Structure in C

Structure in c language is a user-defined datatype that allows you to hold a different type of elements.

Each element of a structure is called a member.

It works as a template in C++ and class in Java. You can have a different type of elements in it.

It is widely used to store student information, employee information, product information, book information etc.

Defining structure

The struct keyword is used to define the structure. Let's see the syntax to define structure in c.

1. struct structure\_name

2. {

3. data\_type member1;

4. data\_type member2;

5. .

6. .

7. data\_type memeberN;

8. };

Let's see the example to define a structure for the employee in c.

1. struct employee

2. { int id;

3. char name[50];

4. float salary;

5. };

Here, struct is the keyword, the employee is the tag name of structure; id, name and salary are the members or fields of the structure. Let's understand it by the diagram given below:

Declaring structure variable

We can declare a variable for the structure so that we can access the member of the structure easily. There are two ways to declare a structure variable:

1. By struct keyword within main() function

2. By declaring variable at the time of defining structure.

1st way:

Let's see the example to declare a structure variable by struct keyword. It should be declared within the main function.

1. struct employee

2. { int id;

3. char name[50];

4. float salary;

5. };

Now write given code inside the main() function.

1. struct employee e1, e2;

2nd way:

Let's see another way to declare a variable at the time of defining structure.

1. struct employee

2. { int id;

3. char name[50];

4. float salary;

5. }e1,e2;

Which approach is good

But if no. of the variable are not fixed, use the 1st approach. It provides you with the flexibility to declare the structure variable many times.

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

Accessing members of structure

There are two ways to access structure members:

1. By . (member or dot operator)

2. By -> (structure pointer operator)

Let's see the code to access the id member of p1 variable by . (member) operator.

1. p1.id

C Structure example

Let's see a simple example of structure in C language.

1. #include <stdio.h>

2. #include <string.h>

3. struct employee

4. { int id;

5. char name[50];

6. }e1; //declaring e1 variable for structure

7. int main( )

8. {

9. //store first employee information

10. e1.id=101;

11. strcpy(e1.name, "Sonoo Jaiswal");//copying string into char array

12. //printing first employee information

13. printf( "employee 1 id : %d\n", e1.id);

14. printf( "employee 1 name : %s\n", e1.name);

15. return 0;

16. }

Output:

employee 1 id : 101

employee 1 name : Sonoo Jaiswal

Let's see another example of structure in C language to store many employees information.

1. #include <stdio.h>

2. #include <string.h>

3. struct employee

4. { int id;

5. char name[50];

6. float salary;

7. }e1,e2; //declaring e1 and e2 variables for structure

8. int main( )

9. {

10. //store first employee information

11. e1.id=101;

12. strcpy(e1.name, "Sonoo Jaiswal");//copying string into char array

13. e1.salary=56000;

14.

15. //store second employee information

16. e2.id=102;

17. strcpy(e2.name, "James Bond");

18. e2.salary=126000;

19.

20. //printing first employee information

21. printf( "employee 1 id : %d\n", e1.id);

22. printf( "employee 1 name : %s\n", e1.name);

23. printf( "employee 1 salary : %f\n", e1.salary);

24.

25. //printing second employee information

26. printf( "employee 2 id : %d\n", e2.id);

27. printf( "employee 2 name : %s\n", e2.name);

28. printf( "employee 2 salary : %f\n", e2.salary);

29.

30. return 0;

31. }

Output:

employee 1 id : 101

employee 1 name : Sonoo Jaiswal

employee 1 salary : 56000.000000

employee 2 id : 102

employee 2 name : James Bond

employee 2 salary : 126000.00000

Array of Structures in C

There can be array of structures in C programming to store many information of different data types. The array of structures is also known as collection of structures.

Let's see an example of structure with array that stores information of 5 students and prints it.

1. #include<stdio.h>

2. #include<conio.h>

3. #include<string.h>

4. struct student{

5. int rollno;

6. char name[10];

7. };

8. void main(){

9. int i;

10. struct student st[5];

11. clrscr();

12. printf("Enter Records of 5 students");

13.

14. for(i=0;i<5;i++){

15. printf("\nEnter Rollno:");

16. scanf("%d",&st[i].rollno);

17. printf("\nEnter Name:");

18. scanf("%s",&st[i].name);

19. }

20.

21. printf("\nStudent Information List:");

22. for(i=0;i<5;i++){

23. printf("\nRollno:%d, Name:%s",st[i].rollno,st[i].name);

24. }

25.

26. getch();

27. }

Output:

Enter Records of 5 students

Enter Rollno:1

Enter Name:Sonoo

Enter Rollno:2

Enter Name:Ratan

Enter Rollno:3

Enter Name:Vimal

Enter Rollno:4

Enter Name:James

Enter Rollno:5

Enter Name:Sarfraz

Student Information List:

Rollno:1, Name:Sonoo

Rollno:2, Name:Ratan

Rollno:3, Name:Vimal

Rollno:4, Name:James

Rollno:5, Name:Sarfraz

Nested Structure in C

Nested structure in c language can have another structure as a member. There are two ways to define nested structure in c language:

1. By separate structure

2. By Embedded structure

1) Separate structure

We can create 2 structures, but dependent structure should be used inside the main structure as a member. Let's see the code of nested structure.

1. struct Date

2. {

3. int dd;

4. int mm;

5. int yyyy;

6. };

7. struct Employee

8. {

9. int id;

10. char name[20];

11. struct Date doj;

12. }emp1;

As you can see, doj (date of joining) is the variable of type Date. Here doj is used as a member in Employee structure. In this way, we can use Date structure in many structures.

2) Embedded structure

We can define structure within the structure also. It requires less code than previous way. But it can't be used in many structures.

1. struct Employee

2. {

3. int id;

4. char name[20];

5. struct Date

6. {

7. int dd;

8. int mm;

9. int yyyy;

10. }doj;

11. }emp1;

Accessing Nested Structure

We can access the member of nested structure by Outer\_Structure.Nested\_Structure.member as given below:

1. e1.doj.dd

2. e1.doj.mm

3. e1.doj.yyyy

C Nested Structure example

Let's see a simple example of nested structure in C language.

1. #include <stdio.h>

2. #include <string.h>

3. struct Employee

4. {

5. int id;

6. char name[20];

7. struct Date

8. {

9. int dd;

10. int mm;

11. int yyyy;

12. }doj;

13. }e1;

14. int main( )

15. {

16. //storing employee information

17. e1.id=101;

18. strcpy(e1.name, "Sonoo Jaiswal");//copying string into char array

19. e1.doj.dd=10;

20. e1.doj.mm=11;

21. e1.doj.yyyy=2014;

22.

23. //printing first employee information

24. printf( "employee id : %d\n", e1.id);

25. printf( "employee name : %s\n", e1.name);

26. printf( "employee date of joining (dd/mm/yyyy) : %d/%d/%d\n", e1.doj.dd,e1.doj.mm,e1.doj.yyyy);

27. return 0;

28. }

Output:

employee id : 101

employee name : Sonoo Jaiswal

employee date of joining (dd/mm/yyyy) : 10/11/2014

C Union

Like structure, Union in c language is a user-defined datatype that is used to hold a different type of elements.

But it doesn't occupy the sum of all members size. It occupies the memory of largest member only. It shares the memory of the largest member.

Advantage of the union over a structure

It occupies less memory because it occupies the memory of largest member only.

Disadvantage of union over structure

It can store data in one member only.

Defining union

The union keyword is used to define union. Let's see the syntax to define union in c.

1. union union\_name

2. {

3. data\_type member1;

4. data\_type member2;

5. .

6. .

7. data\_type memeberN;

8. };

Let's see the example to define union for employee in c.

1. union employee

2. { int id;

3. char name[50];

4. float salary;

5. };

C Union example

Let's see a simple example of union in C language.

1. #include <stdio.h>

2. #include <string.h>

3. union employee

4. { int id;

5. char name[50];

6. }e1; //declaring e1 variable for union

7. int main( )

8. {

9. //store first employee information

10. e1.id=101;

11. strcpy(e1.name, "Sonoo Jaiswal");//copying string into char array

12. //printing first employee information

13. printf( "employee 1 id : %d\n", e1.id);

14. printf( "employee 1 name : %s\n", e1.name);

15. return 0;

16. }

Output:

employee 1 id : 1869508435

employee 1 name : Sonoo Jaiswal

As you can see, id gets garbage value because name has large memory size. So only name will have actual value.

C Strings

String in C language is an array of characters that is terminated by \0 (null character).

There are two ways to declare string in c language.

1. By char array

2. By string literal

Let's see the example of declaring string by char array in C language.

1. char ch[10]={'j', 'a', 'v', 'a', 't', 'p', 'o', 'i', 'n', 't', '\0'};

As you know well, the array index starts from 0, so it will be represented as in the figure given below.

While declaring string, size is not mandatory. So you can write the above code as given below:

1. char ch[]={'j', 'a', 'v', 'a', 't', 'p', 'o', 'i', 'n', 't', '\0'};

You can also define string by string literal in C language. For example:

1. char ch[]="javatpoint";

In such case, '\0' will be appended at the end of string by the compiler.

Difference between char array and string literal

The only difference is that string literal cannot be changed whereas string declared by char array can be changed.

String Example in C

Let's see a simple example to declare and print string. The '%s' is used to print string in c language.

1. #include <stdio.h>

2. void main ()

3. {

4. char ch[11]={'j', 'a', 'v', 'a', 't', 'p', 'o', 'i', 'n', 't', '\0'};

5. char ch2[11]="javatpoint";

6.

7. printf("Char Array Value is: %s\n", ch);

8. printf("String Literal Value is: %s\n", ch2);

9. }

Output:

Char Array Value is: javatpoint

String Literal Value is: javatpoint

C gets() and puts() functions

The gets() function reads string from user and puts() function prints the string. Both functions are defined in <stdio.h> header file.

Let's see a simple program to read and write string using gets() and puts() functions.

1. #include<stdio.h>

2. #include<conio.h>

3. void main(){

4. char name[50];

5. clrscr();

6. printf("Enter your name: ");

7. gets(name); //reads string from user

8. printf("Your name is: ");

9. puts(name); //displays string

10. getch();

11. }

Output:

Enter your name: Sonoo Jaiswal

Your name is: Sonoo Jaiswal

C String Functions

There are many important string functions defined in "string.h" library.

No. Function Description

1) strlen(string\_name)

returns the length of string name.

2) strcpy(destination, source)

copies the contents of the source string to the destination string.

3) strcat(first\_string, second\_string)

concats or joins first string with the second string. The result of the string is stored in the first string.

4) strcmp(first\_string, second\_string)

compares the first string with the second string. If both strings are same, it returns 0.

5) strrev(string)

returns reverse a string.

6) strlwr(string)

returns string characters in lowercase.

7) strupr(string)

returns string characters in uppercase.

C String Length: strlen() function

The strlen() function returns the length of the given string. It doesn't count null character '\0'.

1. #include <stdio.h>

2. void main()

3. {

4. char ch[20]={'j', 'a', 'v', 'a', 't', 'p', 'o', 'i', 'n', 't', '\0'};

5. printf("Length of string is: %d",strlen(ch));

6. }

Output:

Length of string is: 10

C Copy String: strcpy()

The strcpy(destination, source) function copies the source string in destination.

1. #include <stdio.h>

2. void main()

3. {

4. char ch[20]={'j', 'a', 'v', 'a', 't', 'p', 'o', 'i', 'n', 't', '\0'};

5. char ch2[20];

6. strcpy(ch2,ch);

7. printf("Value of second string is: %s",ch2);

8. }

Output:

Value of second string is: javatpoint

C String Concatenation: strcat()

The strcat(first\_string, second\_string) function concatenates two strings and result is returned to first\_string.

1. #include <stdio.h>

2. void main()

3. {

4. char ch[10]={'h', 'e', 'l', 'l', 'o', '\0'};

5. char ch2[10]={'c', '\0'};

6. strcat(ch,ch2);

7. printf("Value of first string is: %s",ch);

8. }

Output:

Value of first string is: helloc

C Compare String: strcmp()

The strcmp(first\_string, second\_string) function compares two string and returns 0 if both strings are equal.

Here, we are using gets() function which reads string from the console.

1. #include <stdio.h>

2. void main()

3. {

4. char str1[20],str2[20];

5. printf("Enter 1st string: ");

6. gets(str1);//reads string from console

7. printf("Enter 2nd string: ");

8. gets(str2);

9. if(strcmp(str1,str2)==0)

10. printf("Strings are equal");

11. else

12. printf("Strings are not equal");

13. }

Output:

Enter 1st string: hello

Enter 2nd string: hello

Strings are equal

C Reverse String: strrev()

The strrev(string) function returns reverse of the given string. Let's see a simple example of strrev() function.

1. #include<stdio.h>

2. #include<conio.h>

3. void main(){

4. char str[20];

5. clrscr();

6. printf("Enter string: ");

7. gets(str);//reads string from console

8. printf("String is: %s",str);

9. printf("\nReverse String is: %s",strrev(str));

10. getch();

11. }

Output:

Enter string: javatpoint

String is: javatpoint

Reverse String is: tnioptavaj

C String Lowercase: strlwr()

The strlwr(string) function returns string characters in lowercase. Let's see a simple example of strlwr() function.

1. #include<stdio.h>

2. #include<conio.h>

3. void main(){

4. char str[20];

5. clrscr();

6. printf("Enter string: ");

7. gets(str);//reads string from console

8. printf("String is: %s",str);

9. printf("\nLower String is: %s",strlwr(str));

10. getch();

11. }

Output:

Enter string: JAVATpoint

String is: JAVATpoint

Lower String is: javatpoint

C String Uppercase: strupr()

The strupr(string) function returns string characters in uppercase. Let's see a simple example of strupr() function.

1. #include<stdio.h>

2. #include<conio.h>

3. void main(){

4. char str[20];

5. clrscr();

6. printf("Enter string: ");

7. gets(str);//reads string from console

8. printf("String is: %s",str);

9. printf("\nUpper String is: %s",strupr(str));

10. getch();

11. }

Output:

Enter string: javatpoint

String is: javatpoint

Upper String is: JAVATPOINT

C String strstr()

The strstr() function returns pointer to the first occurrence of the matched string in the given string. It is used to return substring from first match till the last character.

Syntax:

1. char \*strstr(const char \*string, const char \*match)

String strstr() parameters

string: It represents the full string from where substring will be searched.

match: It represents the substring to be searched in the full string.

String strstr() example

1. #include<stdio.h>

2. #include<conio.h>

3. #include<string.h>

4. void main(){

5. char str[100]="this is javatpoint with c and java";

6. char \*sub;

7. clrscr();

8. sub=strstr(str,"java");

9. printf("\nSubstring is: %s",sub);

10. getch();

11. }

Output:

javatpoint with c and java