DSA LAB PROGRAMS-Structures-Strings-pointers

By Sindhu Sankati AP19110010477 CSE-G

PROGRAM 25

Title: Implementation of structure

Objective:

Write a C Program using structure for entering details of the five students as name, admission number, Date of birth, department and display all the details.

Explanation:

Inputting and outputs of structure having some objects. In c we access the structure objects by ‘ .’ for example if we have structure student, which contains objects name, rollno, let stu be the variable intializing the structure then we can access the objects by, stu.name, stu.rollno. We do the same with any number of objects.

Pseudo Code:

>Entering details

```
1. SET student s[5]
2. FOR i = 1 till 5 DO
   {
3.     input s objects data
   }
4. END
```

>displaying details

```
1. SET student s[5]
2. FOR i = 1 till 5 DO
   {
3.     PRINT s objects data
   }
4. END
```

Code:

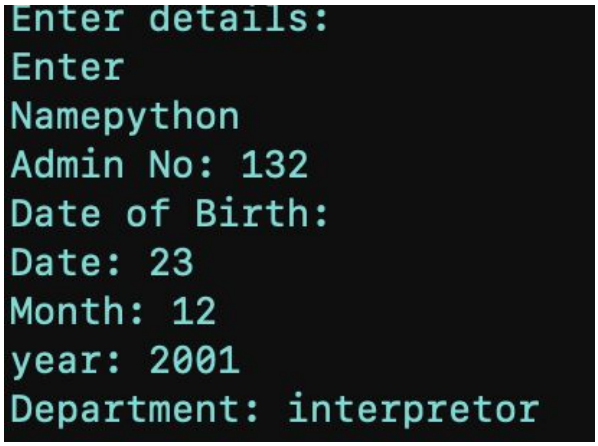
```c
#include <stdio.h>

//structure for date
typedef struct Date{
    int month;
    int day;
    int year;
```

```c
}date;
//structure for student
typedef struct Student{
     char name[30];
     int adminNo;
     date DOB;
     char dept[30];
}student;

int main()
{
     student s[5];
     int i;
     //inputtting the details of 5 students
     printf("Enter details: \n");
     for(i=0;i<5;i++){
          printf("Enter\n");
          printf("Name");
          scanf("%s",s[i].name);

          printf("Admin No: ");
          scanf("%d",&s[i].adminNo);

          printf("Date of Birth:\n");
          printf("Date: ");
          scanf("%d",&s[i].DOB.day);
          printf("Month: ");
          scanf("%d",&s[i].DOB.month);
          printf("year: ");
          scanf("%d",&s[i].DOB.year);

          printf("Department: ");
          scanf("%s",s[i].dept);
     }

     //printing details of five students
     printf("\n Details: \n");

     for(i=0;i<5;i++){
          printf("Name: %s\n",s[i].name );
          printf("Admin No: %d\n",s[i].adminNo );
```

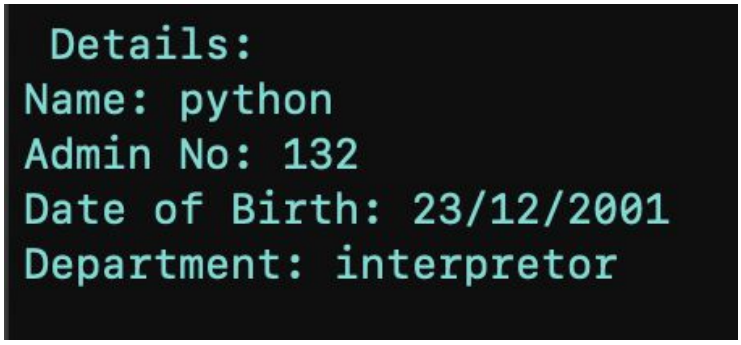
```
            printf("Date of Birth:
    %d/%d/%d\n",s[i].DOB.day,s[i].DOB.month,s[i].DOB.year);
            printf("Department: %s\n\n",s[i].dept);
        }
        return 0;
    }
```

Outputs:

sample input



sample output



Conclusion:

This code is giving expected result. The time complexity of enterning and displaying is constant. If instead of five students we need to enter and display n students, the timecomplexity would be O(n).

PROGRAM 26

Title: String length

Objective:

Write a C program to find length of string using pointers.

Explanation:

In C progrmass, strings end we a null character, represented by '\0'. THus to find the string length we can simply count the characters till '\0' from 0

Pseudo Code:

>Sum N natural numbers

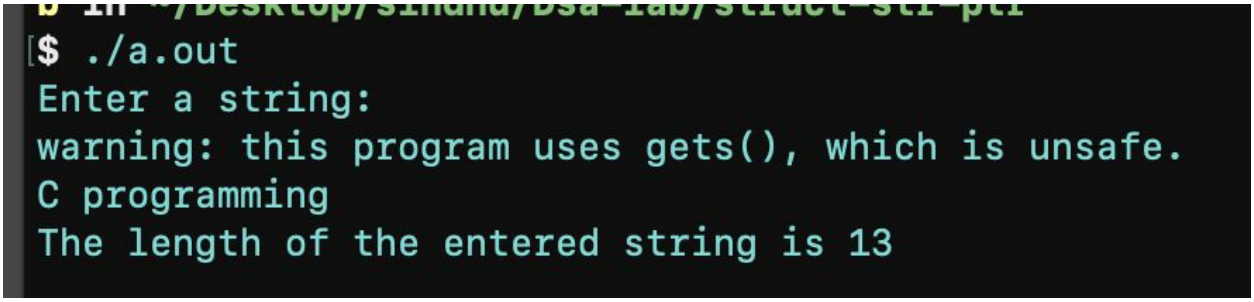
```
1. READ str[]
2. SET *ptr=str, count =0
3. WHILE *ptr != '\0'
   {
4.     count++,ptr++
   }
5. RETURN count
6. END
```

Code:

```c
#include <stdio.h>

//calculates the length of the string and return the length
int string_ln(char* p){
	int count = 0;
	//runs the loop till the end of the list
	while(*p != '\0'){
		count++;
		p++;
	}
	return count;
}
int main(int argc, char const *argv[])
{
	char str[40];
	//inputting string
	printf("Enter a string:\n");
	gets(str);
	//printing the length
	printf("The length of the entered string is
%d\n",string_ln(str));
```

```
        return 0;
    }
```

Outputs:



Conclusion:

There are 13 characters in “C programming”, therfore it gives ecpected result. The time complexity is O(n). WHere n is the length of the string.

PROGRAM 27

Title: String copy

Objective:

Write a C program to copy one string to another using pointers.

Explanation:

To copy a string to another we can just intialize every character of the string to the another, till end of the string which is represented by '\0'.

Pseudo Code:

>Sum N natural numbers

```
1. SET *dest, *src
2. while *src != '\0
3.     *dest++ = &src++
4. *dest = '\0'
5. END
```

Code:

```c
#include <stdio.h>
void copystr(char *dest,char *src)
{
    while(*src!='\0')
        *dest++=*src++;
    *dest='\0';
}
int main(int argc, char const *argv[])
{
     char str1[30],str2[30];
     //inputting string
     printf("Enter a string:\n");
     gets(str1);
     copystr(str2,str1);
     printf("String 1:\n");
     printf("%s\n",str1 );
     printf("string 2:\n");
     printf("%s\n",str2 );
     return 0;

}
```

Outputs:

Conclusion:

The function copies the string to another string, and the only exception is that of space, if desitated string is smaller the source string then the while string doesnot get copied.The time complexity is O(n).

PROGRAM 28

Title: String compare

Objective:

Write a C program to compare two strings using pointers.

Explanation:

For comparing string we travese throught the sting simulateniously if we bith string reach '\0' at same time it would mean they are of equal length. If one reached earler than other it would mean they are of unequal lenghts.

Pseudo Code:

>Sum N natural numbers

```
1. READ tsr1, str2
2. while(*str1 == *str)
   {
3.     IF str1 or str2 reeached end of string THEN BREAK
4.     str1++, str2++
5. }
   IF str1 ==str2 == '\0' RETURN 1
6. ELSE RETURN 0
7. END
```

Code:

```c
#include <stdio.h>
int strcomp(char *str1,char *str2)
{
    while(*str1 == *str2){
     if ( *str1 == '\0' || *str2 == '\0' )
         break;
      str1++;
      str2++;
    }
    if( *str1 == '\0' && *str2 == '\0' )
      return 1;
    return 0;

}
int main(int argc, char const *argv[])
{
```

```
        char str1[30],str2[30];
        //inputting string
        printf("Enter a string:\n");
        gets(str1);
        printf("Enter a string:\n");
        gets(str2);
        if(strcomp(str1,str2) == 1) printf("Stringers are equal\n");
        else printf("strings are not equal\n");
        return 0;
    }
```

Outputs:

```
Enter a string:
warning: this program uses gets(), which is unsafe.
House
Enter a string:
Rise
strings are not equal
```

Conclusion:

Since house is of 5 charaters and rise is of 4 characters, it print not equal. If there is house and asdfg it would print equal. The time complexity is O(n), where n is the length of the shorter string.

PROGRAM 29

Title: String reverse

Objective:

Write a C program to find the reverse of a string recursively and non-recursively.

Explanation:

We can reverse string by swapping the elments of first half with the last half.

Pseudo Code:

>Recursively

```
/*Assuming that n is the length of the string*/
1. SET beg = 0, end = n-1
2. RECREV(str,beg,end):
   {
3.    if beg>=end RETURN
4.    swap(str+beg,str,end);
5.    RECREV(str,beg++,end--)
   }
6. END
```

>Non-Recursively

```
/*assuming that n is the length of the string and str is the string to
be reversed*/
1. SET beg =0, end = n-1
2. FOR i form 0 to n/2{
3.     swap(str+beg,str+end)
4.     beg++, end--
   }
5. END
```

Code:

```c
#include <stdio.h>
void recRev(char *x, int beg, int end)
{
   char temp;

   if (beg >= end)
      return;

   temp = *(x+beg);
```

```
        *(x+beg) = *(x+end);
        *(x+end)   = temp;

        recRev(x, ++beg, --end);
    }
    int string_ln(char* p){
         int count = 0;
         //runs the loop till the end of the list
         while(*p != '\0'){
              count++;
              p++;
         }
         return count;
    }
    void rev(char *str)
    {
        int len, i;
        char *beg, *end, temp;

        len = string_ln(str);

        beg = str;
        end = str;

        for (i = 0;i < (len - 1 ) ; i++ )
            end++;
        for ( i = 0 ; i < len/2 ; i++ )
        {
            temp = *end;
            *end = *beg;
            *beg = temp;

            beg++;
            end--;
        }
    }
    int main(int argc, char const *argv[])
    {
         char str1[30];
         //inputting string
         printf("Enter a string:\n");
         gets(str1);
```

```
        printf("reversing string using recursion\n");
        recRev(str1,0,string_ln(str1)-1);
        printf("%s\n",str1);
        printf("Reversing the reversed string using iteration\n");
        rev(str1);
        printf("%s\n",str1);
        return 0;
    }
```

Outputs:

```
b in ~/Desktop/sindhu/Dsa-lab/struct-str-ptr
$ ./a.out
Enter a string:
warning: this program uses gets(), which is unsafe.
Dolphin
reversing string using recursion
nihploD
Reversing the reversed string using iteration
Dolphin
```

Conclusion:

Since the inputted string is “Dolphin” the reversed string found is “nihploD”, and the reverse of the is again “Dolphin”. Thus bith function are reversing the string and giving expected results. The time complexity for reversing string is O(n). BUt since we dont know the lenght of the string we need to find the lenght of the string explicitly which add another O(n) to complexity.