C Programming

Structures

Dr. Asif Uddin Khan

Structure in C

- Structure is a user-defined data type that enables us to store the collection of different data types.
- Each element of a structure is called a member.
- **struct** keyword is used to define the structure.

How to define structures?

- Before you can create structure variables, you need to define its data type. To define a structure, the struct keyword is used.

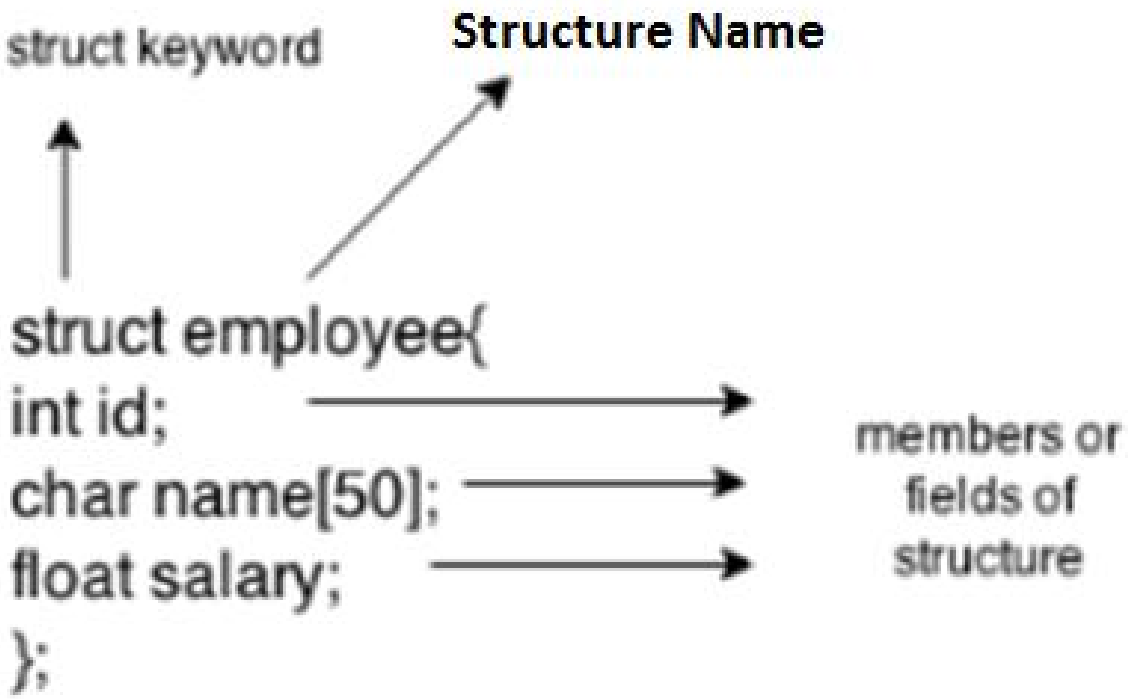
Syntax

```
struct structure_name
{
    data_type member1;
    data_type member2;
    .
    .
    data_type memeberN;
};
```

Example

```
struct employee
{   int id;
    char name[20];
    float salary;
};
```

Structure Elements



Declaring structure variable

- By struct keyword within main() function
- By declaring a variable at the time of defining the structure.

1st way:

```
struct employee
{   int id;
    char name[50];
    float salary;
};
```

Now write given code inside the main() function.

```
struct employee e1, e2;
```

2nd way:

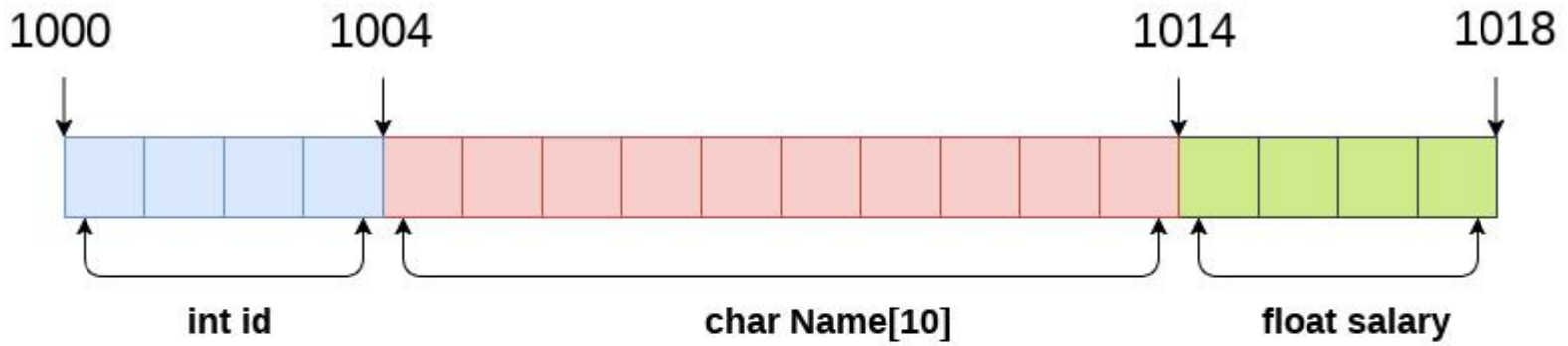
```
struct employee
{   int id;
    char name[50];
    float salary;
}e1,e2;
```

Which approach is good?

•If number of variables are not fixed, use the 1st approach. It provides you the flexibility to declare the structure variable many times.

•If no. of variables are fixed, use 2nd approach. It saves your code to declare a variable in main() function.

Structure in memory




```
struct Employee
{
    int id;
    char Name[10];
    float salary;
} emp;
```

sizeof (emp) = 4 + 10 + 4 = 18 bytes

where;
- sizeof (int) = 4 byte
- sizeof (char) = 1 byte
- sizeof (float) = 4 byte

Accessing members of the structure

Two ways to access structure members:

- By . (member or dot operator)
- By -> (structure pointer operator)

Example: Using of dot operator :

emp.id

Where
- emp is the structure variable.
- id is the structure member

Example: Using pointer operator :

emp->id

C Structure example

```c
#include<stdio.h>
#include <string.h>
struct employee
{    int id;
     char name[50];
}emp;  //declaring e1 variable for structure
int main( )
{
   //store first employee information
   emp.id=101;
   scanf("%[^\n]",emp.name);
   //printing first employee information
   printf( "employee 1 id : %d\n", emp.id);
   printf( "employee 1 name : %s\n", emp.name);
return 0;
}
```

Example-2

```c
#include<stdio.h>
#include <string.h>
struct employee
{   int id;
    char name[50];
    float salary;
}e1,e2;  //declaring e1 and e2 variables for structure
int main( )
{
   //store first employee information
   printf("First employee\n:");
   printf("Enter id of e1:");
   scanf("%f",&e1.id);
   printf("Enter name of e1:");
   scanf("%s",e1.name);
   printf("Enter Salary of e1:");
   scanf("%f",&e1.salary);
```

```c
//store second employee information
  printf("Second employee\n:");
  printf("Enter id of e2:");
  scanf("%f",&e2.id);
  printf("Enter name of e2:");
  scanf("%s",e2.name);
  printf("Enter Salary of e2:");
  scanf("%f",&e2.salary);

  //printing first employee information
  printf( "employee 1 id : %d\n", e1.id);
  printf( "employee 1 name : %s\n", e1.name);
  printf( "employee 1 salary : %f\n", e1.salary);

  //printing second employee information
  printf( "employee 2 id : %d\n", e2.id);
  printf( "employee 2 name : %s\n", e2.name);
  printf( "employee 2 salary : %f\n", e2.salary);
  return 0;
}
```

Structures and Pointers

C Pointers to struct

Here's how you can create pointers to structs.

```
struct name {
    member1;
    member2;
    .
    .
};

int main()
{
    struct name *ptr, Harry;
}
```

Here, `ptr` is a pointer to `struct`.

Example

Example: Access members using Pointer

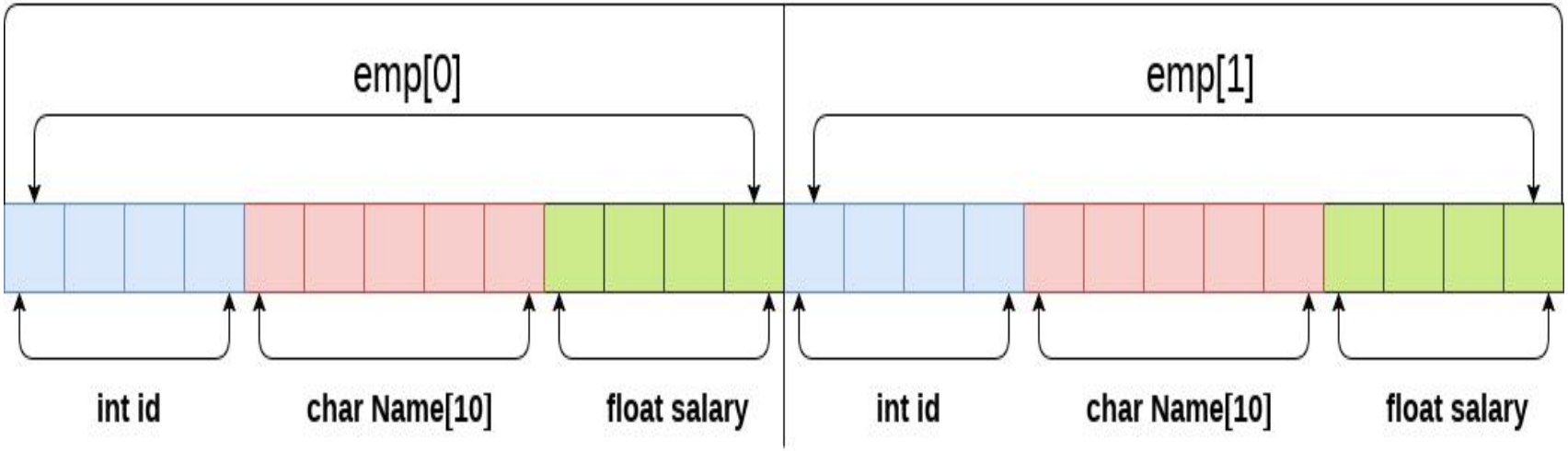
To access members of a structure using pointers, we use the `->` operator.

```c
#include <stdio.h>
struct person
{
   int age;
   float weight;
};

int main()
{
    struct person *personPtr, person1;
    personPtr = &person1;

    printf("Enter age: ");
    scanf("%d", &personPtr->age);

    printf("Enter weight: ");
    scanf("%f", &personPtr->weight);

    printf("Displaying:\n");
    printf("Age: %d\n", personPtr->age);
    printf("weight: %f", personPtr->weight);

    return 0;
}
```

Array of Structures in C

- Collection of multiple structures variables where each variable contains information about different entities.
- Used to store information about multiple entities

Array of structures




```
struct employee
{
    int id;
    char name[5];
    float salary;
};
struct employee emp[2];
```

sizeof (emp) = 4 + 5 + 4 = 13 bytes

sizeof (emp[2]) = 26 bytes

Example

```c
#include<stdio.h>
#include <string.h>
struct student{
int rollno;
char name[10];
};
int main(){
int i;
struct student st[5];
printf("Enter Records of 5 students");
for(i=0;i<5;i++){
printf("\nEnter Rollno:");
scanf("%d",&st[i].rollno);
printf("\nEnter Name:");
scanf("%s",&st[i].name);
}
printf("\nStudent Information List:");
for(i=0;i<5;i++){
printf("\nRollno:%d, Name:%s",st[i].rollno,st[i].name);
}
   return 0;
}
```

Example: Add two distances

```
// Program to add two distances (feet-inch)
#include <stdio.h>
struct Distance
{
    int feet;
    float inch;
} dist1, dist2, sum;

int main()
{
    printf("1st distance\n");
    printf("Enter feet: ");
    scanf("%d", &dist1.feet);
    printf("Enter inch: ");
    scanf("%f", &dist1.inch);
    printf("2nd distance\n");
    printf("Enter feet: ");
    scanf("%d", &dist2.feet);
    printf("Enter inch: ");
    scanf("%f", &dist2.inch);
    // adding feet
    sum.feet = dist1.feet + dist2.feet;
    // adding inches
    sum.inch = dist1.inch + dist2.inch;
    // changing to feet if inch is greater than 12
    while (sum.inch >= 12)
    {
        ++sum.feet;
        sum.inch = sum.inch - 12;
    }
    printf("Sum of distances = %d\'-%.1f\"", sum.feet, sum.inch);
    return 0;
}
```

Keyword typedef

- We use the typedef keyword to create an alias name for data types.
- It is commonly used with structures to simplify the syntax of declaring variables.

```
This code
struct Distance{
    int feet;
    float inch;
};
int main() {
    struct Distance d1, d2;
}
```

```
is equivalent to
typedef struct Distance{
    int feet;
    float inch;
} distances;

int main() {
    distances d1, d2;
}
```

Nested Structures

- You can create structures within a structure in C programming. For example,

```
struct complex
{
 int imag;
 float real;
};

struct number
{
    struct complex comp;
    int integers;
} num1, num2;
```

Suppose, you want to
set imag of num2 variable to 11.
Here's how you can do it:

```
num2.comp.imag = 11;
```

Program for nested structure

```c
//program for nested structure
#include<stdio.h>
struct address
{
    char city[20];
    int pin;
    char phone[14];
};
struct employee
{
    char name[20];
    struct address add;
};
void main ()
{
    struct employee emp;
    printf("Enter employee information?\n");
    scanf("%s %s %d %s",emp.name,emp.add.city, &emp.add.pin, emp.add.phone);
    printf("Printing the employee information....\n");
    printf("name: %s\nCity: %s\nPincode: %d\nPhone: %s",emp.name,emp.add.city,emp.add.pin,emp.add.phone);
}
```

Passing structure as function argument

```c
// passing structure as function argument
#include <stdio.h>
struct student {
    char name[50];
    int age;
};
// function prototype
void display(struct student s);
int main() {
    struct student s1;
    printf("Enter name: ");
    // read string input from the user until \n is entered
    // \n is discarded
    scanf("%[^\n]", s1.name);
    printf("Enter age: ");
    scanf("%d", &s1.age);
    display(s1); // passing struct as an argument
    return 0;
}
void display(struct student s) {
    printf("\nDisplaying information\n");
    printf("Name: %s", s.name);
    printf("\nAge: %d", s.age);
}
```

typedef example

```c
// typedef example
#include <stdio.h>
typedef struct student {
    char name[50];
    int age;
}st;
// function prototype
void display(st s);
int main() {
  st s1;
    printf("Enter name: ");
    // read string input from the user until \n is entered
    // \n is discarded
    scanf("%[^\n]", s1.name);
    printf("Enter age: ");
    scanf("%d", &s1.age);
    display(s1); // passing struct as an argument
}
void display(st s) {
    printf("\nDisplaying information\n");
    printf("Name: %s", s.name);
    printf("\nAge: %d", s.age);
}
```

Dynamic memory allocation and structure

```c
// dynamic memory allocation and structure
#include <stdio.h>
#include<stdlib.h>
typedef struct student {
   char name[50];
   int age;
}st;
// function prototype
void display(st* s , int n);
int main() {
    int n,i;
  st s1,*ptr;
  printf("Enter size:");
  scanf("%d",&n);
  ptr=(st*)malloc(n*sizeof(st));

  for(i=0;i<n;i++){
   printf("Enter name of student-%d: ",i+1);
   scanf("%s", (ptr+i)->name);
   printf("Enter age of student-%d: ",i+1);
   scanf("%d", &(ptr+i)->age);
  }
   display(ptr,n); // passing struct as an argument
   free(ptr);
}
void display(st* s, int n) {
    int j;
    for(j=0;j<n;j++){
   printf("\nDisplaying information of student-%d\n",j+1);
   printf("Name: %s", (s+j)->name);
   printf("\nAge: %d", (s+j)->age);
    }
}
```

Output

```
Enter size:2
Enter name of student-1: aa
Enter age of student-1: 23
Enter name of student-2: bb
Enter age of student-2: 24

Displaying information of student-1
Name: aa
Age: 23
Displaying information of student-2
Name: bb
Age: 24
```

References

1. C programming by E Balaguruswami
2. Programming C by Y. kanitkar
3. Programming C by Denis Ritchie
4. NPTEL Lecture note of Dr. Partha Pratim Das, Department of Computer Science and Engineering, Indian Institute of Technology, Kharagpur.
5. https://docs.oracle.com/cd/E18752_01/html/817-6223/chp-typeopexpr-2.html
6. https://data-flair.training/blogs/escape-sequence-in-c/
7. Internet source

References

1. NPTEL Lecture note of Dr. Partha Pratim Das, Department of Computer Science and Engineering, Indian Institute of Technology, Kharagpur.
2. https://docs.oracle.com/cd/E18752_01/html/817-6223/chp-typeopexpr-2.html
3. https://data-flair.training/blogs/escape-sequence-in-c/
4. Internet source