

جلسه ی دوازدهم (استراکچر ها قسمت اول)

اگر مبحث آرایه هارو به یاد داشته باشد برای ذخیره لیست لیست اسامی از یک آرایه ی دو بعدی کاراکتری استفاده می کردیم حالا فرض کنید هر اسمی یک نمره هم داره و می خوایم این نمره هارو هم کنار اسامی ذخیره کنیم اولین ایده اینه که یک آرایه از نوع float داشته باشیم که عناصر یکم تا انم آن با لیست اسامی مرتبط می شه خوب تا اینجا قضیه هیچ مشکلی نیست ام حالا فرض کنید بخوایم این لیست رو بر اساس نام مرتب کنیم این مطلب واضح است که باید برای مرتب سازی عناصر آرایه رو جابجا کنیم اما مشکل اینه که باید برای هر نفر هم اسم و هم نمره فرد رو جابجا کنیم تا نمره کسی با نمره دیگری عوض نشه خوب می گیم باشه قبول چند تا دستور باید بیشتر بنویسیم دیگه!! غیر از اینه... اما اگه فیلد هامون بیشتر از دوتا بود چی؟؟ هممون می دونیم که بانک های اطلاعاتی معمولا بیشتر از دو فیلد داده ای دارن . برای نوشتن دستور های جابجایی رکودها برای مثلا یک رکورد ده فیلدی باید ۳۰ خط کد بنویسیم خوب حالا به مشکل دیگه برای فرستادن یه رکورد به یه تابع برای مثلا ده فیلد باید تابعمون ۱۰ آرگمان داشته باشه (بعد فردا پس فردا تو روزنامه ها مینویسن فردی بر اثر کد نویسی بش از اندازه جان باخت!!!!) و اینطوری شد که چیزی به نام استراکچر (structure) ساخته شد.

تعریف:

استراکچر امکاناتی است که به ما اجازه ساخت انواع داده ای جدید را از ترکیب انواع داده ای موجود می دهد.

با استفاده از این امکانت ما می توانیم چندین متغیر از انواع دیگر را در یک بسته بندی بعنوان نوع جدیدی معرفی کنیم به هر یک از این متغیر های درون این بسته بندی فیلد و به هر بسته یک استراکچر یا به علت شباهت زیاد به یک رکورد، رکورد می گویند. بعد از تعریف یک استراکچر بعنوان یک نوع داده ای جدید می توانیم با این بسته ی متغیر ها مثل یک متغیر معمولی رفتار کنیم و هر زمان که نیاز بود محتویات هر یک از متغیر های درونش یا همان فیلد ها را تغییر دهیم.

نحوه تعریف یک استراکچر:

```
struct {نام نوع داده ای جدید  
تعاریف فیلدها  
};
```

مثال:

```
struct person{  
char name[۴۰];  
float grade;  
};
```

در این مثال استراکچری بانام person تعریف شده پس از این به بعد نوع داده ای جدیدی به این نام داریم که می توانیم متغیر هایی از آن تعریف کنیم برای مثال

```
person p۱;
```

یا آرایه ای از آن تعریف کنیم

```
Person list[۲۰];
```

مقدار دهی اولیه به استراکچر ها شبیه مقدار دهی اولیه به آرایه هاست

```
person p۱={"Ali Ahmadi",۹,۷۵};
```

```
person list[۲۰]={{"Ali Ahmadi",۹,۷۵},{"Reza Amini",۱۰,۰}};
```

خب حالا ما متغیر هایی داریم که نام و نمره افراد رو تحت یک عنوان و یک متغیر در یک بسته بندی ذخیره می کنند اگر بخوایم محتویات این متغیر هارا جابجا کنیم در عوض

جابجایی تک تک فیلدها فقط استراکچرها را جابجا می‌کنیم برای مثال می‌خواهیم آرایه‌ی لیستی را که چند خط قبل تعریف کردیم عناصر صفرم و یکم را جابجا کنیم

```

person tmp;
tmp=list[0];
list[0]=list[1];
list[1]=tmp;

```

به همین‌راحتی اول یک متغیر کمکی گرفتیم و بعد عناصر نام‌برده را جابجا کردیم!!! برای آرگمان توابع هم دیگه نیازی نیست به تعداد فیلدها آرگمان تعریف کنیم فقط یک آرگمان از نوع استراکچر مورد نظر تعریف میکنیم تا اینجا توانستیم چند متغیر را از انواع مختلف را در یک بسته بندی قراردهیم و بدون اینکه نامی از محتویات بسته‌ها ببریم، بسته‌ها را جابجا کنیم و در یکدیگر کپی کنیم ام‌بالاخره همه این‌ها برای این است که بتوانیم محتویات بسته‌ها را بهتر استفاده کنیم پس مسلمه که نیاز به دسترسی به محتویات این بسته‌ها یا همان فیلدها داریم برای این موضوع از عملگر نقطه استفاده میکنیم

مثال:

```

p1.name
P1.grade
list[0].name;
list[index].grade;

```

عبارات فوق برای دسترسی به نام و نمره فرد در استراکچریست که قبلاً تعریف کردیم عبارات فوق دقیقاً از نوع داده‌ای فیلدهای نام‌برده هستند یعنی آرایه‌ی کاراکتر و ممیز شناور و دقیقاً مثل انواع نام‌برده با آنها رفتار می‌شود برای مثال دستورهای زیر کاملاً مجاز هستند مثل انواع داده‌ای معمولی!!!

```

cin>>p1.name; خواندن آرایه‌ی کاراکتر;
cin>>p1.grade; خواندن ممیز شناور;
cout<<p1.name; چاپ رشته (آرایه‌ی کاراکتر);
cout<<p1.grade; چاپ ممیز شناور;
cin>>list[0].name; خواندن آرایه‌ی کاراکتر برای نام نفر اول لیست;
cin>>list[index].grade; خواندن ممیز شناور برای نمره‌ی نفر اندیس لیست;

```

جلسه‌ی آینده به بررسی استراکچرها بعنوان آرگمان توابع و اشاره‌گرهای آنها می‌پردازیم بعد سعی میکنم را جمع sort, search صحبت کنم تا چیزهای که تا حالا خواندیم بخصوص استراکچرها یا همان ساختمان‌ها رو کار بردی یاد بگیریم