Pointers and Structures

Here we will learn to use pointers with structures in C Programming Language.

Creating a structure

Lets start by creating a structure variable student as shown below.

```
// student structure
struct student {
  char id[15];
  char firstname[64];
  char lastname[64];
  float points;
};
```

Now we will create a student structure variable std. For this we will write the following code.

```
// student structure variable
struct student std;
```

Access the members of a structure

We know that to access a member of a structure we use the . operator.

In the following example we are accessing the members of the student structure.

printf("ID: %s\n", std.id);

printf("First Name: %s\n", std.firstname);

printf("Last Name: %s\n", std.lastname);

printf("Points: %f\n", std.points);

Creating pointer for structure

Following is the syntax to create a pointer for a structure.

struct tagName *ptrName;

So, to create a pointer for the student structure we will write the following code.

struct student *ptr;

Assigning structure variable to pointer

We use the following syntax to assign a structure variable address to a pointer.

ptrName = &structVarName;

In the following example we are assigning the address of the structure variable std to the structure pointer variable ptr. So, ptr is pointing at std.

ptr = &std;

Accessing the members of a structure via pointer

We use the arrow operator also known as member selection operator -> to access the members of a structure via pointer variable.

Following is the syntax for accessing members of a structure via pointer.

ptrName->member

In the following example we are accessing the firstname member of the student structure via pointer variable ptr.

printf("First Name: %s", ptr->firstname);

Complete code

```
#include <stdio.h>
int main(void) {
  // student structure
  struct student {
    char id[15];
    char firstname[64];
    char lastname[64];
    float points;
  };

  // student structure variable
  struct student std;

  // student structure pointer variable
  struct student *ptr = NULL;
  // assign std to ptr
  ptr = &std;
```

```
  // get student detail from user
  printf("Enter ID: ");
  scanf("%s", ptr->id);
  printf("Enter first name: ");
  scanf("%s", ptr->firstname);
  printf("Enter last name: ");
  scanf("%s", ptr->lastname);
  printf("Enter Points: ");
  scanf("%f", &ptr->points);

  // display result via std variable
  printf("\nResult via std\n");
  printf("ID: %s\n", std.id);
  printf("First Name: %s\n", std.firstname);
  printf("Last Name: %s\n", std.lastname);
  printf("Points: %f\n", std.points);

  // display result via ptr variable
  printf("\nResult via ptr\n");
  printf("ID: %s\n", ptr->id);
  printf("First Name: %s\n", ptr->firstname);
  printf("Last Name: %s\n", ptr->lastname);
  printf("Points: %f\n", ptr->points);

  return 0;
}
```

Output:

Enter ID: s01

Enter first name: Yusuf

Enter last name: Shakeel

Enter Points: 8.44

Result via std

ID: s01

First Name: Yusuf

Last Name: Shakeel

Points: 8.440000

Result via ptr

ID: s01

First Name: Yusuf

Last Name: Shakeel

Points: 8.440000

Pointers and Array of Structures

Create an array of structure variable

In the following example we are considering the student structure that we created in the previous tutorial and we are creating an array of student structure variable std of size 3 to hold details of three students.

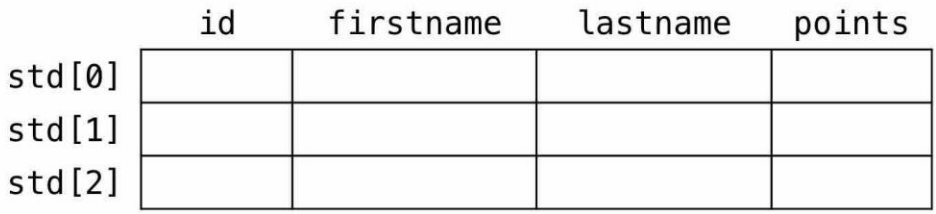
```
// student structure
struct student {
  char id[15];
  char firstname[64];
  char lastname[64];
  float points;
};

// student structure variable
struct student std[3];
```

We can represent the std array variable as following.

```
struct student {
  char id[15];
  char firstname[64];
  char lastname[64];
  float points;
};
```

```
struct student std[3];
```



Create pointer variable for structure

Now we will create a pointer variable that will hold the starting address of the student structure variable std.

```
// student structure pointer variable
struct student *ptr = NULL;
```

```
// assign std to ptr
ptr = std;
```

Note! std is an array variable and the name of the array variable points at the memory location so, we are assigning it to the structure pointer variable ptr.

Accessing each element of the structure array variable via pointer

For this we will first set the pointer variable ptr to point at the starting memory location of std variable. For this we write ptr = std;.

Then, we can increment the pointer variable using increment operator ptr++ to make the pointer point at the next element of the structure array variable i.e., from str[0] to str[1].

We will loop three times as there are three students. So, we will increment pointer variable twice. First increment will move pointer ptr from std[0] to std[1] and the second increment will move pointer ptr from std[1] to std[2].

To reset the pointer variable ptr to point at the starting memory location of structure variable std we write ptr = std;.

Complete code

```
#include <stdio.h>

int main(void) {

  // student structure
  struct student {
    char id[15];
    char firstname[64];
    char lastname[64];
    float points;
  };

  // student structure variable
  struct student std[3];

  // student structure pointer variable
  struct student *ptr = NULL;
```

```
// other variables
int i;

// assign std to ptr
ptr = std;

// get detail for user
for (i = 0; i < 3; i++) {
  printf("Enter detail of student #%d\n", (i + 1));
  printf("Enter ID: ");
  scanf("%s", ptr->id);
  printf("Enter first name: ");
  scanf("%s", ptr->firstname);
  printf("Enter last name: ");
  scanf("%s", ptr->lastname);
  printf("Enter Points: ");
  scanf("%f", &ptr->points);

  // update pointer to point at next element
  // of the array std
  ptr++;
}

// reset pointer back to the starting
// address of std array
ptr = std;
```

```
  for (i = 0; i < 3; i++) {
    printf("\nDetail of student #%d\n", (i + 1));

    // display result via std variable
    printf("\nResult via std\n");
    printf("ID: %s\n", std[i].id);
    printf("First Name: %s\n", std[i].firstname);
    printf("Last Name: %s\n", std[i].lastname);
    printf("Points: %f\n", std[i].points);

    // display result via ptr variable
    printf("\nResult via ptr\n");
    printf("ID: %s\n", ptr->id);
    printf("First Name: %s\n", ptr->firstname);
    printf("Last Name: %s\n", ptr->lastname);
    printf("Points: %f\n", ptr->points);

    // update pointer to point at next element
    // of the array std
    ptr++;
  }

  return 0;
}
```

Output:

Enter detail of student #1

Enter ID: s01

Enter first name: Yusuf

Enter last name: Shakeel

Enter Points: 8

Enter detail of student #2

Enter ID: s02

Enter first name: Jane

Enter last name: Doe

Enter Points: 9

Enter detail of student #3

Enter ID: s03

Enter first name: John

Enter last name: Doe

Enter Points: 6

Detail of student #1

Result via std

ID: s01

First Name: Yusuf

Last Name: Shakeel

Points: 8.000000

Result via ptr

ID: s01

First Name: Yusuf

Last Name: Shakeel

Points: 8.000000

Detail of student #2

Result via std

ID: s02

First Name: Jane

Last Name: Doe

Points: 9.000000

Result via ptr

ID: s02

First Name: Jane

Last Name: Doe

Points: 9.000000

Detail of student #3

Result via std

ID: s03

First Name: John

Last Name: Doe

Points: 6.000000

Result via ptr

ID: s03

First Name: John

Last Name: Doe

Points: 6.000000

Passing structure pointer to function

Create a structure

In the following example are are creating a student structure.

```
// student structure
struct student {
  char id[15];
  char firstname[64];
  char lastname[64];
  float points;
};
```

Function declaration to accept structure pointer

Following is the syntax of the function declaration that accepts structure pointer.

returnType functionName(struct tagName *);

returnType is the return type of the function functionName. If the function is not returning anything then set it to void. The function takes structure tagName pointer.

In the following example we are creating two function declarations that takes address of student structure.

void getDetail(struct student *);

void displayDetail(struct student *);

Creating an array of structure variable

We will now create an array of student structure variable by writing the following code.

// student structure variable

struct student std[3];

So, we have created an array of student structure variable of size 3 to store the detail of three students.

We can represent the std array variable as follows.

```
struct student {
  char id[15];
  char firstname[64];
  char lastname[64];
  float points;
};
```

```
struct student std[3];
```

	id	firstname	lastname	points
std[0]				
std[1]				
std[2]				

Now we will write the complete code that will help us to get students data and then display them.

Complete code

```
#include <stdio.h>

// student structure
struct student {
  char id[15];
  char firstname[64];
  char lastname[64];
  float points;
};

// function declaration
void getDetail(struct student *);
```

```
void displayDetail(struct student *);

int main(void) {

  // student structure variable
  struct student std[3];

  // get student detail
  getDetail(std);

  // display student detail
  displayDetail(std);

  return 0;
}

// function definition


void getDetail(struct student *ptr) {

  int i;

  for (i = 0; i < 3; i++) {
    printf("Enter detail of student #%d\n", (i + 1));
    printf("Enter ID: ");
    scanf("%s", ptr->id);
```

```
    printf("Enter first name: ");
    scanf("%s", ptr->firstname);
    printf("Enter last name: ");
    scanf("%s", ptr->lastname);
    printf("Enter Points: ");
    scanf("%f", &ptr->points);

    // update pointer to point at next element
    // of the array std
    ptr++;
  }

}

void displayDetail(struct student *ptr) {

  int i;

  for (i = 0; i < 3; i++) {
    printf("\nDetail of student #%d\n", (i + 1));

    // display result via ptr variable
    printf("\nResult via ptr\n");
    printf("ID: %s\n", ptr->id);
    printf("First Name: %s\n", ptr->firstname);
    printf("Last Name: %s\n", ptr->lastname);
    printf("Points: %f\n", ptr->points);
```

```
    // update pointer to point at next element
    // of the array std
    ptr++;
  }

}
```

Output:

```
Enter detail of student #1
Enter ID: s01
Enter first name: Yusuf
Enter last name: Shakeel
Enter Points: 8
Enter detail of student #2
Enter ID: s02
Enter first name: John
Enter last name: Doe
Enter Points: 7
Enter detail of student #3
Enter ID: s03
Enter first name: Jane
Enter last name: Doe
Enter Points: 9

Detail of student #1
```

Result via ptr

ID: s01

First Name: Yusuf

Last Name: Shakeel

Points: 8.000000

Detail of student #2

Result via ptr

ID: s02

First Name: John

Last Name: Doe

Points: 7.000000

Detail of student #3

Result via ptr

ID: s03

First Name: Jane

Last Name: Doe