دانشگاه بین المللی امام خمینی قزوین

دانشکدهٔ فنی و مهندسی

مبانی کامپیوتر و برنامه سازی

فصل دوازدهم: ساختارها در C

نستوه طاهری جوان

nastooh@aut.ac.ir

ساختمان

- یک structure یک ساختمان داده برای نگهداری عناصر با نوع های مختلف است.

- نحوهٔ تعریف ساختمان:

```
struct    نام ساختمان {
  نوع       نام فیلد ۱;
  نوع       نام فیلد ۲;
....
  نوع       نام فیلد n ;
  }          نام متغیرها;
```

- نوشتن نام ساختمان و نام متغیرها اختیاری است.

nastooh@aut.ac.ir

ساختمان

- مثال: یک ساختمان برای اطلاعات دانشجویان

```
struct  student{
    char  name [20];
    char  family [20];
    float  mark;
    long int  id;
} std1, std2, std3;
```

این ساختمان سه فیلد برای نام، فامیل و نمره دارد.

سه متغیر از نوع این ساختمان تعریف شده است که هر متغیر، هر چهار فیلد تعریف شده را دارند.

nastooh@aut.ac.ir

ساختمان

- برای دسترسی به فیلدهای هر متغیر از عملگر **نقطه** استفاده می کنیم.

```
struct  student{
    char  name [20];
    char  family [20];
    float  mark;
} std1, std2, std3;

...

std2.mark = 12.5;
strcpy (std3.name, “ALI”);
scanf (“%f”,  &std1.mark);
gets (std3.family);
```


nastooh@aut.ac.ir

ساختمان

- میتوان متغیرهای از نوع ساختمان را بعد از تعریف ساختمان تعریف کرد.

```
struct  student{
    char  name [20];
    char  family [20];
    float  mark;
};

...

struct  student  std1, std2;
```


nastooh@aut.ac.ir

ساختمان

- حتی میتوان برای یک ساختمان، یک نوع دادهٔ جدید تعریف کرد.
  - برای تعریف نوع دادهٔ جدید از typedef استفاده می کنیم.

```
struct  student{
    char  name [20];
    char  family [20];
    float  mark;
};

typedef   struct student   Student;

...

Student  std1, std2;
```


nastooh@aut.ac.ir

ساختمان

- آرایه ای از ساختمان (توضیح با مثال)

```
struct  student{
    char  name [20];
    char  family [20];
    float  mark;
};

struct  student  std [20];


std [1]. mark = 13.6;
scanf (“%f”, &std[5].mark);
std[3].name[5] = ‘A’;
strcpy (std[0].family, “Taheri”);
```

حرف ششم از اسم دانشجوی چهارم را تغیر میدهد.

nastooh@aut.ac.ir

ساختمان

- ساختمان تو در تو (توضیح با مثال)

```
struct  date{
        int   year;
        int   month;
        int   day;
 };
struct  student {
            char  name [20] ;
            char  family [20];
            struct  date  b_d;
 };

struct   student  st[20];

st[2].b_d.year = 1375;
st[0].b_d.day = 25;
strcpy (st[5].name, “ALI”);
```

به نحوهٔ ارجاع به فیلدها دقت کنید!!

nastooh@aut.ac.ir

ساختمان

- اشاره گر به ساختمان (توضیح با مثال)

```
struct  student {
            char  name [20] ;
            char  family [20];
            long int  id;
 };
struct student  st,  *p;

p = &st;

(*p).id = 123456;
p->id = 123456;
```

به نحوهٔ ارجاع به فیلدها دقت کنید!!
برای این کار دو روش وجود دارد.

nastooh@aut.ac.ir

ساختمان

- ساختمان خود ارجاع

```
struct  student {
            char  name [20] ;
            char  family [20];
            long int  id;
            struct  student  *pst;
 };
```

فیلد چهارم این ساختمان، یک اشاره گر به یک عنصر از همین جنس است.

- این نوع ساختمان درآینده کاربردهای زیادی در برنامه نویسی برای شما پیدا خواهد کرد.
  - برای درک مثال های این ساختمان، مفهومِ "لیست پیوندی" را جستجو کنید.

nastooh@aut.ac.ir

ساختمان

- یادآوری چند نکته از scanf()

این نکات را هنگام دریافت فیلدهای ساختمان مد نظر قرار دهید.

- وقتی با تابع scanf() یک داده خوانده شود، کاراکتر اینترِ انتهای ورودی در بافر باقی خواهد ماند.
- اگر پس از scanf() بخواهیم یک رشته را با gets() بخوانیم، اینتر باقی مانده از scanf() را به عنوان ورودی gets() در نظر میگیرد و رشته را با تهی مقدار دهی می کند.
- با scanf() نمی توان رشته های حاوی فضای خالی را خواند.
- می توان اعداد را نیز به صورت رشته ای خواند و با توابع کتابخانه ای مناسب، به مقدار عددی تبدیل کرد.

nastooh@aut.ac.ir

ساختمان

- تمرین مهم:
  - دربارهٔ مفاهیم زیر تحقیق کنید:
    - مفهوم کلاس.
    - مفهوم برنامه نویسی شی گرا.

Union

- مفهوم اتحاد
  - اشغال حافظهٔ یکسان توسط فیلدها

```
union  student {
            char  name [10] ;
            long int  id;

 };
```

  - همهٔ فیلدها از یک نقطهٔ حافظه شروع می شوند.

در مورد اتحاد، در اینترنت جستجو کرده و بر روی نکات و مثالهای مطرح شده در کلاس، تامل کنید.

nastooh@aut.ac.ir

پایان

nastooh@aut.ac.ir