعنی مقدار زمانی که برای اجرای کامل برنامه لازم است )

میزان حافظه مصرفی ازدوجنبه موردبررسی قرارمی گیرد:

1- فضای ثابت : شامل فضای دستورالعملها برای ذخیره که برنامه فضای لازم برای متغیرهای ساده ،متغیرهای ساختاری بااندازه ثابت وثابت ها

2- پیچیدگی فضای متغیر : شامل فضای موردنیاز برای متغیرهای ساخت یافته مانندآرایه پویافضای اضافی برای توابع بازگشتی

نیازمندیهای فضای کل هربرنامه بصورت زیربیان می شود:

S(P)=C+SP(I)

که c نشان دهنده نیازمندی فضای ثابت است و SP(I) نشان دهنده نیازمندی های فضای متغیرمی‌باشد.

مثال

```
float  sum(float a,float b,float c)
{
        return  a+b+c;
}
```

زیربرنامه sum که مجموع سه عددرامحاسبه میکند. نیازمندی متغیرندارد ونیازمندی ثابت آن c,b,a و مقدار بازگشتی هستند که هرکدام 4بایت حافظه مصرف می کنند ودرمجموع

16 بایت که نیازمند فضای ثابت برای حافظه 16 بایت می باشد(البته بدون درنظر گرفتن فضای لازم برای کد برنامه).

مثال : مجموع لیستی ازاعدادبه صورت غیر بازگشتی.

```
float  sum(float  list[ ] ,int n)
{
        float  temp=0;
        int i;
        for(i=0;i<n;i++)
                temp+=list[i];
        return temp;
}
```

نیازمندی فضای متغیرندارد.

درزبان c مقادیر آرایه به تابع ارسال نمی شود بلکه آدرس شروع آرایه به تابع ارسال می شود، بنابراین متغییر list در تابع فوق 2 بایت مصرف حافظه دارد . ولی درزبان پاسکال آرایه به صورت مقدار منتقل می شود.

مثال :جمع لیستی ازاعدادبه صورت بازگشتی

```
float  rsum( float list[], int n)
{
        if  (n<0) return rsum(list,n-1)+list[n-1];
        return 0;
}
```

دراین برنامه نیازمندی فضای متغیرداریم (به خاطر استفاده ازپشته در فراخوانیها )

نتیجه : باتوجه به اینکه توابع بازگشتی الگوریتم رادراکثرموارد ساده ترمی سازد ولی میزان حافظه مصرفی رابیشترمیکند.

ب – پیچیدگی زمان : منظورازپیچیدگی زمان، زمان لازم برای اجرای کامل برنامه است که عبارتنداز:

مجموع زمان کمپایل + زمان اجرای برنامه

*زمان کمپایل رامی‌توان نادیده گرفت (چون یک زمان ثابتی است).

یک روش برای بدست آوردن زمان اجرای برنامه استفاده از **زمان سنج ماشین** است. در استفاده از زمانسنج ماشین، قبل و بعد از اجرای الگوریتم زمان را ذخیره کرده و تفاوت این دو زمـان، زمـان اجرای الگوریتم را نسان می‌دهد.

```
t1=clock();
        algorithm();
t2=clock();
t=t2-t1;
```

روش دیگر **شمارش تعداد خطوط اجرایی** برنامه باشد که به نوبه خود می تواند تخمینی از زمـان اجرای برنامه باشد.

مثال : شمارش تعداد خطوط اجرایی تابع جمع لیستی ازاعداد:

| | S/e | F | Total |
|---|---|---|---|
| float sum(float list[], int n) { | 0 | 0 | 0 |
| float temp=0; | 1 | 1 | 1 |
| int i; | 0 | 0 | 0 |
| for (i=0; i<n ; i++ ) | 1 | N+1 | N+1 |
| temp+=list[i]; | 1 | N | N |
| return temp; | 1 | 1 | 1 |
| } | | | 2n+3 |

مثال :شمارش تعداد خطوط اجرایی تابع جمع لیستی ازاعدادبه روش بازگشتی

| | s/e | F | Total |
|---|---|---|---|
| float rsum(float list[],int n) { | 0 | 0 | 0 |
| if(n>0) | 1 | n+1 | n+1 |
| return list[n-1]+rsum(list,n-1); | 1 | N | N |
| else return 0; | 1 | 1 | 1 |
| } | | | 2n+2 |

مثال : شمارش تعداد خطوط اجرایی تابع محاسبه جمع دوماتریس B,A (m در n).

| m , n are predefined as constants; | S/e | F | Total |
|---|---|---|---|
| float sum(float a[m][n],float b[m][n] int c[m][n]) | 0 | 0 | 0 |
| int i,j; | 0 | 0 | 0 |
| for(i=0;i<m;i++) | 1 | m+1 | m+1 |
| for(j=0; j<n; j++) | 1 | m(n+1) | m(n+1) |
| c[i][j]=a[i][j]+b[i][j] | 1 | m.n | m.n |
| } | | | 2mn+2m+1 |

**تمرین: تعداد خطوط اجرایی تابع ترانهاده ماتریس $A_{nxn}$ را محاسبه کنید.**

```
void  transpose ( int a[n][n] )
{
  for(i=1; i <n ; i++)
        for(j=0; j<i; j++) {
                swap(a[i][j],a[j][i]);
        }
}
```

روش سوم، محاسبه مرتبه زمانی الگوریتمی باشد. $(O,q,\Omega)$

**نماد O بزرگ** : فرض کنید f یک الگوریتم و n تعداد داده های ورودی می باشد. واضح است که پیچیدگی الگوریتم f با زیادشدن تعداد داده های ورودی n افزایش می یابد.

تعریف : فرض کنید f(n) , g(n) دوتابعی باشند که برروی اعداد صحیح مثبت تعریف شده ودارای این خاصیت هستندکه f(n) به ازاء تمام مقادیر n توسط مضربی از g(n) محدودشده است .یابه عبارتی عددصحیح مثبت n0 وعددمثبت c وجود دارد به گونه ای که :

$$\exists n_0, c \forall n > n_0 \qquad f(n) \le cg(n)$$

درآن صورت می توان نوشت $f(n) = O(g(n))$

یعنی f(n) ازمرتبه g(n) می باشد.

قضیه: برای چندجمله f(n) ازدرجه m می توان نوشت $f(n)=O(n^m)$

$$f(n)=a_m n^m + a_{m-1}n^{m-1} + \ldots + a_1 n^1 + a0 = \sum_{i=0}^{m} a_i n_i \le \sum_{i=0}^{m} |a_i| n_i$$

$$\le n^m \sum |a_i| \frac{n^i}{n^m} \le n^m \sum |ai| n^{i-m} \le cn^m \Rightarrow f(n)=O(n^m)$$

**پیچیدگی اجرای چندالگوریتم معروف :**

جستجوی خطی : دربهترین حالت پیچیدگی زمان آن O(1) است ودربدترین حالت O(n)

جستجوی دودوئی : پیچیدگی زمان در O(log n) بدست می آید.

مرتب سازی حبابی : پیچیدگی زمان $O(n^2)$

مرتب سازی باادغام : پیچیدگی زمان.O(nlogn)

اگر بتوان الگوریتمی نوشت که مرتبه اجرایی آن O(1) باشد آنگاه زمان اجرای الگوریتم، یک زمـان ثابتی است و به اندازه مساله بستگی ندارد.

**رشته ها** :

مجموعه ای از کاراکترهای پشت سرهم . در زبان c طول رشته ذخیره نمی شود، ولی انتهای رشته با کاراکتر '\0' یا NULL مشخص می شود . (در زبان پاسکال طول رشته در خانه صفرم رشته ذخیره می شود) . در زبان c رشته به صورت یک آرایه تعریف می شود . اشاره گر به کاراکتر نیز نوعی رشته است .

به طور مثال به تعاریف زیر دقت کنید:

```
 char s[80];
char  * t ;
```

با توجه به اینکه در زبان c ، در انتهای رشته کاراکتر '\0' قرار می گیرد، تعریف اول یک رشته به طول حداکثر 79 کاراکتر را نشام می دهد. یعنی برای s یک فضای 80 بایتی در اختصاص داده شده است. اما تعریف دوم یک اشاره گر رشته ای را مشخص می کند. این اشاره گر می تواند به آدرس ابتدای یک رشته اشاره نمایید. مثلا می توانیم بنویسیم:

```
t= s;
```

با این کار اشاره گر t به آدرس ایتدای رشته s اشاره می کند.

***عملیاتی که روی رشته ها می توان انجام داد** :

1- بدست آوردن طول رشته .

2- مقایسه دو رشته .

3- چسباندن یک رشته به انتهای رشته دیگر .

4- استخراج یک زیر رشته از یک رشته دیگر .

5- پیدا کردن یک زیر رشته در یک رشته دیگر .

6- درج یک رشته در یک رشته دیگر .

7- پیدا کردن یک کاراکتر در یک رشته .

8- معکوس کردن رشته .