FUNCTIONS IN C

```
typedef struct {
   float looks;
   int smarts;
} Stats;
```

```
typedef struct {
   char name[20];
   Stats s;
} UPStudent;
```

```
int main(){
   UPStudent s1, *p;
   p = &s1;
   //assign 7.5 to looks using s1
   //assign 9 to smarts using p
}
```

:P

```c
typedef struct {
   float looks;
   int smarts;
} Stats;
```

```c
typedef struct {
   char name[20];
   Stats s;
} UPStudent;
```

```c
int main(){
   UPStudent s1, *p;
   p = &s1;
   s1.s.looks = 7.5;
   p->s.smarts = 9;
}
```

Objective

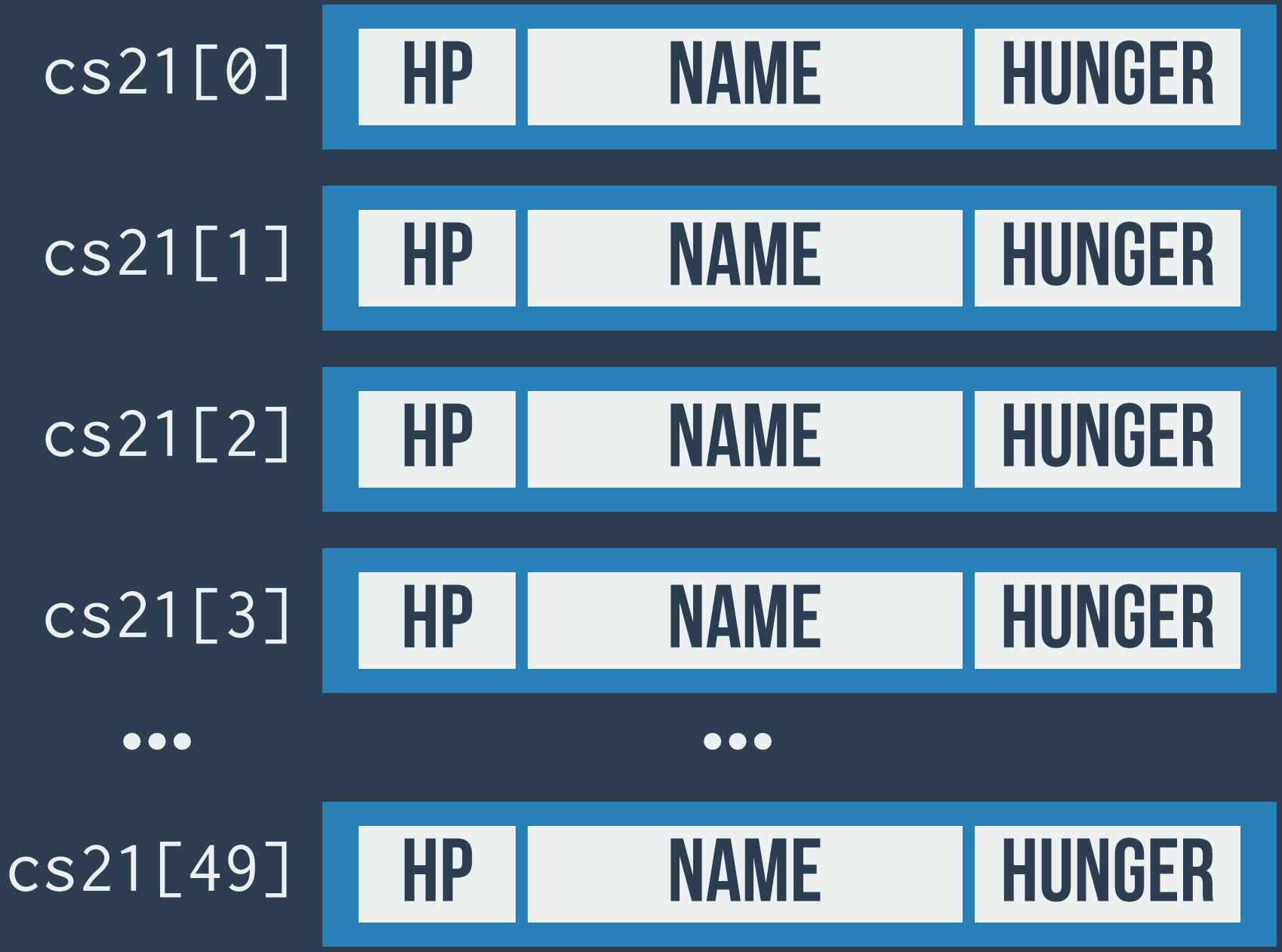
To declare, access, and pass arrays of structures

ARRAY OF STRUCTURES

A collection of structures of the same type.

```c
typedef struct {
   int hp;
   char name[20];
   float hunger;
} Student;
```

```
Student cs21[50];
```


cs21[0]
HP
NAME
HUNGER
cs21[1]
HP
NAME
HUNGER
cs21[2]
HP
NAME
HUNGER
cs21[3]
HP
NAME
HUNGER
cs21[49]
HP
NAME
HUNGER

ACCESSING ARRAY OF STRUCTURES

```
<var_name>[index].field
```

Where var_name is the structure variable identifier
and field is the field identifier.

```
//assign values to the first
//element
Student cs21[50];

cs21[0].hp = 55;
strcpy(cs21[0].name, “Simon”);
cs21[0].hunger = 90.5;
```

```
//accessing all elements via a loop
Student cs21[50];
   …
for(i=0;i<50;i++)
{
   scanf(“%d”, &cs21[i].hp);
   scanf(“%s”, cs21[i].name);
   scanf(“%f”, &cs21[i].hunger);
}
```

Via pointer arithmetic (*)

```
//accessing via pointer arithmetic
Student cs21[50], *p;
p = cs21;

//the first element
(*p).hp = 13;
strcpy((*p).name, “Kamina”);
printf(“%s”, (*p).hunger);
```

```
//accessing via pointer arithmetic
Student cs21[50], *p;
p = cs21;

//the second element
(*(p+1)).hp = 15;
strcpy((*(p+1)).name, “Yoko”);
printf(“%s”, (*(p+1)).hunger);
```

```
//the third element
(*(p+2)).hp = 15;
strcpy((*(p+2)).name, “Viral”);
printf(“%s”, (*(p+2)).hunger);
```

```
// * with loops
Student cs21[50];
   …
for(i=0;i<50;i++)
{
   scanf(“%d”, &(*(cs21+i)).hp);
   scanf(“%s”, (*(cs21+i)).name);
   scanf(“%f”, &(*(cs21+i)).hunger);
}
```

Via pointer arithmetic (->)

```
//accessing via *
Student cs21[50], *p;
p = cs21;

//the first element
(*p).hp = 13;
strcpy((*p).name, “Kamina”);
printf(“%s”, (*p).hunger);
```

```
//accessing via ->
Student cs21[50], *p;
p = cs21;

//the first element
p->hp = 13;
strcpy(p->name, “Kamina”);
printf(“%s”, p->hunger);
```

```
//accessing via *
Student cs21[50], *p;
p = cs21;

//the second element
(*(p+1)).hp = 15;
strcpy((*(p+1)).name, “Yoko”);
printf(“%s”, (*(p+1)).hunger);
```

```
//accessing via ->
Student cs21[50], *p;
p = cs21;

//the second element
(p+1)->hp = 15;
strcpy((p+1)->name, “Yoko”);
printf(“%s”, (p+1)->hunger);
```

```
// -> with loops
Student cs21[50];
   …
for(i=0;i<50;i++)
{
   scanf(“%d”, &(cs21+i)->hp);
   scanf(“%s”, (cs21+i)-> name);
   scanf(“%f”, &(cs21+i)-> hunger);
}
```

DYNAMIC STRUCTURES

a pointer + malloc()

```
//to create a DYNAMIC structure
Student *one;

one = (Student *) malloc(sizeof(Student));
```

```
//to create a DYNAMIC ARRAY of
//structures
Student *many;

many = (Student *)
          malloc(50 * sizeof(Student));
```

free() for deallocating

```
free(one);
free(many);
```

PASSING ARRAYS OF STRUCTURES

1

PASS THE ADDRESS OF THE FIRST ELEMENT

Done if the function needs to access all the array elements.

```
int main()
{
   Student cs21[50];
   getInput(cs21);
}
```

```
void getInput(Student *s)
{
   int i;
   for(i=0;i<50;i++)
   {
      scanf("%d", &s[i].hp);
      scanf("%s", s[i].name);
      scanf("%f", &s[i].hunger);
   }
}
```

```
void getInput(Student *s)
{
   int i;
   for(i=0;i<50;i++)
   {
      scanf("%d", &(*(s+i)).hp);
      scanf("%s", (*(s+i)).name);
      scanf("%f", &(*(s+i)).hunger);
   }
}
```

```
void getInput(Student *s)
{
   int i;
   for(i=0;i<50;i++)
   {
      scanf("%d", &(s+i)->hp);
      scanf("%s", (s+i)->name);
      scanf("%f", &(s+i)->hunger);
   }
}
```

2

PASS INDIVIDUAL ELEMENTS

Each array element is treated as a single structure.

```
int main()
{
   Student cs21[50];
   getInput(cs21[0]); //1st element
}
```

```
void getInput(Student s)
{
   int i;
   for(i=0;i<50;i++)
   {
      scanf("%d", &s.hp);
      scanf("%s", s.name);
      scanf("%f", &s.hunger);
   }
}
```