CSCI 112
Programming with C

Chapter

User-defined Structure Types

Program 2

Struct
Structured Collection of Data

- We want to categorize all the planets
- We want to know the name, diameter, how many moons, time to orbit and how long it takes to rotate
- We can do this by creating our own data type using the keyword “struct”

Struct

- **<u>2 ways to do it</u>**

```
struct Planet {
        char name[70];
        double diameter;
        int moons;
        double orbit_time,
                rotation_time;
};
// in a function
struct Planet p1, p2;
```

MY PREFERRED METHOD

```
typedef struct {
        char name[70];
        double diameter;
        int moons;
        double orbit_time,
                rotation_time;
} Planet;
// in a function
Planet p1, p2;
```

Struct

- Syntax

```
typedef struct {
        <type> <id>;
        <type> <id>;
        …
} <name of structure data type>;
```

Struct
Setting the element values

```
typedef struct {
        char name[70];
        double diameter;
        int moons;
        double orbit_time,
                rotation_time;
} Planet;
// in a function
Planet p1 = { “Jupiter”, 18000000.0, 16, 91000.5, 10.6};
```

Start Here Wednesday

- Monday's help session – mon_1019.c - %u
- Program 2 due 11/4 and Lab 5 due 11/9
  - I will have the help session at the beginning of class on 11/9 to help anyone with lab 5
  - Lab 6 will now be done on Monday 11/16
    - we will do review and help session that day.

Struct
Setting the element values

```
typedef struct {
        char name[70];
        double diameter;
        int moons;
        double orbit_time,
                rotation_time;
} Planet;
// in a function
Planet p1, p2;
p1.moons = 16;
strcpy(p1.name, “Jupiter”);
```

Struct Operations

```
typedef struct {
        char name[70];
        double diameter;
        int moons;
        double orbit_time,
                rotation_time;
} Planet;
// in a function
Planet p1, p2;
p1.moons = 16;
strcpy(p1.name, “Jupiter”);
p2 = p1;
```

Struct Arrays

```
typedef struct {
        char name[70];
        double diameter;
        int moons;
        double orbit_time,
                rotation_time;
} Planet;
// in a function
Planet planets[20];
int i;
for (i = 0; i < 20; i++) {
        strcpy(planets[i].name, “none”);
        planets[i].moons = 0;
        …
}
```

Struct
A Structure As A Pointer

```
typedef struct {
        char name[70];
        double diameter;
        int moons;
        double orbit_time,
                rotation_time;
} Planet;
// in main
Planet p1;
setPlanet(&p1);
..
void setPlanet(Planet *p) {
        (*p).moons = 0;
        strcpy((*p).name, “none”);
        …
}
```

DON’T USE THIS METHOD

Struct
A Structure As A Pointer

```
typedef struct {
        char name[70];
        double diameter;
        int moons;
        double orbit_time,
                rotation_time;
} Planet;
// in main
Planet p1;
setPlanet(&p1);
..
void setPlanet(Planet *p) {
        p->moons = 0;
        strcpy(p->name, “none”);
        …
}
```

switch over to ch10_notes.txt

Struct12.c Explained

- In main:
  - Student _t st_read; // Address is 59c0
  - st_read = StudentScan(); // st_read keeps 59c0
- In StudentScan():
  - Student_t studnt; // Address is 5860
  - // name: 5860 (first element in struct) – same address as studnt
  - // id: 5888 (5860+28Hex(40)) year, 588c (5888+4), year: 5890 (588c+4)
- Back in main:
  - Student_t st_upd; // Address is 5980
  - st_upd = student_update_credits(st_read);

local now to function

- In student_update_credits(Student_t st) { // st (input param) has address: 5900)
  - - st.credits += 15;
  - return st; // st is local and output (as st_read) – not the same as st_read which was input to this function
- Back in main:
  - // &st_read.name: 59c0 (first element in struct) – same address as st_read, &st_upd.name: 5980 (same as st_upd)
  - // st_read.name: 59c0 – same address as st_read, st_upd.name: 5980 (same as st_upd)
  - st_read.credits – 12 (as read in)
  - st_upd.credits – 27 (12 + 15)

```
typedef struct {
    char name[40];
    int id, credits;
    char year;
} Student_t;
```

Struct13.c Explained

- In main:
  - Student st_read; // Address is 59c0
  - st_read = StudentScan(); // st1 keeps 59c0
- In StudentScan():
  - Student_t studnt; // Address is 5860
  - // name: 5860 (first element in struct) – same address as studnt
  - // id: 5868 (5860+8) year, 586c (5888+4), year: 5870 (586c+4)
- Back in main:
  - Student_t st_upd; // Address is 5980
  - st_upd = student_update_credits(str_read);

local now to function

- In student_update_credits(Student_t st) { // st (input param) has address: 5900)
  - - st.credits += 15;
  - return st; // st is local and output (as str) – not the same as st_read which was input to this function
- Back in main:
  - // &st_read.name: 59c0 (first element in struct) – same address as st_read, &st_upd.name: 5980 (same as st_upd)
  - // st_read.name: 76b0, st_upd.name: 76b0
  - st_read.credits – 12 (as read in)
  - st_upd.credits – 27 (12 + 15)

```
typedef struct {
    char *name;
    int id, credits;
    char year;
} Student_t;
```