Intermediate Programming

— Structures —

Waseda Univ.

Today's topics

- Tutorial of structures in C
  - Usage of structures
  - How to initialize a structure variable and access members
  - Pointer to a structure
  - Return a structure variable in a function
  - Array of structure

Arrays and Structures

【Arrays】

- An array is a set of variables under the same name.
- All elements in the array must have the same data type.

Scores									Data type
91	78	69	84	86	92	77	81	90	int

【Structures】

- Structures store many different data of different types under the same name. These data are defined by the user.

Student	Student1	Student2	Student3	...	Data type
Name	Aki	Yu	Kou		char
Score	90	94	72		int
Deviation	64.2	67.1	56.3		double

Define a structure

```
#include <stdio.h>

struct student{
  char name[20];
  int  math;
  int  phys;
};

int main(void){
  struct student a, b;
  struct student c={"Frank", 90};
  :
  return 0;
}
```

- Structure tag is the name of the entire type of the structure.
- Members are the variables within the structure.
- You need to declare structure variables by **struct** statement.
- Structure variables can be initialized by the same way as the array.

Accessing structure members

- In order to assign a value to a structure member, the dot (.) operator (or member access operator) links each member name with the structure variable.

```
#include <stdio.h>
#include <string.h>      /*necessary for using the strcpy function*/

struct student{
  char name[20];
  int  math;
  int  phys;
};

int main(void){
  struct student a, b;

  strcpy(a.name,"Frank");
  a.math = 90;
  a.phys = 83;
  ...
```

Assignment of structure variables

- The assignment operator (=) copies all structure members to another structure variable if their structure tags are same.
- Such an assignment is different from arrays.

```
#include <stdio.h>
#include <string.h>

struct student{
  char name[20];
  int math;
  int phys;
};
int main(void){

  struct student a,b;

  strcpy(a.name, "Frank");
  a.math = 90;
  a.phys = 83;

  printf("     Name:%s\n",a.name);
  printf("     Math:%d\n",a.math);
  printf("  Physics:%d\n",a.phys);

  b=a;
  printf("     Name:%s\n",b.name);
  printf("     Math:%d\n",b.math);
  printf("  Physics:%d\n",b.phys);

  return 0;

}
```

【Output】

```
     Name:Frank
     Math:90
  Physics:83
     Name:Frank
     Math:90
  Physics:83
```

Pointer to structure

- We can have a pointer to a single structure variable.
- Such a pointer points to the address of first member of the structure.
- The "&" operator returns the address of the structure variable.

```
struct student{
  char name[20];
  int  math;
  int  phys;
};

int main(void){
  struct student a, *pa;
  pa = &a;
  ...
```

Pointer	Allocated to
pa	a.name
	a.math
	a.phys

The layout of the computer memory will be above.

Pointer to structure

- We can have a pointer to a single structure variable.
- Such a pointer points to the address of first member of the structure.
- The "&" operator returns the address of the structure variable.

```
struct student{
  char name[20];
  int  math;
  int  phys;
};

int main(void){
  struct student a, *pa;
  pa = &a;
  ...
```

Pointer	Allocated to
pa	a.name
	a.math
	a.phys

The layout of the computer memory will be above.

Accessing structure members with pointer

- We use arrow (->) to access structure members when we have a pointer of structure. Usage is name_of_pointer->member_name.
- This means (*name_of_pointer).member_name as well.

```
#include <stdio.h>
#include <string.h>

struct student{
  char name[20];
  int math;
  int phys;
};

int main(void){

  struct student a,*pa;

  strcpy(a.name, "Frank");
  a.math = 90;
  a.phys = 83;

  printf("     Name:%s\n",a.name);
  printf("     Math:%d\n",a.math);
  printf("  Physics:%d\n",a.phys);

  pa=&a;
  strcpy(pa->name, "Thomas");
  pa->phys = 92;

  printf("     Name:%s\n",pa->name);
  printf("     Math:%d\n",pa->math);
  printf("  Physics:%d\n",pa->phys);

  return 0;

}
```

【Output】

```
     Name:Frank
     Math:90
  Physics:83
     Name:Thomas
     Math:90
  Physics:92
```

Accessing structure members with pointer

- We use arrow (->) to access structure members when we have a pointer of structure. Usage is name_of_pointer->member_name.
- This means (*name_of_pointer).member_name as well.

```
#include <stdio.h>
#include <string.h>

struct student{
  char name[20];
  int math;
  int phys;
};

int main(void){

  struct student a,*pa;

  strcpy(a.name, "Frank");
  a.math = 90;
  a.phys = 83;

  printf("     Name:%s\n",a.name);
  printf("     Math:%d\n",a.math);
  printf("  Physics:%d\n",a.phys);

  pa=&a;
  strcpy(pa->name, "Thomas");
  pa->phys = 92;

  printf("     Name:%s\n",pa->name);
  printf("     Math:%d\n",pa->math);
  printf("  Physics:%d\n",pa->phys);

  return 0;

}
```

【Output】

```
     Name:Frank
     Math:90
  Physics:83
     Name:Thomas
     Math:90
  Physics:92
```

Example:

- Consider the output of the following program:

```
#include <stdio.h>
#include <string.h>

struct student{
  char name[20];
  int math;
  int phys;
};
int main(void){
  struct student a,*pa;
  strcpy(a.name, "Frank");
  a.math = 90;
  a.phys = 83;

  printf("     Name:%s\n",a.name);
  printf("     Math:%d\n",a.math);
  printf("  Physics:%d\n",a.phys);

  pa=&a;
  strcpy(pa->name, "Thomas");
  pa->phys = 92;

  printf("     Name:%s\n",pa->name);
  printf("     Math:%d\n",pa->math);
  printf("  Physics:%d\n",pa->phys);
  printf("     Name:%s\n",a.name);
  printf("     Math:%d\n",a.math);
  printf("  Physics:%d\n",a.phys);

  return 0;
}
```

【Output】

```
   Name:Frank
   Math:90
Physics:83
   Name:Thomas
   Math:90
Physics:92
   Name:Thomas
   Math:90
Physics:92
```

Example:

- Consider the output of the following program:

```
#include <stdio.h>
#include <string.h>

struct student{
  char name[20];
  int math;
  int phys;
};
int main(void){
  struct student a,*pa;
  strcpy(a.name, "Frank");
  a.math = 90;
  a.phys = 83;

  printf("     Name:%s\n",a.name);
  printf("     Math:%d\n",a.math);
  printf("  Physics:%d\n",a.phys);

  pa=&a;
  strcpy(pa->name, "Thomas");
  pa->phys = 92;

  printf("     Name:%s\n",pa->name);
  printf("     Math:%d\n",pa->math);
  printf("  Physics:%d\n",pa->phys);
  printf("     Name:%s\n",a.name);
  printf("     Math:%d\n",a.math);
  printf("  Physics:%d\n",a.phys);

  return 0;
}
```

【Output】

```
     Name:Frank
     Math:90
  Physics:83
     Name:Thomas
     Math:90
  Physics:92
     Name:Thomas
     Math:90
  Physics:92
```

- Sample program using the pointer as function arguments.

```
#include <stdio.h>

struct student{ /*Declaring structure*/

  char name[20];
  int  math;
  int  phys;
  double ave;

};

void Average(struct student *std){
/*Pointer as function arguments*/

  int sum;
  sum = std->math + std->phys;
  std->ave = (double) sum/2;

}
```

```
int main(void){

  struct student S1 = {"Frank",90,83};

  Average(&S1);

  printf("Name =%s\n",S1.name);
  printf("Math =%d\n",S1.math);
  printf("Phys =%d\n",S1.phys);
  printf("Ave  =%.2f\n",S1.ave);

  return 0;
}
```

【Output】

```
Name =Frank
Math =90
Phys =83
Ave  =86.50
```

- Sample program using the pointer as function arguments.

```
#include <stdio.h>

struct student{ /*Declaring structure*/

  char name[20];
  int  math;
  int  phys;
  double ave;

};

void Average(struct student *std){
/*Pointer as function arguments*/

  int sum;
  sum = std->math + std->phys;
  std->ave = (double) sum/2;

}
```

```
int main(void){

  struct student S1 = {"Frank",90,83};

  Average(&S1);

  printf("Name =%s\n",S1.name);
  printf("Math =%d\n",S1.math);
  printf("Phys =%d\n",S1.phys);
  printf("Ave  =%.2f\n",S1.ave);

  return 0;
}
```

【Output】

```
Name =Frank
Math =90
Phys =83
Ave  =86.50
```

Structures as the return value of function

- Structures can be used as the return value of functions
- The "Average" function in previous page is rewritten as follows. Note that the parameter of the function is also the structure variable.

```
struct student Average(struct student temp){
  temp.ave = (double) (temp.math + temp.phys)/2;
  return temp;
}

int main(void){
  ...
  S1=Average(S1);
  ...
```

Array of structure

- We can also declare an array of structure variables. Each element of the array represents a structure variable including all members of the structure.

```
struct structure tag name_of_array[#_of_elements];
```

- Declaration and usage are the same as the array.

```
struct student Std[20];
  ...

for(i=0;i<N;i++){
   Std[i].ave = (double)(Std[i].math+Std[i].phys)/2;
}
```

Summary

- Tutorial of structures in C
  - Usage of structures
  - How to initialize a structure variable and access members
  - Pointers to the structure variable
  - Return a structure variable
  - Array of structure