مبانی کامپیوتر و برنامه سازی

فصل دهم: ساختارها

Structure

- ساختار یک ظرف است که چیزهای مختلفی درون آن هستند
  - این چیزها می توانند از هر نوعی باشند.
- کاربرد اصلی ساختار، سازماندهی متغییرهای وابسته به هم تحت یک مجموعه خوب و بی نقص می باشد.

Example - Student Record

- Student Record:
  - Name — a string
  - HW Grades — an array of 3 doubles
  - Test Grades — an array of 2 doubles
  - Final Average — a double

Structure Members

- به هر کدام از چیزهای داخل ساختار، عضو گفته می شود.
- هر عضو دارای اسم، نوع و مقدار است.
- اسم هر عضو از قوانین نامگذاری متغییرها پیروی می کند.
- نوع می تواند هم از انواع پیش فرض زبان و هم از انواع تعریف شده توسط کاربر باشد.

Example Structure Definition

```
struct StudentRecord {
    char *name;          // student name
    double hw[3]; // homework grades
    double test[2];// test grades
    double ave;          // final average
};
```

Using a struct

- هر گاه که یک ساختار تعریف می کنید، نوع داده جدیدی را ایجاد می کنید.
- بعد از تعریف هر ساختار، می توان متغییرهای از آن نوع ایجاد کرد.

- StudentRecord stu;

Accessing Members

- با اعضای ساختار باید مثل متغییرهای معمولی برخورد نمود.
- برای دسترسی به هر عضو از "." استفاده می شود.

- cout << stu.name << endl;
- stu.hw[2] = 82.3;
- stu.ave = total/100;

Structure Assignment

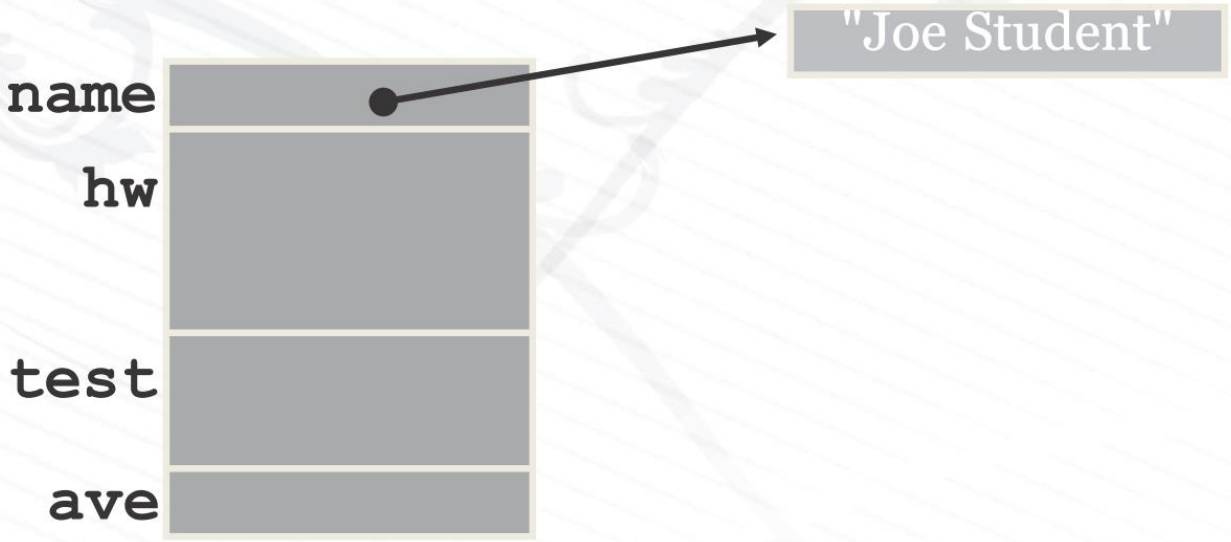
- با خود ساختار نیز می شود عین متغییرها برخورد نمود.

- StudentRecord s1,s2;
- s1.name = "Joe Student";
- …
- s2 = s1;

Copies the entire structure

Be Careful

- دقت کنید که هنگام کپی کردن ساختارها، اگر عضو ساختار به صورت اشاره گر باشد، خود اشاره گر کپی می شود نه مقداری که اشاره گر به آن اشاره می کند.



Probably not what you want

- StudentRecord s1,s2;
- s1.name = "Joe Student";
- …
- s2 = s1;
- s2.name = "Jane Doe";

- // now s1.name and s2.name are both
- // "Jane Doe"

Pointers to Structures

- معمولا ساختارها را بصورت اشاره گر استفاده می کنند.
- در این حالت برای دسترسی به اعضا باید از <- استفاده نمود.

- StudentRecord *sptr;
- …
- cout << "Name is" << sptr->name;
- cout << "Ave is " << sptr->ave;

Sample Function (won't work!)

```
void update_average( StudentRecord stu) {
    double tot=0;

    for (int i=0;i<3;i++)
        tot += stu.hw[i];
    for (int i=0;i<3;i++)
        tot += stu.test[i];
    stu.ave = tot/5;
}
```

This one works

```
void update_average( StudentRecord *stu) {
    double tot=0;

    for (int i=0;i<3;i++)
        tot += stu->hw[i];
    for (int i=0;i<3;i++)
        tot += stu->test[i];
    stu->ave = tot/5;
}
```

Or use a reference parameter

```
void update_average( StudentRecord &stu) {
    double tot=0;

    for (int i=0;i<3;i++)
        tot += stu.hw[i];
    for (int i=0;i<3;i++)
        tot += stu.test[i];
    stu.ave = tot/5;
}
```

Other stuff you can do with a struct

- حتی توابع می توانند جزء اعضای ساختار باشند. (اعضای تابعی)
- یک کلاس C++ خیلی شبیه ساختار است.
  - کلاسهای می توانند اعضای داده ای داشته باشند.
  - کلاسهای می توانند اعضای تابعی داشته باشند.
  - کلاسهای می توانند بعضی از اعضا را پنهان کنند.

Quick Example

```
struct StudentRecord {
  char *name;          // student name
  double hw[3];        // homework grades
  double test[2];      // test grades
  double ave;          // final average

  void print_ave() {
    cout << "Name: " << name << endl;
    cout << "Average: " << ave << endl;
  }
};
```

Using the member function

- doubleStudentRecord stu;
- … // set values in the structure
- stu.print_ave();