DANE TEKSTOWE W JĘZYKU C - TABLICE ZNAKOWE

Stała tekstowa / łańcuchowa jest tablicą znaków zakończoną znakiem o kodzie: **0**

np. stała łańcuchowa: **"Jestem tekstem"**

ASCII →	. . .	**'J'**	**'e'**	**'s'**	**'t'**	**'e'**	**'m'**	**' '**	**'t'**	**'e'**	**'k'**	**'s'**	**'t'**	**'e'**	**'m'**	**'\0'**	**. . .**
wartości→	**. . .**	74	101	115	116	101	109	32	116	101	107	115	116	101	109	0	**. . .**
adresy →	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24

Definicje i inicjalizacje zmiennych „tekstowych":

```
char  tekst[ ] = { 'J', 'e ', 's ', 't ', 'e ', 'm ', ' ', 't ', 'e ', 'k ', 's ', 't ', 'e ', 'm ', '\0' };

char  tekst2[ ] = { "Jestem tekstem" };

char  tekst3[ ] = "Jestem tekstem";

char  tekst4[100] = "Jestem tekstem";

char * tekst5;          // wskaźnik na znak == wskaźnik na początek łańcucha znaków
tekst5 = "Jestem tekstem" ;   // przypisanie zmiennej tekst5 adresu stałej tekstowej

tekst5 = tekst4 ;                          // poprawne przypisanie adresu tablicy

char  tekst6[100];                         // 100-elementowa tablica znakow
tekst6 = "Jestem tekstem" ;                // błędne przypisanie !!!
memcpy( tekst6, "Jestem tekstem", 15 );    // poprawne przypisanie
strcpy ( tekst6, "Jestem tekstem" );       // poprawne przypisanie
```

Przykładowe operacje na pojedynczych literach tekstu:

```
tekst[1] = 'E';                            // zamiana drugiej litery na dużą
tekst[2] = tekst[2] – 32;                  // zamiana trzeciej litery na dużą (?)
tekst[3] = toupper( tekst[3] );            // zamiana czwartej litery na dużą

for( int i=4; i<10; i++)
   tekst[ i ] = toupper( tekst[ i ] );     // zamiana kolejnych sześciu liter

tekst[5] = '\0';                           // skrócenie tekstu do 5 liter

for( int i=0; tekst[ i ] != '\0' ; i++)
   cout << tekst[ i ];                     // wydrukowanie zawartości tekstu (1)

for( char* wsk=tekst; *wsk ; wsk++)
   cout << *wsk;                           // wydrukowanie zawartości tekstu (2)
```

Przykład przetwarzania tekstów → dodanie rozszerzenia „***.txt**” na końcu nazwy pliku

```
#include <iostream.h>
void main(void)
{
   char nazwa[100];
   cout << "Podaj nazwe pliku: ";
   cin.getline(nazwa,100);

   // poszukiwanie ostatniej kropki w łańcuchu
   int i, poz_kropki=-1;
   for(i=0; nazwa[i] != '\0'; i++)
      if( nazwa[i] == '.')
         poz_kropki=i;

   // sprawdzenie obecności rozszerzenia txt
   bool jest_txt=false;
   if(poz_kropki!=-1)
      if(  nazwa[poz_kropki+1]=='t' && nazwa[poz_kropki+2]=='x' &&
           nazwa[poz_kropki+3]=='t' && nazwa[poz_kropki+4]=='\0'  )
         jest_txt=true;

   //jeżeli nie ma rozszerzenia ".txt" to dopisujemy je na końcu nazwy
   if( !jest_txt )
      {
         nazwa[i+0] = '.';          // zmienna 'i' nadal wskazuje koniec nazwy
         nazwa[i+1] = 't';
         nazwa[i+2] = 'x';
         nazwa[i+3] = 't';
         nazwa[i+4] = '\0';
      }

   //    // To samo co powyżej, ale z wykorzystaniem gotowych funkcji  <string.h>
   //    char* poz_kropki=strrchr(nazwa,'.');
   //    bool jest_txt=false;
   //    if( poz_kropki  &&  strcmp( poz_kropki,".txt"  )==0 )
   //       jest_txt=true;
   //    if( !jest_txt )
   //       strcat(nazwa,".txt");


   // wyświetlenie wyniku – nazwy z rozszerzeniem txt na końcu
   cout << endl << endl;
   cout << "Nazwa z rozszerzeniem \"txt\" = [ " << nazwa << "]";

   cout << "Nacisnij ENTER, aby zakonczyc program";
   cin.get();
}
```

Funkcje operujące na łańcuchach znaków <string.h>

Funkcja kopiowania zawartości jednej tablicy znakowej do drugiej (ang. „**str**ing **c**o**py**”).
Prototyp:

char *strcpy(**char** *dest, **const char** *src);

```
// przykładowa implementacja (z wykorzystaniem zapisu „indeksowego”)
char * strcpy( char tekst_wyj[ ], char tekst_wej[ ] )
{
   int i = 0;
   while(   ( tekst_wyj[ i ] = tekst_wej[ i ] ) != '\0'   )
      i++;
   return( tekst_wyj );
}
```

```
// kopiowanie jednego łańcucha do drugiego → wersja wskaźnikowa (1)
char * strcpy( char *tekst_wyj, char *tekst_wej )
{
   char *pocz=tekst_wyj;
   while(   ( *tekst_wyj = *tekst_wej ) != '\0'   )
      {
         tekst_wyj++;
         tekst_wej++;
      }
   return( pocz );
}
```

```
// funkcja kopiująca łańcuchy – wersja wskaźnikowa (2)
char * strcpy( char *tekst_wyj, char *tekst_wej )
{
   char *pocz=tekst_wyj;
   while(  *tekst_wyj++ = *tekst_wej++  ) ;
   return( pocz );
}
```

```
// kopiowanie, z ograniczeniem długości kopiowanego łańcucha
char * strncpy( char tekst_wyj[ ], char tekst_wej[ ], unsigned maks_dlugosc )
{
   int i = 0;
   while(   ( tekst_wyj[ i ] = tekst_wej[ i ] ) != '\0'   &&  i < maks_dlugosc )
      i++;
   return( tekst_wyj );
}
```

Funkcja porównująca teksty: **int** strcmp (**char** ∗tekst_1, **char** ∗tekst_2)
(*ang. „**str**ing **comp**are”*)

funkcja zwraca wartość:

< 0	gdy	tekst_1	< tekst_2
= 0	gdy	tekst_1	== tekst_2
> 0	gdy	tekst_1	> tekst_2

```
int  strcmp( char tekst_1[ ], char tekst_2[ ] )                    // wersja tablicowa
{
   int i = 0;
   while(  tekst_1[ i ] == tekst_2[ i ]  )
      if(  tekst_1[ i++ ] == '\0'  )
         return( 0 );
   return(  tekst_1[ i ] – tekst_2[ i ]  );
}
```

```
int  strcmp( char ∗tekst_1, char ∗tekst_2  )                       // wersja wskaźnikowa (1)
{
   while( ∗tekst_1 == ∗tekst_2 )
      {
         if(  ∗tekst_1 == '\0'  )
            return( 0 );
         tekst_1 = tekst_1 + 1;
         tekst_2 = tekst_2 + 1 ;
      }
   return(  ∗tekst_1 – ∗tekst_2  );
}
```

```
int  strcmp( char ∗tekst_1, char ∗tekst_2  )                       // wersja wskaźnikowa (2)
{
   for(   ; ∗tekst_1 == ∗tekst_2 ; tekst_2++ )
      if(  ! ∗tekst_1++  )
         return( 0 );
   return(  ∗tekst_1 – ∗tekst_2  );
}
```

```
   . . .                                  // przykładowe zastosowanie funkcji strcmp
   char tekst[100];
   cin >> tekst;
   if(  strcmp( tekst , ”Kowalski” )==0 )
      cout << ”Podales tekst: \”Kowalski\” ”;
   . . .
```

Inne wybrane funkcje z biblioteki <**string.h**>

size_t strlen(**const char** *s)

od ang. „ ***str****ing* ***len****gth ”*
Funkcja wyznacza i zwraca długość (ilość znaków) łańcucha **s** (bez znaku ‘\0’)

char *strcat(**char** *dest, **const char** *src)

od ang. „ ***str****ing con****cat****enate ”*
Funkcja dodaje łańcuch **src** (*ang. source*) do łańcucha **dest** (*ang. destination*)
Zwraca wskaźnik na połączony łańcuch (**dest**)

---

char *strchr(**const char** *s, **int** c)

od ang. „ ***str****ing* ***ch****a****r*** *”*
Funkcja szuka pierwszego wystąpienia znaku **c** w podanym łańcuchu **s**
Zwraca wskaźnik na znalezioną pozycję wystąpienia lub adres **NULL**.

char *strrchr(**char** *s, **int** c)

od ang. „ ***str****ing* ***r****ight* ***ch****a****r*** *”*
Funkcja szuka ostatniego wystąpienia znaku **c** w podanym łańcuchu **s**
Zwraca wskaźnik na znalezioną pozycję wystąpienia lub adres **NULL**.

char *strstr(**char** *s, **const char** *sub)

od ang. „ scans ***str****ing for sub****str****ing ”*
Funkcja szuka pierwszego wystąpienia łańcucha **sub** w podanym łańcuchu **s**
Zwraca wskaźnik na znalezioną pozycję wystąpienia lub adres **NULL**.

---

char* strupr(**char** *s)

od ang. „ ***str****ing* ***up****per ”*
Funkcja zamienia zawartość łańcucha **s** na duże litery

char* strlwr(**char** *s)

od ang. „ ***str****ing* ***l****o****w****e****r ”***
Funkcja zamienia zawartość łańcucha **s** na małe litery

Przykłady operacji na łańcuchach znaków

1)
```
#include <stdio.h>                    // przykład zamiany wszystkich liter na duże
#include <ctype.h>

        //  standardowe funkcje zamiany łańcuchów na małe lub duże litery
        //  #include <string.h>  →  char *strlwr(char *s);      char *strupr(char *s);

char *Zamien_Na_Duze( char* tekst )
{
   char *wsk = tekst;
   do
      *wsk = toupper(*wsk );          //  zamiana pojedynczej litery na dużą
   while(*wsk++ );
   return( tekst );
} //------------------------------------------------------------------------ Zamien_Na_Duze

void main( void )
{
   char lancuch_testowy[100] = "abcdefghijklmnopqrstuvwxyz";
   printf( "%s\n" , Zamien_Na_Duze ( lancuch_testowy ) );
}
```

2)
```
#include <stdio.h>                    // przykład zamiany pierwszych liter wyrazów
#include <ctype.h>

char *Slowa_Na_Duze( char* tekst )
{
   char *wsk = tekst;
   if( !*wsk )       // jeżeli tekst pusty to zakończ działanie
      return(tekst);
   *wsk = toupper( *wsk );            // zamiana pierwszej litery
   while( *++wsk )
      if( *(wsk-1) == ' '  )          // jeżeli poprzedzający znak jest spacją
         *wsk = toupper( *wsk );      // zamiana znaku na dużą literę
   return( tekst );
} //------------------------------------------------------------------------ Slowa_Na_Duze

void main( void )
{
   char lancuch[100] = "to  jest     probka  tekstu    ";
   printf( "%s\n" , Slowa_Na_Duze( lancuch ) );
}
```

3)
```
#include <stdio.h>                    // funkcja zamieniająca zadane fragmenty tekstu
#include <string.h>

void Zamien_Fragmenty(  char* tekst,
                        char* stary_wzorzec,
                        char* nowy_wzorzec )
{
   char* wsk = tekst;
   int dlugosc_starego = strlen( stary_wzorzec );
   int dlugosc_nowego = strlen( nowy_wzorzec );
   do {
      wsk = strstr( tekst, stary_wzorzec );
      if( wsk )                      // if( wsk != null )
        {
          // ewentualne zsunięcie lub rozsunięcie tekstu
          memmove(  wsk + dlugosc_nowego ,
                    wsk + dlugosc_starego ,
                    strlen( wsk + dlugosc_starego ) +1 );
          // wpisanie nowego wzorca w przygotowane miejsce
          memcpy( wsk, nowy_wzorzec, dlugosc_nowego);
        }
   } while( wsk );
} //---------------------------------------------------------------------- Zamien_Fragmenty


void main( void )
{
   char tekst[200] = "Ala ma kota a Ola ma Asa";
   printf( "Stary tekst: %s\n" , tekst );
   Zamien_Fragmenty( tekst, "ma", "miala" );
   printf( " Nowy tekst: %s\n" , tekst );          // "Ala miala kota a Ola miala Asa"
}
```

UWAGA !

- *Zastosowanie w powyższym przykładzie funkcji* strcpy *zamiast* memmove *będzie generować błędy (gdy* nowy_wzorzec *będzie dłuższy od* stary_wzorzec*) np.* strcpy(wsk+dlugosc_nowego, wsk+dlugosc_starego);
  utworzy tekst: " Ala ma ko ko ko ko ko ko k"

- *Definicja:* char* tekst = "Ala ma kota a Ola ma Asa";
  jest równoważna: char tekst[24+1] = "Ala ma kota a Ola ma Asa";
  Ponieważ podczas zamiany tekst może się wydłużyć (poprzez wstawienie dłuższych fragmentów), więc zmienna tekst *powinna być tablicą większą niż długość inicjującego tekstu.*