C Programming

Class- BCA IInd Semester

Dr. Dharm Raj Singh

Assistant Professor, (HOD)

Department of Computer Application

Jagatpur P. G. College, Varanasi

Affiliated to Mahatma Gandhi Kashi Vidhyapith Varanasi

Email- dharmrajsingh67@yahoo.com

Outline

unit 1 : String

- Some of the String Functions in string.h
- Example of strcpy() function
- Example of strlen() function
- Example of strcat() function
- Example of strcmp() function
- Write a program to reverse a string without using strrev() function
- Write a program to check given string is palindrome or not
- Write a program check given string is palindrome or not without using string functions
- Write a program to convert string lower case to upper case without using string functions
- Write a program to c concatenate two Strings without using library function
- Write a program to search the occurrence of character in string

Union

A **union** is a derived data type available in C like structure and it also contain member of different data type and allow us to store them in the same memory location. We can define a union with multiple members, but only one member can contain a value at a time. Italso provides an efficient way of using the same memory location for multiple-purpose. The syntax of union are as follows-

```
union number
{
  int x;
  float y;
  char str[15];
};
```

So the memory size occupied by the number union will be 15 byte but if we take a structure of same type it take 21 byte of memory.

Example

```
#include<stdio.h>
#include<string.h>
 typedef struct employee
{
   char name[50];
   int salary;
}emp;

int main( )
{
   emp e1;
   printf("\nEmployee name:\t");
   scanf("%s", e1.name);
   printf("\nEnter Employee salary: \t");
   scanf("%d", &e1.salary);
   printf("\nstudent name is %s", e1.name);
   printf("\nroll is %d", e1.salary);
   return 0;
}
```

Example

```
#include <string.h>
 typedef struct student
{
  int id;
  char name[20];
  float percentage;
} status;

int main()
{
  status record;
  record.id=1;
  strcpy(record.name, "Raju");
  record.percentage = 86.5;
  printf(" Id is: %d \n", record.id);
  printf(" Name is: %s \n", record.name);
  printf(" Percentage is: %f \n", record.percentage);
  return 0;
}
```

Example

```
#include <stdio.h>
#include <string.h>
 struct student_str                          //normal structure declaration
{               char name[30];
                int age;
};
typedef struct {                            //structur declaration with typedef
                char name[30];
                int age;
}emp_str;
int main()
{               struct student_str std;
                        emp_str emp;
                strcpy(std.name, "Amit Shukla");     //assign values to std
                std.age = 21;
                strcpy(emp.name, "Abhishek Jain");       //assign values to emp
                emp.age = 27;
                printf("Student detail:\n");
                printf("Name: %s\n",std.name);
                printf("Age: %d\n",std.age);
                printf("Employee detail:\n");
                printf("Name: %s\n",emp.name);
                printf("Age: %d\n",emp.age);
                        return 0;
}
```

Initializing Unions

The difference between a structure and a union is that in case of a union, the fields share the same memory space, so new data replaces any existing data.

```
union Data
{
int i;
float f;
char str[20];
};
int main( )
{
union Data d;
data.i = 10;
printf( "d.i : %d\n", d.i);
data.f = 220.5;
printf( "d.f : %f\n", d.f);
strcpy( d.str, "C Programming");
printf( "d.str : %s\n", d.str);
return 0;
}
```

Initializing Unions

```
typedef struct POINT1
{
int x, y;
};
typedef union POINT2
{
int x;
int y;
};
int main()
{
POINT1 P1 = {2,3};      /*POINT2 P2 ={4,5}; Illegal in case of unions*/
POINT2 P2;
P2.x = 4;
P2.y = 5;
printf("\n The coordinates of P1 are %d and %d", P1.x, P1.y);
printf("\n The coordinates of P2 are %d and %d", P2.x, P2.y);
return 0;
}
```

Output

The coordinates of P1 are 2 and 3

The coordinates of P2 are 5 and 5

Difference between Structure & Union

Structure	Union
In structure each member get separate space in memory. struct Student { char[15] name; int rollno; int age; }s1; Therefore total memory required for structure is equal to the sum of the size of all the members that is 19 byte.	But in union, the total memory space is equal to the member with largest size and all other members of union share the same memory space. This is the main difference between structure and union. union student { char[15] name; int rollno; int age; }s1; Therefore total memory required for structure union is 15bytes.
We can access any member in any sequence.	We can access only that variable whose value is recently used.

References

- Kanetkar, Yashavant P. "Let Us C Fifth Edition." (2017).
- Kernighan, Brian W., and Dennis M. Ritchie. *The C programming language*. 2006.
- Ritchie, Dennis M., Brian W. Kernighan, and Michael E. Leask. The C programming language. Englewood Cliffs: Prentice Hall, 1988.
- McGraw-Hill, Herbert Schildt Tata. "The Complete Reference C fourth Edition”." (2005).
- Griffiths, David, and Dawn Griffiths. Head First C: A Brain-Friendly Guide. " O'Reilly Media, Inc.", 2012.
- Programming in C-Balguruswamy
- Structured programming approach using C-Forouzah & Ceilberg Thomson learning Publication

Declaration

The content is exclusively meant for academic purpose and for enhancing aching and learning. Any other use for economic/commercial purpose is strictly ohibited. The users of the content shall not distribute, disseminate or share it th anyone else and its use is restricted to advancement of individual owledge. The information provided in this e-content is authentic and best as r knowledge".

Dr. Dharm Raj Singh
Assistant Professor, (HOD)
Department of Computer Application
JagatpurP. G. College, Varanasi

Thanks