Обробка символьних рядків

Визначення. Базові поняття

Рядком називається послідовність символів, з якими маніпулюють як з одним елементом.

Рядок може містити букви, цифри, різні *спеціальні символи*, такі як **+**, **-**, ***/***, **?**, **$** та інші.

В мові C/C++ рядкові літерали або рядки-константи обмежуються подвійними лапками: "Ігор Іванов", "03127, м. Київ", "(044) 258-1212".

З точки зору компілятора мови С, рядок – це масив символів, який завершується *нульовим символом* ('\0').

Визначення. Базові поняття

Доступ к рядку здійснюється за допомогою **покажчика**, який посилається на перший символ рядка.

Таким чином, рядок подібний масиву, оскільки масив також є покажчиком на його перший елемент.

Рядок може бути присвоєний в оголошенні або масиву символів, або покажчику на символ.

Визначення. Базові поняття

Кожне з оголошень ініціює змінну рядком “***yellow***”:

char color [] = “yellow”;

char *ptrColor = “yellow”;

Перше оголошення створює масив ***color***, який складається з 7 елементів і містить символи ‘*y*',’e’, ’l’, ‘l', ‘*o*', ‘*w*', '\0’.

Друге оголошення створює змінну-покажчик ***ptrColor***, яка містить адресу рядка *“yellow"*, що знаходиться у деякому місці пам'яті.

При оголошенні масивів символів для збереження рядка вони повинні мати достатній розмір, щоб вмістити і рядок, і обмежувальний нульовий символ.

Робота з рядками

Робота з рядками реалізується за допомогою стандартних функцій, які поділяються за їхнім призначенням на такі групи.

1. **Введення-виведення рядків.**
2. **Перетворення рядків.**
3. **Операції над рядками.**
4. **Порівняння рядків.**
5. **Пошук символів та рядків**
6. **Операції з пам'яттю.**

Введення-виведення рядків

Введення рядків:

scanf(“%s”, str1);

Використання функції scanf () для введення рядка - працює, але це може призвести до переповнення буфера. Адже вхідний рядок може виявитися більше, ніж розмір рядка-буфера

Виведення рядків:

printf(“%s”, str1);

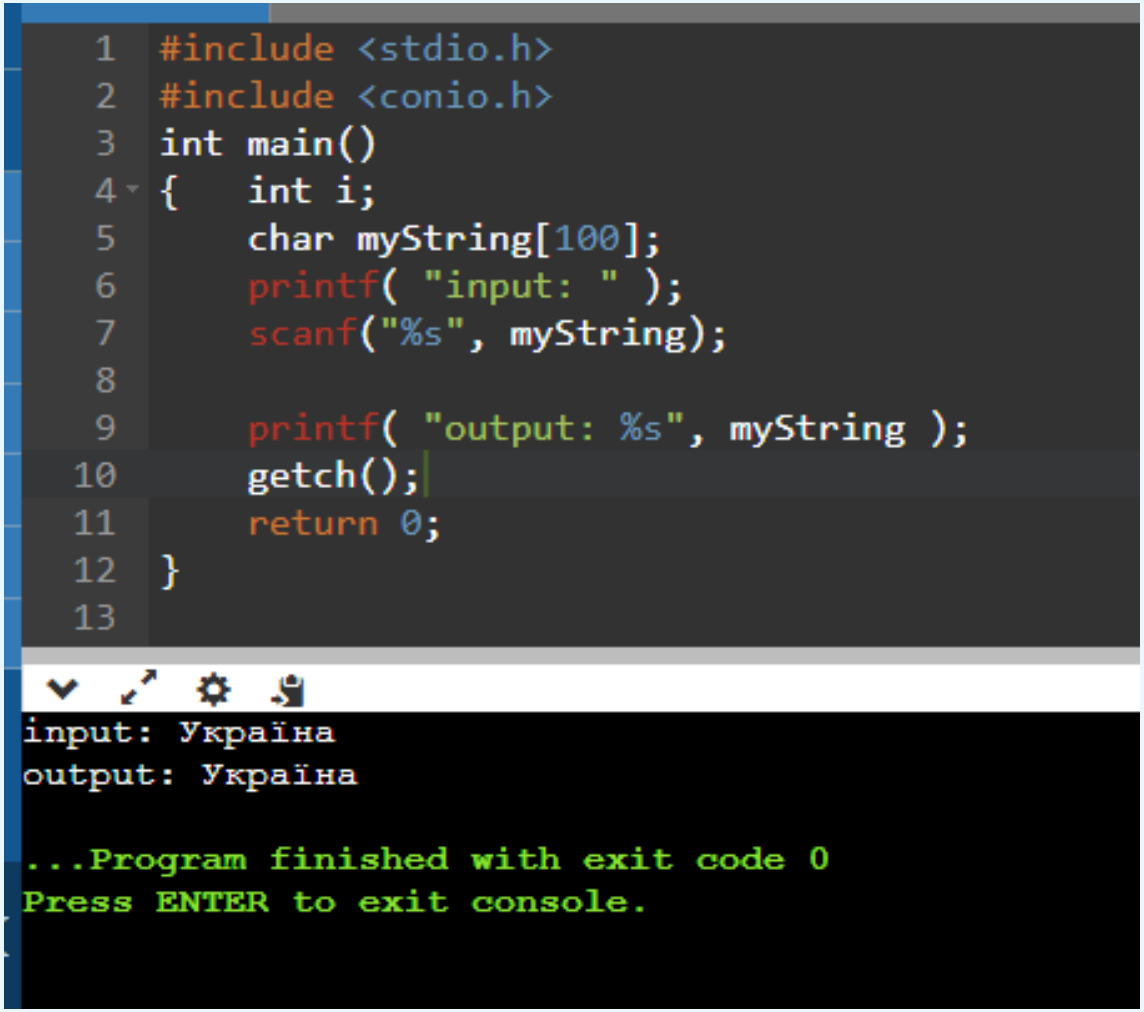
Введення-виведення рядків

```
#include <stdio.h>
#include <conio.h>
int main()
{   int i;
        char myString[100];
        printf( "input: " );
        scanf("%s", myString);
            printf( "output: %s", myString );
        getch();
        return 0;
}
```

& перед **myString** у функції scanf відсутнє.

Сама назва масиву представляє адресу першого елемента, тому оператор & у цьому контексті не потрібен.

Введення-виведення рядків



```c
#include <stdio.h>
#include <conio.h>
int main()
{   int i;
    char myString[100];
    printf( "input: " );
    scanf("%s", myString);

    printf( "output: %s", myString );
    getch();
    return 0;
}
```

```
input: Україна
output: Україна

...Program finished with exit code 0
Press ENTER to exit console.
```

Введення-виведення рядків

Для обробки символьних типів даних бібліотека функцій **string.h** має велику кількість вбудованих функцій, що збільшують продуктивність праці програмістів та скорочують час на розробку програм

Введення-виведення рядків

Функція **gets()** дозволяє читати рядок з клавіатури

Функція **gets()** читає рядок символів, введених з клавіатури і поміщає їх за адресою, вказаною в аргументі.

Можна набирати символи, поки не буде натиснуто **Enter**.

Символ, що відповідає клавіші **Enter** - повернення каретки, - не стане частиною рядка.

Замість цього в кінці рядка з'явиться нульовий символ, і **gets()** закінчить роботу.

Якщо при введенні допущені помилки, то вони можуть бути виправлені натисканням на клавішу BACKSPASE перед натисканням введення.

Введення-виведення рядків

Функція **gets()** має прототип:

char *gets(char *str);

де **str** - це масив символів.

Функція **gets()** повертає покажчик на **str**.

```
#include <stdio.h>
#include <string.h>
int main() {
        char str[80];
        printf("input string\n");
        gets(str);
        printf("Output string\n%s", str);
        return 0;
}
```

```
input string
1234qwerty
Output string
1234qwerty_
```

Введення-виведення рядків

Функція **fgets** найбезпечніша функція зчитування в С.

На відміну від функції **gets** одним із її аргументів є обмежувач числа символів, які потрібно зчитати, тому небезпека переповнення буферу набагато менша.

Функцією **fgets** рекомендується зчитувати не тільки рядки з файлів, а й рядки з вхідного потоку **stdin**.

Введення-виведення рядків

У цьому випадку виклик функції має вигляд:

result = fgets(str, num, stdin);

Функцію **fgets** рекомендується використовувати і в тому випадку, коли з файла потрібно зчитати число.

Оскільки програміст, який пише програму, не може бути впевненим, що в файлі, вказаному коритувачем, завжди буде правильна очікувана інформація, безпечно спочатку зчитати вхідний рядок як рядок, проаналізувати його, а потім зчитати потрібні числа уже з буфера.

fgets(str, sizeof(str), stdin);

```c
#include <stdio.h>
#include <string.h>

int main(){
    char str[80];
    printf("input string\n");
    fgets(str, sizeof(str), stdin);
    printf("Output string\n%s", str);
return 0;
}
```

```
input string
Доброго ранку!
Output string
Доброго ранку!


...Program finished with exit code 0
Press ENTER to exit console.
```

Введення-виведення рядків

Є проблема, пов'язана з **gets()** , про яку слід знати:

використовуючи **gets()** , можна перейти межі масиву, з яким вона викликалася.

Це можливо, оскільки не існує способу вказати **gets()** , де знаходиться межа масиву.

Наприклад, якщо викликати **gets()** з масивом довжиною в 40 байт, а потім ввести 40 або більше символів, то станеться вихід за межі масиву.

Функція **puts()** дозволяє виводити рядок на консоль (екран).

Функція **puts()** виводить аргумент, що отримала, на екран, завершуючи виведення перехідом на новий рядок.

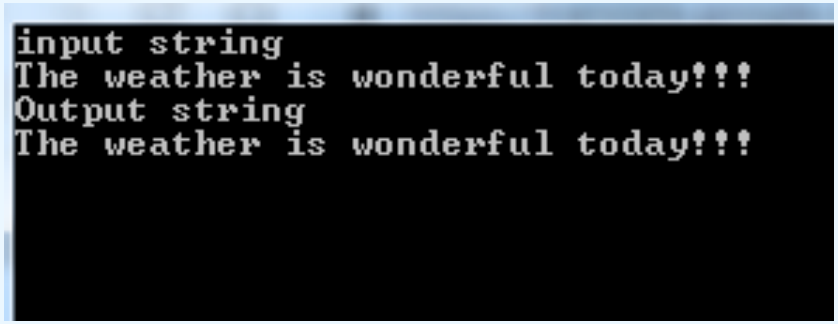
Введення-виведення рядків

Функція має такий прототип:

int puts(const char *str);

де str - це рядок, що потрєбує виведення

```
#include <stdio.h>
#include <string.h>
int main() {
	char str[80];
	printf("input string\n");
	gets(str);
	printf("Output string\n");
	puts(str);
	return 0;
}
```



Введення-виведення рядків

```c
#include <stdio.h>
#include <string.h>

int main() {
    char str[80];
    printf("input string\n");

    fgets(str, sizeof(str), stdin);

    size_t len = strlen(str);
    if (len > 0 && str[len - 1] == '\n') {
        str[len - 1] = '\0';
    }

    printf("\nOutput string\n");
    puts(str);

    return 0;
}
```

```
input string
Welcome

Output string
Welcome


...Program finished with exit code 0
Press ENTER to exit console.
```

size_t - беззнаковий цілий тип, призначений для подання розміру будь-якого об'єкта в пам'яті (включаючи масиви) у конкретній реалізації.

Введення-виведення рядків

Функція повертає неціле число в разі успіху і **EOF** - в разі невдачі.

Вона розуміє коди зі зворотним слешем, як printf(), наприклад \ t сприймається як табуляція.

Виклик функції **puts()** вимагає набагато менше процесорного часу на реалізацію, ніж **printf()**, оскільки **puts()** виводить тільки рядок символів.

Вона не може виводити числа або виконувати перетворення форматів.

Довжина рядка

Функція **strlen** обчислює кількість символів в рядку до першого входження символу кінця рядка.

При цьому символ кінця рядка не входить в підраховану кількість символів.

```
char str[] = "13156676";
printf ("length of string *%s* - %d symbols\n", str, strlen (str) );
```

```
length of string *13156676* - 8 symbols
```

Перетворення рядків

char *strlwr (char*st); — перетворює символи рядка **st** верхнього регістра в символи нижнього регістра, при цьому інші символи не враховуються

char *strupr (char *st); — перетворює символи рядка **st** нижнього регістра в символи верхнього регістра, інші символи не враховуються

char *strrev (char *st); — записує символи в рядку **st** у зворотному порядку (реверсує рядок)

Приклад. Перетворення рядків

```
#include <stdio.h>
#include <string.h>

int main() {
        char str[80];
        printf("Input string\n");
        gets(str);

        printf("Output strings\n");
        puts(strlwr(str));
        puts(strupr(str));
        puts(strrev(str));
        return 0;
}
```

char *strlwr (char*st);

```
input string
VERY Well
Output string
very well
```

char *strupr (char *st);

```
input string
verY weLl 15/03/19
Output string
VERY WELL 15/03/19
```

char *strrev (char *st);

```
Input string
1234567890
Output string
0987654321
```

```
Input string
Today Is Friday
Output string
today is friday
yadirf si yadot
YADIRF SI YADOT
TODAY IS FRIDAY
```

```
Input string
tOday is FridaY
Output string
today is friday
TODAY IS FRIDAY
YADIRF SI YADOT
```

Перетворення рядків

функції **strlwr**, **strupr** і **strrev** недоступні в онлайн-компіляторі, який ви використовуєте. Щоб отримати ту саму функціональність, ви можете вручну реалізувати ці функції.

Операції над рядками

Операція конкатенації

char *strcat (char *st1, const char *st2); — поєднує **st1 і st2** та повертає **st1**

char *strncat (char *st1, const char *st2, int n); — додає до рядка **st1 n** символів рядка **st2** і повертає знову в **st1**

Приклад. Операція конкатенації


```
#include <stdio.h>
#include <string.h>

int main() {
        char str1[80];
        char str2[80];
        printf("Input string 1\n");
        gets(str1);
        printf("Input string 2\n");
        gets(str2);

        printf("Output string\n");
        puts(strcat(str1,str2));
        puts(str1);
        puts(str2);
        return 0;
}
```

```
1  #include <stdio.h>
2  #include <string.h>
3
4  int main() {
5      char str1[80];
6      char str2[80];
7
8      printf("Input string 1\n");
9      fgets(str1, sizeof(str1), stdin);
10
11     size_t len1 = strlen(str1);
12     if (len1 > 0 && str1[len1 - 1] == '\n') {
13         str1[len1 - 1] = '\0';
14     }
15
16     printf("Input string 2\n");
17     fgets(str2, sizeof(str2), stdin);
18
19     size_t len2 = strlen(str2);
20     if (len2 > 0 && str2[len2 - 1] == '\n') {
21         str2[len2 - 1] = '\0';
22     }
23
24     printf("\nOutput string\n");
25     puts(strcat(str1, str2));
26     puts(str1);
27     puts(str2);
28
29     return 0;
30 }
```

Приклад. Операція конкатенації

char *strcat (char *st1,
const char *st2);

```
Input string 1
12345
Input string 2
zxcvb
Output string
12345zxcvb
```

char *strncat (char *st1,const char *st2, int n);

strncat(str1,str2,10);

```
Input string 1
qwert
Input string 2
12345
Output string
qwert12345
```

strncat(str1,str2,5);

```
Input string 1
asd
Input string 2
123456789
Output string
asd12345
```

Приклад. Операція конкатенації

```
Input string 1
1234556
Input string 2
qwerty
Output string
1234556qwerty
1234556qwerty
qwerty
```

```
Input string 1
12345
Input string 2
qwert

Output string
12345qwert
12345qwert
qwert


...Program finished with exit code 0
Press ENTER to exit console.
```

Приклад. Операція конкатенації

```
#include <stdio.h>
#include <string.h>

int main() {
        char str1[80];
        char str2[80];
        printf("Input string 1\n");
        gets(str1);
        printf("Input string 2\n");
        gets(str2);

        printf("Output string\n");
        puts(strncat(str1,str2,3));
        puts(str1);
        puts(str2);
        return 0;
}
```

```
#include <stdio.h>
#include <string.h>

int main() {
    char str1[80];
    char str2[80];

    printf("Input string 1\n");
    fgets(str1, sizeof(str1), stdin);

    size_t len1 = strlen(str1);
    if (len1 > 0 && str1[len1 - 1] == '\n') {
        str1[len1 - 1] = '\0';
    }

    printf("Input string 2\n");
    fgets(str2, sizeof(str2), stdin);

    size_t len2 = strlen(str2);
    if (len2 > 0 && str2[len2 - 1] == '\n') {
        str2[len2 - 1] = '\0';
    }

    printf("\nOutput string\n");
    puts(strncat(str1, str2, 3));
    puts(str1);
    puts(str2);

    return 0;
}
```

Приклад. Операція конкатенації

```
Input string 1
qwerty
Input string 2
1234567890
Output string
qwerty123
qwerty123
1234567890
```

```
Input string 1
123456
Input string 2
qwerty

Output string
123456qwe
123456qwe
qwerty


...Program finished with exit code 0
Press ENTER to exit console.
```

Функції копіювання рядків:

char strcpy (s, *st); — виконує операцію копіювання байтів рядка **st** у рядок **s**(включаючи **"\0"**; повертає **s**),

char *strdup (const char *str); — виконує копіювання рядка **str** і повертає покажчик на рядок-копію

char * strncpy (char *st1, const char *st2, int n); — виконує копіювання **n** символів з рядка **st2 y st1** (рядок **st1** повинен бути більше або дорівнювати **st2**, інакше виникне помилка),

Приклад використання
strcpy (s, *st);


```
#include <stdio.h>
#include <string.h>

int main() {
        int t, i;
        char str1[80];
        char str2[80]="\0";
        printf("Input string 1\n");
        gets(str1);
        printf(" string1\n");
        puts(str1);
        printf(“
        strcpy(str2,str1)\n”);

        strcpy(str2,str1);
        printf(" string2\n");

        puts(str2);
        return 0;
}
```

```
1  #include <stdio.h>
2  #include <string.h>
3  int main()
4  {
5      int t,i;
6      char str1[80];
7      char str2[80]="\0";
8      printf("Ввод рядка: ");
9      fgets(str1, sizeof(str1), stdin);
10
11     size_t len1 = strlen(str1);
12     if (len1 > 0 && str1[len1 - 1] == '\n')
13         str1[len1 - 1] = '\0';
14
15     printf("\nПочатковий рядок: ");
16     puts(str1);
17     printf("\nВиконання функції strcpy(str2,str1) ");
18     printf("\nта отримання результату копіювання:\n ");
19
20     strcpy(str2,str1);
21     puts(str2);
22     return 0;
23 }
```

Приклад використання
strcpy (s, *st);

```
Input string 1
12345
 string1
12345
 strcpy(str2,str1)
12345
 string2
12345
```

char strcpy (s, *st);

```
Ввод рядка: Welcome!!!

Початковий рядо: Welcome!!!

Виконання функції strcpy(str2,str1)
та отримання езультату копіювання:
 Welcome!!!


...Program finished with exit code 0
Press ENTER to exit console.
```

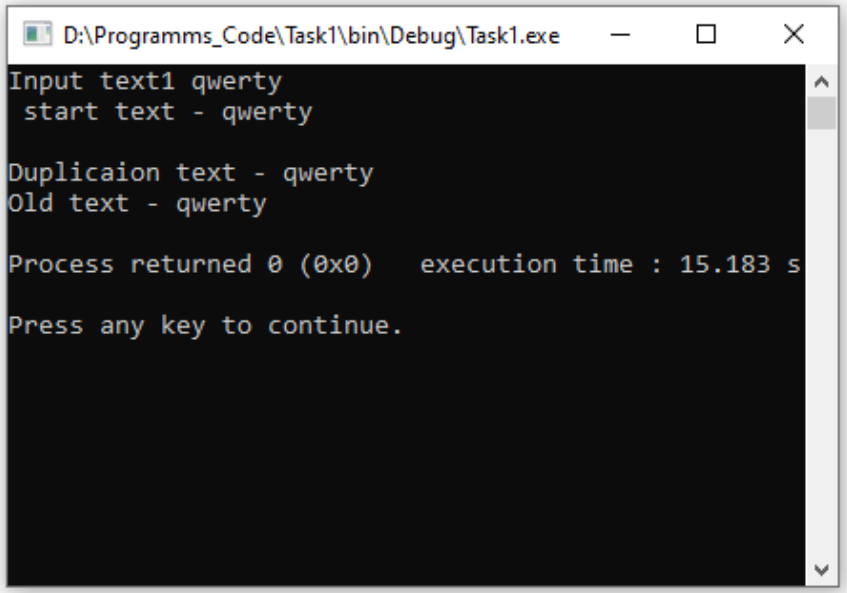
Приклад використання
strcpy (s, *st);


```
1  #include <stdio.h>
2  #include <string.h>
3
4  int main(){
5      char str1[80];
6      char str2[80];
7      printf("Input string 1\n");
8      gets(str1);
9      printf("Input string 2\n");
10     gets(str2);
11
12     printf("Output string\n");
13     puts(strcat(str1,str2));
14
15     printf("\n");
16     puts(str1);
17     puts(str2);
18     return 0;
19 }
20
```

```
D:\Programms_Code\Task1\bin\Debug\Task1.exe
Input string 1
123456
Input string 2
welcome
Output string
123456welcome

123456welcome
welcome

Process returned 0 (0x0)   execution time : 25.238 s
Press any key to continue.
```

Приклад використання
*strdup (const char *str)

```c
#include <stdio.h>
#include <string.h>

int main(){
    int t;
    char strl[80];
    char *str3;

    printf("Input textl ");
    gets(strl);
    printf(" start text - ");
    puts(strl);

    str3=strdup(strl);
        //дублювання рядку

    printf("\nDuplicaion text - %s\n",str3);
    printf("Old text - ");
    puts(strl);
    return 0;
}
```

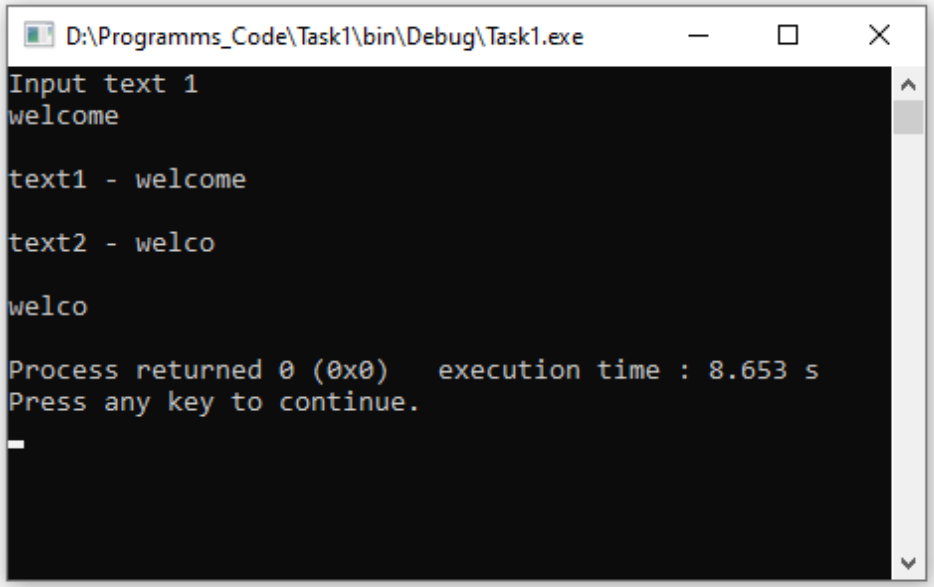


Приклад використання *strdup (const char *str)


```
#include <stdio.h>
#include <string.h>

int main(){
    int t;
    char str1[80];
    char *str3;

    printf("Input text1 ");
    gets(str1);
    printf(" start text - ");
    puts(str1);
    str3=strdup(str1);
            //дублювання рядку

    printf("\nDuplicaion text -
%s\n",str3);
    printf("Old text - ");
    puts(str1);
    return 0;
}
```

Приклад використання
strncpy (s, *st, n);

```c
#include <stdio.h>
#include <string.h>

int main(){
    int t,i;
    char str1[80];
    char str2[80]="\0";
    printf("Input text 1\n");
    gets(str1);
    printf(" \ntext1 - ");
    puts(str1);
    strncpy(str2,str1,5);
    printf(" \ntext2 - ");
     for (i=0;i<5;i++)
          printf ("%c", str2[i]);
       printf ("\n\n");

    puts(str2);
    return 0;
}
```



Приклад використання
strncpy (s, *st, n);


```
#include <stdio.h>
#include <string.h>

int main(){
    int t,i;
    char str1[80];
    char str2[80]="\0";
    printf("Input text 1\n");
    gets(str1);
    printf(" \ntext1 - ");
    puts(str1);
    strncpy(str2,str1,5);
    printf(" \ntext2 - ");
    for (i=0;i<5;i++)
        printf ("%c", str2[i]);
    printf ("\n\n");

    puts(str2);
    return 0;
}
```

Порівняння рядків.

int strcmp (char *stl, char *st2); — порівнює рядки **st1 і st2** та повертає цілу величину, що дорівнює:

<0 — якщо st1 < st2;

= 0 — якщо st1 = st2;

>0 — якщо st1 > st2;

int stricmp (const char *stl, const char *st2); — виконує порівняння рядків, не враховуючи регістра символів; повертає цілу величину, як і функція **strcmp()**,

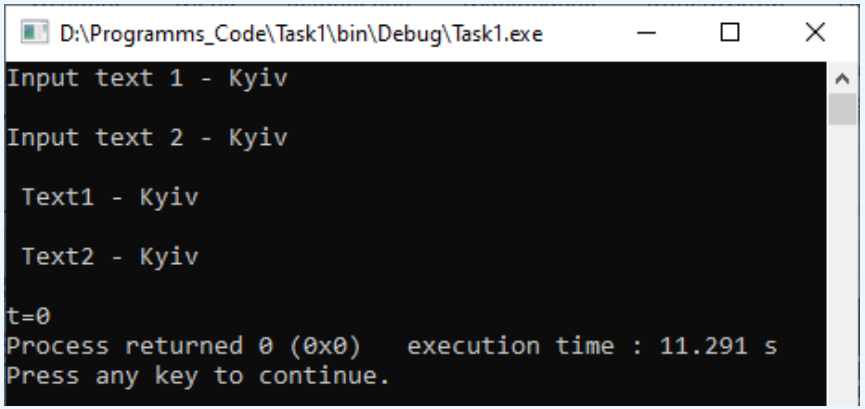
Приклад. Порівняння рядків

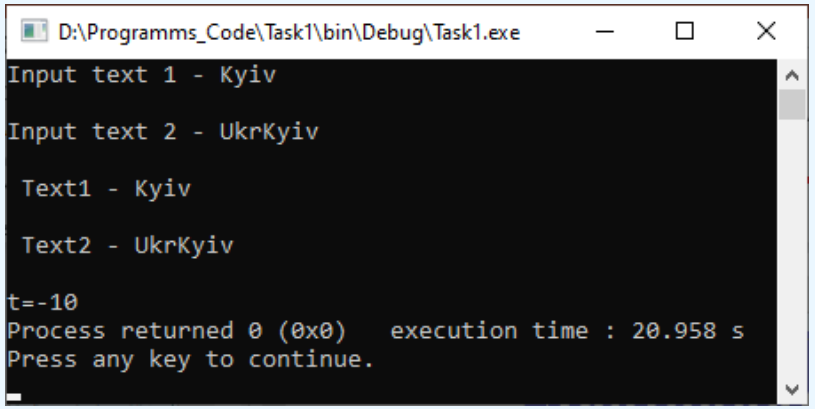
```
1   #include <stdio.h>
2   #include <string.h>
3
4   int main() {
5       char str1[80];
6       char str2[80]="\0";
7
8       printf("Input text 1 - ");
9       gets(str1);
10      printf("\nInput text 2 - ");
11      gets(str2);
12      printf("\n Text1 - ");
13      puts(str1);
14      printf("\n Text2 - ");
15      puts(str2);
16          //int t=strcmp(str1,str2);
17      int t=stricmp(str1,str2);
18      printf ("\nt=%d",t);
19      getch();
20      return 0;
21  }
```

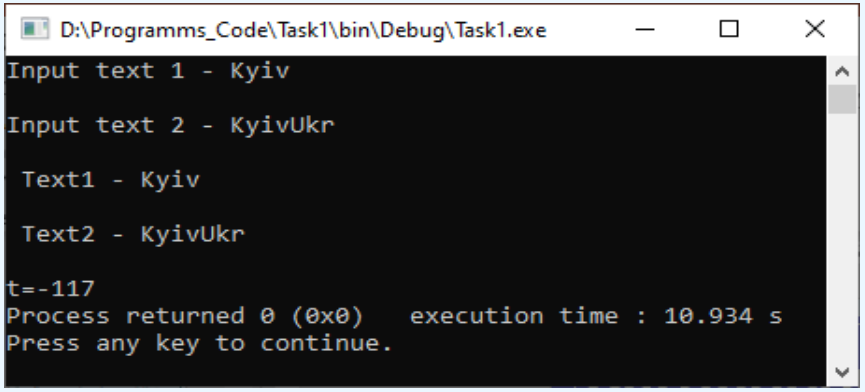
Приклад. Порівняння рядків

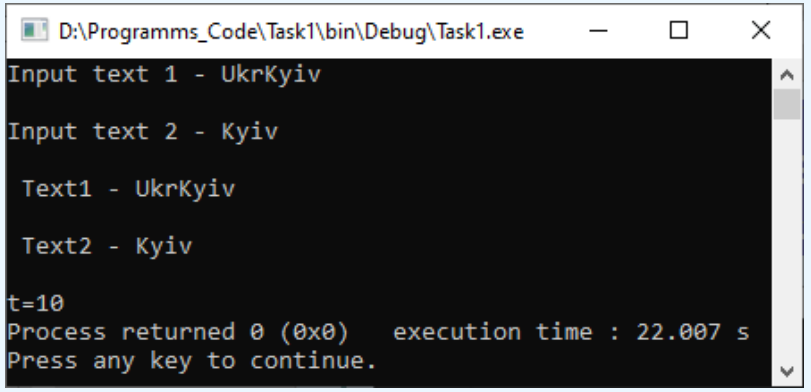
```
#include <stdio.h>
#include <string.h>

int main() {
        char str1[80];
        char str2[80]="\0";

        printf("Input string 1\n");
        gets(str1);
        printf("Input string 2\n");
        gets(str2);
        printf(" string1\n");
        puts(str1);
        printf(" string2\n");
        puts(str2);
        //t=strcmp(str1,str2);
        int t=stricmp(str1,str2);
        printf ("\nt=%d",t);
        getch();
        return 0;
}
```

Приклад. Порівняння рядків









Пошук символів та рядків

Функції пошуку підрядка в рядку:

int strspn (const char *st1, const char *st2); — повертає кількість символів від початку рядка **st1**, що збігаються із символами рядка **st2**, де б вони не знаходилися в **st2**,

char *strstr (const char *st1, const char *st2); — функція шукає в рядку **st1** перше входження **st2** і повертає покажчик на перший символ, знайдений у **st1**, з підрядка**st2**; якщо рядок **st2** не виявлений в **st1**, функція повертає **0**,

Приклад. Пошук символів та рядків

```c
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

int main() {
   char s1 [15]="0123456789345";
   char s2 [10]="345";
   char *Ps;
   puts(s1);
   puts(s2);
   Ps = strstr (s1,s2);

   if ( Ps == NULL)
         printf ("String not found\n");
      else
         printf ("The search string starts with a character %d\n",Ps-s1+1);

   return 0;
}
```

```
D:\Programms_Code\task2\bin\Debug\task2.exe
0123456789345
345
The search string starts with a character 4

Process returned 0 (0x0)   execution time : 0.264 s
Press any key to continue.
```

```
Select D:\Programms_Code\task2\bin\Debug\task2....
0123456789345
210
String not found

Process returned 0 (0x0)   execution time : 0.262 s
Press any key to continue.
```

```
D:\Programms_Code\task2\bin\Debug\task2.exe
qwertyuiop
ui
The search string starts with a character 7

Process returned 0 (0x0)   execution time : 0.221 s
Press any key to continue.
```

Приклад. Пошук символів та рядків


```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
```

```
int main() {
   char s1 [15]="qwertyuiop";
   char s2 [10]="ui";
   char *Ps;
   puts(s1);
   puts(s2);
   Ps = strstr (s1,s2);
```

```
   if ( Ps == NULL)
      printf ("String not found\n");
   else
      printf ("The search string starts with a character %d\n",Ps-s1+1);

   return 0;
}
```

Приклад.

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

int main() {
char s[] ="Hello! Very beatiful weather, on this spring is sunny days!";
   char * ps,ss[10];
   int l, i, f=0, r;
   puts(s);

   printf("\nInput word   \n");
   fgets(ss, sizeof(ss), stdin);
   l=strlen(ss);
   ps = strtok (s," ,.!");
```

```
while (ps != NULL) {
      i=0;
      if (*ps==*ss) {
         while (i<l) {
            printf ("\n%d  -  %s\n",i+1,ps);
            i++;f++;
         }
      }
      ps = strtok (NULL, " ,.-!");
   }
   if(f==0) printf("\nword %s not found", ss);
   return 0;
}
```

Розділювачі, що є у рядку

Пошук символів у тексті

char *strchr (const char *str, int ch);

Функція **strchr** шукає перше входження символу, код якого зазначений у аргументі ch, в рядку, на яку вказує аргумент str.

```
1   #include <stdio.h>
2   #include <stdlib.h>
3
4   int main(){
5       char str[] = "13156676";
6       char *pch;
7        int ch = '6';
8       pch=strchr (str,ch);
9       puts(str);
10      if (pch==NULL)
11            printf ("symbol not found\n");
12         else
13            printf ("sumbol %c on place %d\n",ch,pch-str+1);
14
15      return 0;
16  }
```

```
D:\Programms_Code\task3\bin\Debug\task3.exe
13156676
sumbol 6 on place 5

Process returned 0 (0x0)   execution time : 0.350 s
Press any key to continue.
```

Пошук символів у тексті

char *strpbrk (const char *str, const char *sym);

Функція **strpbrk** шукає перше входження в рядок, на який вказує аргумент str, одного з символів, що входять в рядок, на який вказує аргумент sym.

```c
#include <stdio.h>
#include <stdlib.h>

int main(){
    char str[] = "13156676";
    char sym [10]="369";
    char *psym;
    psym = strpbrk (str,sym);
    puts(str);
    if ( psym == NULL)
          printf ("symbol not found\n");
       else
          printf ("sumbols %s on place %d\n",sym,psym-str+1);

    return 0;
}
```

```
D:\Programms_Code\task3\bin\Debug\task3.exe
13156676
sumbols 369 on place 2

Process returned 0 (0x0)   execution time : 0.263 s
Press any key to continue.
```

Операції з пам'яттю

char *myString; *// покажчик типу char*

myString = malloc(sizeof(*myString) * 64);
// виділення пам’яті

В цьому прикладі виделено 64 комірки памя’ті для масиву **myString**.

Для звільнення пам’яті використовуємо функцію **free()**.

free(myString);

Операції з пам'яттю

Для переведення рядка в інші типи існує бібліотека **stdlib.h**

Функції:

int atoi (char *str)

long atol (char *str)

double atof (char *str)

Операції з пам'яттю

```c
#include <stdio.h>
#include <stdlib.h>

int main(){
    char *Str = "4567.99gfd";
    int Num=0;
    float  Num1=0;
    long int  Num2=0;
    puts(Str);
    Num = atoi (Str);
    Num1 = atof (Str);
    Num2 = atol (Str);
    printf ("%d\n",Num);
    printf ("%f\n",Num1);
    printf ("%ld\n",Num2);

    return 0;
}
```

```
D:\Programms_Code\task3\bin\Debug\task3.exe
4567.99gfd
4567
4567.990234
4567

Process returned 0 (0x0)   execution time : 0.374 s
Press any key to continue.
```

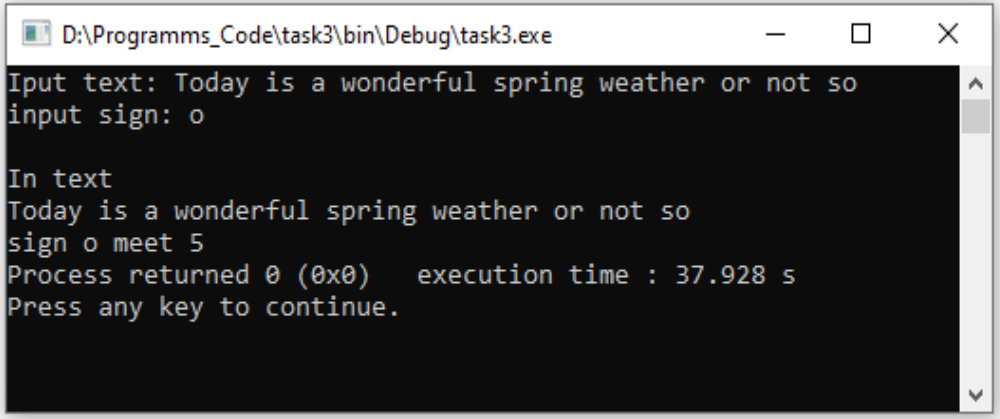
```c
#include <stdio.h>
#include <stdlib.h>

int main(){
    char *Str = "13144567.9089786556";
    double  Num1=0;
    puts(Str);
    Num1 = atof (Str);
    printf ("%lf\n",Num1);

    return 0;
}
```

```
D:\Programms_Code\task3\bin\Debug\task3.exe
13144567.9089786556
13144567.908979

Process returned 0 (0x0)   execution time : 0.273 s
Press any key to continue.
```

Приклад

Підрахувати кількість символів в рядку (як в масиві)

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int main(){
    char s[80], sym;
    int count, i;
    printf("Iput text: ");
    gets(s);
    printf("input sign: ");
    sym = getchar();
    count = 0;
    for(i=0; s[i]!='\0'; i++)  {
        if(s[i]==sym)   count++;
    }
    printf("\nIn text \n");
    puts(s);
    printf("sign ");
    putchar(sym);
    printf(" meet %d ",count);
    getchar();

    return 0;
}
```



Приклад

Підрахувати кількість символів в рядку (використовуючи функції роботи з рядками)

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int main(){
    char s[80], sym,*z;
    int count, i,t ;
    printf("Input text: ");
    gets(s);
    printf("Input character: ");
    sym = getchar();
    count = 0;
    puts(s);
    printf ("\nLooking for the %c character in \"%s\"...\n",sym,s);
    z=strchr(s,sym);
    while (z!=NULL) {
        printf ("found at %d\n",z-s+1);
        z=strchr(z+1,sym);
        count++;
    }

    if (count==1)
        printf("\n\nCharter %c  in string there is %d times",sym, count);
    else  if (count>1)
            printf("\n\nCharter %c  in string there are %d times",sym, count);
        else printf("Charter %c not found(((",sym);
    getchar();

    return 0;
}
```

Приклад

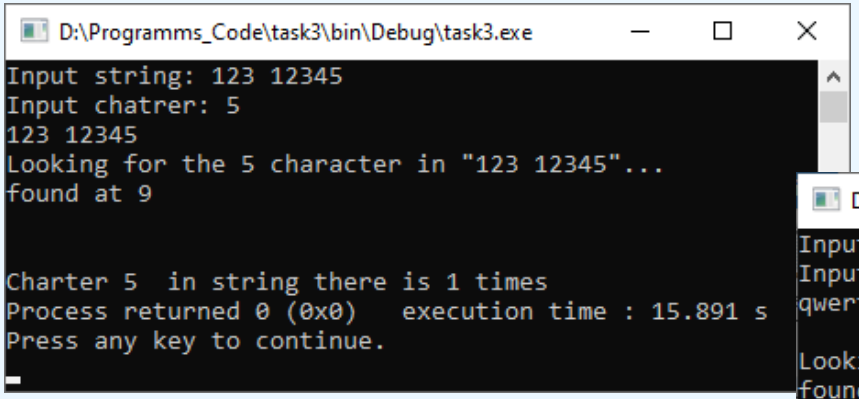
```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int main(){
    char s[80], sym,*z;
    int count, i,t ;
    printf("Input text: ");
    gets(s);
```

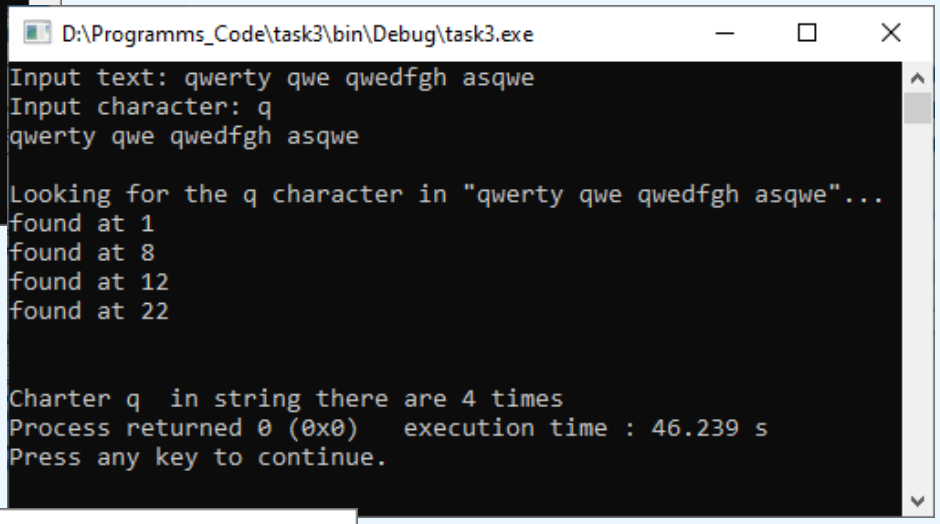
```
printf("Input character: ");
sym = getchar();
count = 0;
puts(s);
printf ("\nLooking for the %c character in \"%s\"...\n",sym,s);
z=strchr(s,sym);
```

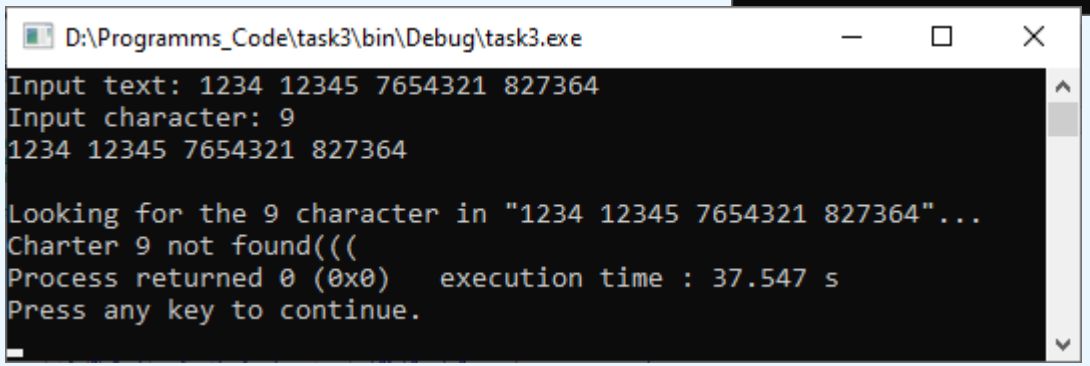
```
while (z!=NULL) {
        printf ("found at %d\n",z-s+1);
        z=strchr(z+1,sym);
        count++;
    }
```

```
    if (count==1)
        printf("\n\nCharter %c  in string there is %d times",sym, count);
    else  if (count>1)
            printf("\n\nCharter %c  in string there are %d times",sym, count);
        else printf("Charter %c not found(((",sym);
    getchar();

    return 0;
}
```

Приклад


D:\Programms_Code\task3\bin\Debug\task3.exe
Input string: 123 12345
Input chatrer: 5
123 12345
Looking for the 5 character in "123 12345"...
found at 9
Charter 5 in string there is 1 times
Process returned 0 (0x0) execution time : 15.891 s
Press any key to continue.


D:\Programms_Code\task3\bin\Debug\task3.exe
Input text: qwerty qwe qwedfgh asqwe
Input character: q
qwerty qwe qwedfgh asqwe
Looking for the q character in "qwerty qwe qwedfgh asqwe"...
found at 1
found at 8
found at 12
found at 22
Charter q in string there are 4 times
Process returned 0 (0x0) execution time : 46.239 s
Press any key to continue.


D:\Programms_Code\task3\bin\Debug\task3.exe
Input text: 1234 12345 7654321 827364
Input character: 9
1234 12345 7654321 827364
Looking for the 9 character in "1234 12345 7654321 827364"...
Charter 9 not found(((
Process returned 0 (0x0) execution time : 37.547 s
Press any key to continue.

Приклад. Варіант 1

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int main(){
    char s[80],st[80] ,sym,*z;
    int count, i,t,a,b,j=0 ;
    printf("Input text: ");
    gets(s);
    printf("Input 1 position : ");
    scanf("%d",&a);
    printf("Input 2 position : ");
    scanf("%d",&b);
    puts(s);
    for (i=0; i< strlen(s); i++) {
        if (i>=a&& i<=b)
            continue;
        else {
            st[j]=s[i];
            j++;
        }
    }
    printf("\n");
    st[j]='\0';
    puts(st);
    getchar();

    return 0;
}
```

Створіть функцію,
яка видаляє частину рядка s,
починаючи з положення a і
закінчуючи позицією b.

```
D:\Programms_Code\task3\bin\Debug\task3.exe
Input text: 12345678
Input 1 position : 3
Input 2 position : 6
12345678

1238

Process returned 0 (0x0)   execution time : 22.602 s
Press any key to continue.
```

Приклад. Варіант 1


```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int main(){
    char s[80],st[80] ,sym,*z;
    int count, i, t, a, b, j=0 ;
    printf("Input text: ");
    gets(s);
    printf("Input 1 position : ");
    scanf("%d",&a);
    printf("Input 2 position : ");
    scanf("%d",&b);
    puts(s);
```

```
for (i=0; i< strlen(s); i++) {
        if (i>=a && i<=b)
            continue;
        else {
            st[j]=s[i];
            j++;
        }
    }
    printf("\n");
    st[j]='\0';
    puts(st);
    getchar();

    return 0;
}
```

Приклад. Варіант 2

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int main(){
    char s[80],st[80] ,stl[80],*z;
    int a, b, l;
    printf("Input string: ");
    gets(s);
    printf("Input 1 position : ");
    scanf("%d",&a);
    printf("Input 2 position : ");
    scanf("%d",&b);
    puts(s);
    z=strdup(s);
    l=strlen(s);
    strncpy(st,s,a);
    z=z+b;
    strncpy(stl,z,l-b);
    st[a]='\0';
    stl[l-b]='\0';
    puts (st);
    puts(stl);
    puts(strcat(st,stl));
    getchar();

    return 0;
}
```

```
D:\Programms_Code\task3\bin\Debug\task3.exe
Input string: 123456 6789 12 3456
Input 1 position : 5
Input 2 position : 10
123456 6789 12 3456
12345
9 12 3456
123459 12 3456

Process returned 0 (0x0)   execution time : 40.846 s
Press any key to continue.
```

Приклад. Варіант 2


```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int main(){
    char s[80],st[80] ,st1[80],*z;
    int a, b, l;
    printf("Input string: ");
    gets(s);
    printf("Input 1 position : ");
    scanf("%d",&a);
    printf("Input 2 position : ");
    scanf("%d",&b);
    puts(s);
```

```
z=strdup(s);
    l=strlen(s);
    strncpy(st,s,a);
    z=z+b;
    strncpy(st1,z,l-b);
    st[a]='\0';
    st1[l-b]='\0';
    puts (st);
    puts(st1);
    puts(strcat(st,st1));
    getchar();

    return 0;
}
```

Розбиття рядка на лексеми

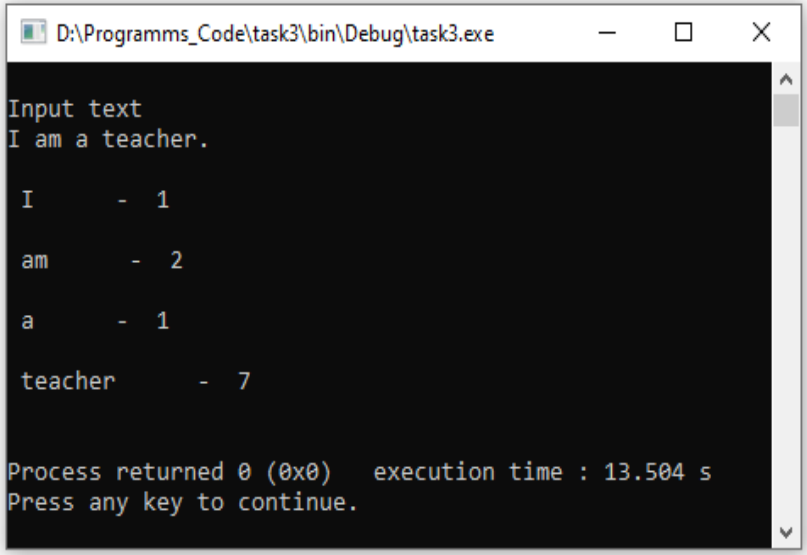
char *strtok (char *st, const char *dlm); — розбиття рядка на лексеми (сегменти), обмежені символами, що входять до складу рядка **dim**.

Цей параметр може містити будь-яку кількість різних обмежників — ознак границь лексем, після виділення лексеми в рядок **st** поміщається символ **«\0»**.

Приклад

Відобразити слова та довжину всіх слів у рядку

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int main(){
    char s[255];
    char * ps;
    printf("\nInput text  \n");
    gets(s);
    ps = strtok (s," !,.-?;");
    while (ps != NULL)
    {
        printf ("\n %s      -  %d\n",ps,strlen(ps));

        ps = strtok (NULL, "  !,.-?;");
    }
    getchar();

    return 0;
}
```



Приклад

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int main() {
    char s[255];
    char * ps;
    printf("\nInput text  \n");
    gets(s);
    ps = strtok (s," !,.-?;");
    while (ps != NULL)     {
        printf ("\n %s      -  %d\n", ps, strlen(ps));
        ps = strtok (NULL, "  !,.-?;");
    }
    getchar();

    return 0;
}
```

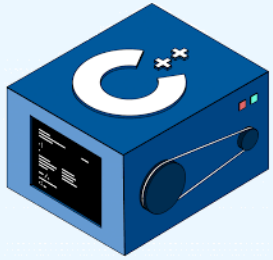
Приклад.

Відобразити слово, яке є в рядку, стільки разів, скільки його довжина

```c
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

int main()
{
char s[] ="Hello! Very beatiful weather, on this spring is sunny days!";
   char * ps,ss[10];
   int l, i, f=0, r;
   puts(s);

   printf("\nInput word   \n");
   fgets(ss, sizeof(ss), stdin);
   l=strlen(ss);
   ps = strtok (s," ,.-!");
   while (ps != NULL) {
      i=0;
      if (*ps==*ss){
         while (i<l){
            printf ("\n%d  -  %s\n",i+1,ps);
            i++;f++;
         }
      }
      ps = strtok (NULL, " ,.-!");
   }
   if(f==0) printf("\nword %s not found", ss);
   return 0;
}
```

gets(ss);

```
Hello! Very beatiful weather, on this spring is sunny days!

Input word
beautiful

1  -  beatiful

2  -  beatiful

3  -  beatiful

4  -  beatiful

5  -  beatiful

6  -  beatiful

7  -  beatiful

8  -  beatiful

9  -  beatiful
```



Дякую за увагу!!!