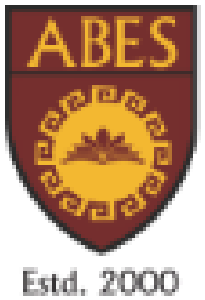

ABES
Engineering College
College Code-032

Data Structure Training

String

Department of CSE, CS & IT
ABES Engineering College, Ghaziabad

Data Structure Training

(String)

String-

- A string in C language is a character sequence stored in a character array and is terminated with null character.
- char str[10];

  Here str is a string of size 10 that can be hold up to 9 characters, one place is left for null character.

- Initialization of string can be done as follows

char str[10]="hello";

- Character after '\0' are garbage

Data Structure Training

(String)

All character array may not be strings

Let us consider two memory allocations

char str1[ ] = “hello”;

Sizeof (str1)=6

str1 is a string and is null terminated.

char str2[ ] = { ‘h’ , ‘e’ , ‘l’ , ‘l’ , ‘o’ };

Sizeof(str2)=5

str2 is a just character array and no extra null character is appended at the end.

Data Structure Training

(String)

Passing String to a function

In passing strings to a function, passing length of string is not required as null character marks end of string.

```
1   #include <stdio.h>
2   int mystrlen(const char *s);
3   int main(void)
4   {
5           char str[]="hello";
6           size_t c=strlen(str);
7           printf("%d",c);
8           return 0;
9   }
10  int mystrlen(const char *s)
11  {
12          int c=0;
13          while(*s != '\0')
14          {
15                  c++;
16                  s++;
17          }
18          return c;
19  }
```

Data Structure Training

(String)

String length v/s sizeof String-

- Strings are null terminated character arrays.
- A special format specifier %s is dedicated for reading/writing strings.
- A character array is used to hold a string.
- There are two length associated with a character array, size of memory allocated to the array and length of string stored in that memory.

char str[10]="hello";

strlen(str)=5
sizeof(str)=10

Data Structure Training

(String)

```c
#include <stdio.h>
int main()
{
        char str[10]="hello";
        printf("\n%d :: %d",sizeof(str),strlen(str));
        return 0;
}
```

Points-

- Sizeof is an operator and strlen is a function.
- Value of sizeof remain same for a variable but strlen may change.

Data Structure Training

(String)

```
1   #include <stdio.h>
2   #include<string.h>
3   int main(void)
4   {
5         char str[10]="hello";
6         printf("\n%d :: %d",sizeof(str),strlen(str));
7         strcpy(str,"bye");
8         printf("\n%d :: %d",sizeof(str),strlen(str));
9         return 0;
10  }
```

Data Structure Training

(String)

String Handling Functions

Strlen()	Length of string
Strcpy()	Copies a string into another
Strcmp()	Compare two string
Strcat()	Concatenate two string
Strrev()	Reverse the string
Strlwr()	Convert into lowercase
Strupr()	Convert into upper case
Strncpy()	Copy first n char of string into another
Strncmp()	Compare first n char of string
Strchr()	Find first occurrence of char into string
Strrchr()	Find last occurrence of char into string
stricmp	Compare two string without regard the case

Data Structure Training

(String)

Strlen()-

- Used for calculate the length of string.
- Syntax

```
int strlen(const char *str)
```

```
1   #include<stdio.h>
2   #include <string.h>
3   int main()
4   {
5         char str[] ="hello";
6         printf("Length of string str is: %d", strlen(str));
7         return 0;
8   }
```

Data Structure Training

(String)

mystrlen()-(implementation of strlen function)

```
1   #include<stdio.h>
2   int mystrlen(char *s);
3   void main(void)
4   {
5           char str[]="hello";
6           int c=mystrlen(str);
7           printf("%d",c);
8   }
9   int mystrlen(char *s)
10  {
11          int c=0;
12          while( *s != '\0')
13          {
14                  ----
15                  ----
16          }
17          return c;
18  }
```

Data Structure Training

(String)

mystrlen()-(implementation of strlen function)

```
1   #include<stdio.h>
2   int mystrlen(char *s);
3   void main(void)
4   {
5           char str[]="hello";
6           int c=mystrlen(str);
7           printf("%d",c);
8   }
9   int mystrlen(char *s)
10  {
11          int c=0;
12          while( *s != '\0')
13          {
14                  c++;
15                  s++;
16          }
17          return c;
18  }
```

Data Structure Training

(String)

mystrlen()-(implementation of strlen function)

```
1   #include<stdio.h>
2   int mystrlen(char *s);
3   int main(void)
4   {
5           char str[]="hello";
6           int c=mystrlen(str);
7           printf("%d",c);
8           return 0;
9   }
10  int mystrlen(char s[])
11  {
12          int c=0;
13          while(_____)
14          {
15                  _____
16          }
17          return c;
18  }
```

Data Structure Training

(String)

mystrlen()-(implementation of strlen function)

```
1   #include<stdio.h>
2   int mystrlen(char *s);
3   int main(void)
4   {
5           char str[]="hello";
6           int c=mystrlen(str);
7           printf("%d",c);
8           return 0;
9   }
10  int mystrlen(char s[])
11  {
12          int c=0;
13          while(s[c]!='\0')
14          {
15                  c++;
16          }
17          return c;
18  }
```

Data Structure Training

(String)

Strcat()-

- Used for concatenation of string.
- Syntax

```
char* strcat(char *str2,const char *str1)
```

append the content of str1 into end of the str2.

```
1  #include<string.h>
2  #include <stdio.h>
3  int main()
4  {
5          char str1[50] = "everyone";
6          char str2[50] = "how are you\t";
7          strcat(str2, str1);
8          printf("%s",str2);
9          return 0;
10 }
```

Data Structure Training

(String)

```
1   #include<stdio.h>
2   char* mystrcat(char *str2, const char *str1);
3   int main()
4   {
5           char str1[]="hello";
6           char str2[]="everyone";
7           mystrcat(str2,str1);
8           printf("%s",str2);
9           return 0;
10  }
11  char* mystrcat(char *str2, const char *str1)
12  {
13          char *ptr2=str2;
14          char *ptr1=str1;
15          while( *ptr2 != '\0')
16                  ptr2++;
17          while(*ptr1 != '\0')
18          {
19                  ------
20                  ptr2++;
21                  ptr1++;
22          }
23          *ptr2='\0';
24          return str2;
25  }
```

mystrcat()-

(implementation of strcat function)

Data Structure Training

(String)

```
1   #include<stdio.h>
2   char* mystrcat(char *str2, const char *str1);
3   int main()
4   {
5           char str1[]="hello";
6           char str2[]="everyone";
7           mystrcat(str2,str1);
8           printf("%s",str2);
9           return 0;
10  }
11  char* mystrcat(char *str2, const char *str1)
12  {
13          char *ptr2=str2;
14          char *ptr1=str1;
15          while( *ptr2 != '\0')
16                  ptr2++;
17          while(*ptr1 != '\0')
18          {
19                  *ptr2=*ptr1;
20                   ptr2++;
21                   ptr1++;
22          }
23          *ptr2='\0';
24          return str2;
25  }
```

mystrcat()-

(implementation of strcat function)

Data Structure Training

(String)

Strrev()

1-Used for reverse the given string.

2-Syntax-

```
char* strrev(char *str)
```

```
1  #include<stdio.h>
2  #include<string.h>
3  int main()
4  {
5          char str[]="hello";
6          printf("%s",strrev(str));
7          return 0;
8  }
```

Data Structure Training

(String)

```
1   #include<stdio.h>
2   #include<string.h>
3   char* mystrrev(char *str);
4   int main()
5   {
6           char str[]="hello";
7           mystrrev(str);
8           printf("%s",str);
9            return 0;
10  }
11  char* mystrrev(char *str)
12  {
13          int n=strlen(str);
14          int beg=0,end=n-1;
15          char temp;
16          while(beg<end)
17          {
18                  _______
19                  _______
20                  _______
21
22                  beg++;
23                  end--;
24          }
25          return str;
    }
```

mystrrev()-

(implementation of strrev function)

Data Structure Training

(String)

```c
#include<stdio.h>
#include<string.h>
char* mystrrev(char *str);
int main()
{
        char str[]="hello";
        mystrrev(str);
        printf("%s",str);
         return 0;
}
char* mystrrev(char *str)
{
        int n=strlen(str);
        int beg=0,end=n-1;
        char temp;
        while(beg<end)
        {
                temp=str[beg];
                str[beg]=str[end];
                str[end]=temp;
                beg++;
                end--;
        }
        return str;
}
```

mystrrev()-

(implementation of strrev function)

Data Structure Training

(String)

```
1   #include <stdio.h>
2   int fun(char *s1)
3   {
4           char *s2 = s1;
5           while(*++s1);
6                   return (s1-s2);
7   }
8   int main()
9   {
10          char *s = "hell0";
11          printf("%d", fun(s));
12  return 0;
13  }
```

Consider the following C program & Choose the correct option.

A. 5

B. 3

C. Syntax Error

D. Compiler error

Data Structure Training

(String)

```
1   #include <stdio.h>
2   int fun(char *s1)
3   {
4           char *s2 = s1;
5           while(*++s1);
6                   return (s1-s2);
7   }
8   int main()
9   {
10          char *s = "hell0";
11          Printf("%d", fun(s));
12  return 0;
13  }
```

Consider the following C program & Choose the correct option.

A. 5

B. 3

C. Syntax Error

D. Compiler error

Data Structure Training

(String)

```
1   #include <stdio.h>
2   int main()
3   {
4           char arr[] = "hello2020";
5           printf("%s", #);
6           return 0;
7   }
```

What would be the place of # symbol to print 2020?

A. arr

B. arr+5

C. arr+4

D. Compiler error

Data Structure Training

(String)

```
1   #include <stdio.h>
2   int main()
3   {
4           char arr[] = "hello2020";
5           printf("%s", arr+5);
6           return 0;
7   }
```

What would be the place of # symbol to print 2020?

A. arr

B. arr+5

C. arr+4

D. Compiler error

Data Structure Training

(String)

```
1  #include <stdio.h>
2  int main()
3  {
4  char str[] = "%d %c", arr[] = "hello2020";
5          printf(str, 0[arr], 2[arr + 3]);
6  return 0;
7  }
```

What would be the output of the above code ? Choose the correct option.

A. 30 1

B. 2 30

C. 104 2

D. Error

Data Structure Training

(String)

```
1  #include <stdio.h>
2  int main()
3  {
4  char str[] = "%d %c", arr[] = "hello2020";
5          printf(str, 0[arr], 2[arr + 3]);
6  return 0;
7  }
```

What would be the output of the above code ? Choose the correct option.

A. 30 1

B. 2 30

C. 104 2

D. Error

Data Structure Training

(String)

```
1  #include<stdio.h>
2  int main()
3  {
4          char str[20] = "hello2020";
5          printf ("%d", sizeof(str));
6  return 0;
7  }
```

What would be the output of the above code ? Choose the correct option.

A. 10

B. 15

C. 20

D. Error

Data Structure Training

(String)

```
1  #include<stdio.h>
2  int main()
3  {
4          char str[20] = "hello2020";
5          printf ("%d", sizeof(str));
6  return 0;
7  }
```

What would be the output of the above code ? Choose the correct option.

A. 10

B. 15

C. 20

D. Error

Data Structure Training

(String)

```
1    #include <stdio.h>
2    int main(void)
3    {
4            char p[]="gate2020";
5                    printf("%s",p+p[3]-p[1]);
6            return 0;
7    }
```

What would be the output of the above code ? Choose the correct option.

A. te2020

B. gte

C. 2020

D. gte2020

Data Structure Training

(String)

```
1    #include <stdio.h>
2    int main(void)
3    {
4            char p[]="gate2020";
5                    printf("%s",p+p[3]-p[1]);
6            return 0;
7    }
```

What would be the output of the above code ? Choose the correct option.

A. te2020

B. gte

C. 2020

D. gte2020

Data Structure Training

(String)

```
1   #include <stdio.h>
2   int main(void)
3   {
4           char a[]="hello";
5           char b[]="hello";
6           if(a==b)
7                   printf("Hello World");
8           else
9                   printf("Hello");
10          return 0;
11  }
```

What would be the output of the above code ? Choose the correct option.

A. Hello world

B. Hello5

C. error

D. Hello

Data Structure Training

(String)

```
1   #include <stdio.h>
2   int main(void)
3   {
4           char a[]="hello";
5           char b[]="hello";
6           if(a==b)
7                   printf("Hello World");
8           else
9                   printf("Hello");
10          return 0;
11  }
```

What would be the output of the above code ? Choose the correct option.

A. Hello world

B. Hello5

C. error

D. Hello

Data Structure Training

(String)

What would be the output of the above code ? Choose the correct option.

A. achi

B. p

C. chacha

D. chachi

```
1   #include <stdio.h>
2   int main(void)
3   {
4   char *a[]= {"papa", "stupid", "gotohell", "break", "chacha", "chachi"};
5   char **b[]={a+2,a+3,a+4,a,a+1,a+5};
6   char ***c=b;
7   *c++;
8   printf("%s",**++c);
9   return 0;
10  }
```

Data Structure Training

(String)

What would be the output of the above code ? Choose the correct option.

A. achi

B. p

C. chacha

D. chachi

```
1   #include <stdio.h>
2   int main(void)
3   {
4   char *a[]= {"papa", "stupid", "gotohell", "break", "chacha", "chachi"};
5   char **b[]={a+2,a+3,a+4,a,a+1,a+5};
6   char ***c=b;
7   *c++;
8   printf("%s",**++c);
9   return 0;
10  }
```

Data Structure Training

(String)

```
1   #include <stdio.h>
2   int main(void)
3   {
4   char *a[]= {"papa", "stupid", "gotohell", "break", "chacha", "chachi"};
5   char **b[]={a+2,a+3,a+4,a,a+1,a+5};
6   char ***c=b;
7   *c++;
8   printf("%s\n",**++c);
9   printf("%s",*++*c+2);
10  return 0;
11  }
```

What would be the output of the above code in second printf() ? Choose the correct option.

A. p

B. achi

C. chacha

D. chachi

Data Structure Training

(String)

```
1   #include <stdio.h>
2   int main(void)
3   {
4   char *a[]= {"papa", "stupid", "gotohell", "break", "chacha", "chachi"};
5   char **b[]={a+2,a+3,a+4,a,a+1,a+5};
6   char ***c=b;
7   *c++;
8   printf("%s\n",**++c);
9   printf("%s",*++*c+2);
10  return 0;
11  }
```

What would be the output of the above code in second printf()? Choose the correct option.

A. p

B. achi

C. chacha

D. chachi

Data Structure Training

(String)

```
1   #include <stdio.h>
2   int main(void)
3   {
4   char *a[]= {"papa", "stupid", "gotohell", "break", "chacha", "chachi"};
5   char **b[]={a+2,a+3,a+4,a,a+1,a+5};
6   char ***c=b;
7   *c++;
8   printf("%s\n",**++c);
9   printf("%s\n",*++*c+2);
10  printf("%c",*(**++c+2));
11  return 0;
12  }
```

What would be the output of the above code in third printf() ? Choose the correct option.

A. achi

B. p

C. chacha

D. chachi

Data Structure Training

(String)

```
1   #include <stdio.h>
2   int main(void)
3   {
4   char *a[]= {"papa", "stupid", "gotohell", "break", "chacha", "chachi"};
5   char **b[]={a+2,a+3,a+4,a,a+1,a+5};
6   char ***c=b;
7   *c++;
8   printf("%s\n",**++c);
9   printf("%s\n",*++*c+2);
10  printf("%c",*(**++c+2));
11  return 0;
12  }
```

What would be the output of the above code in third printf() ? Choose the correct option.

A. achi

B. p

C. chacha

D. chachi