**Wprowadzenie do tablic znaków (łańcuchów) w ANSI C**

Tablice znaków stanowią specjalny rodzaj tablic o budowie ułatwiającej przetwarzanie tekstów. Przetwarzanie obejmuje szereg operacji takich jak tworzenie, modyfikację, przeszukiwanie i konwersje tablicy znakowej na liczbę i liczby na tablicę znakową.

Tablice znakowe mogą być definiowane w sposób następujący:

main ()

{

**char** s1[5];

**char** s2[5 ]="ABCD";

**char** s3[ ]="abcd";

}

Definicje tablic s2 i s3 zawierają też inicjalizację. Tablice znaków zakończone są znakiem ‘\0’. Znak ten umożliwia wykrycie końca tablicy, a tym samym umożliwia traktowanie tablic znaków w nieco inny sposób niż zwykłych tablic.

Przykładowo, tablica s2 ma następującą budowę:

**Budowa tablicy znaków**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 'A’  | 'B'  | 'C'  | 'D'  | '\0'  |

 Definiując tablicę znaków należy przewidzieć jedną komórkę na znak końca tekstu '\0'.

**Stałe łańcuchowe i wskaźniki do znaków**

Stałe łańcuchowe (stałe typu tablica znaków), np. "Tekst", przechowywane są w pamięci jako tablice znaków z ostatnim elementem równym ‘\0’.

Stała "Tekst" jest typu char \*, czyli wskaźnik do znaku. Wskaźnik do znaku można zainicjować stałą łańcuchową:

**char** \**ps*="Tekst"; // sposób 1

Inicjalizacja taka jest równoważna parze instrukcji

**char** \**ps;* // sposób 2

*ps*="Tekst";

W przypadku sposobu 1 do wskaźnika *ps* jest przypisywany wskaźnik do tablicy znaków, a nie do \**ps* ( czyli nie do miejsca wskazywanego przez *ps*).

**Wczytywanie i drukowanie tablic znakowych**

W języku C istnieje specjalny deskryptor formatu %s umożliwiający wczytywanie i drukowanie tablic znakowych przy użyciu funkcji *scanf* i *printf*, istnieją też funkcje biblioteczne realizujące te operacje.

Jeśli zdefiniowano tablicę znakową

**char** *s1*[5];

to można ją wczytać i wydrukować w poniższy sposób:

*printf("\ Podaj lancuch :")*;

*scanf(”%s”,s1);* /// nie stosuje się znaku & przed s2, gdyż s2 jest

//wskaźnikiem.

*printf(”\n Wczytany lancuch to :%s ”, s1);*

Jednak wczytywanie łańcuchów przy użyciu funkcji *scanf* nie pozwala na wczytanie łańcuchów zawierających spacje, gdyż wczytywanie łańcucha jest przerywane po napotkaniu spacji.

Dla takich łańcuchów należy zastosować funkcję *gets* lub też funkcję *fgets*. Funkcja *gets* te ma następujący prototyp:

**char** \**gets*(**char** \**s*);

Opis działania funkcji *gets*:

Funkcja wczytuje znaki ze standardowego wejścia ( *stdin*) aż do momentu napotkania znaku nowej linii (*Enter*). Wczytywane znaki są wyświetlane na ekranie i zapamiętywane począwszy od miejsca wskazywanego przez *s*. Znak nowej linii jest zastępowany w łańcuchu znakiem końca łańcucha '\0'. Ciąg wejściowy może zawierać pewne białe znaki (np. spacje i znaki tabulacji poziomej). Funkcja zwraca *s*, gdy operacja się powiodła, lub *NULL* w przypadku wystąpienia błędu. Długość wczytywanego ciągu znaków nie jest ograniczana, co przy ciągach dłuższych niż to wynika z rozmiaru argumentu funkcji gets, może powstać uszkodzenie sąsiednich obszarów pamięci.

Przykład. Użycie funkcji gets do wczytywania tablicy znakowej.

**char** *lan [20]*;

*printf* ("\*n Podaj lancuch*:");

gets(lan);

Bardziej uniwersalną funkcją służącą do wczytywania łańcuchów jest *fgets*.

Funkcja *fgets* ma następujący prototyp:

**char** *\*fgets(char \*s, int n, FILE \*stream)*;

Funkcja ta czyta do łańcucha *s* znaki ze strumienia wejściowego (pliku) określonego przez wskaźnik *stream*. Funkcja kończy wczytywanie znaków po przeczytaniu *n* - 1 znaków lub też po pojawieniu się znaku nowej linii. Funkcja wpisuje znak nowej linii do łańcucha. Funkcja zwraca łańcuch lub NULL w przypadku pojawienia się końca pliku lub błędu.

Przykład. Wczytywanie tablicy znakowej ze standardowego strumienia wejściowego przy użyciu funkcji *fgets*.

**char** *lan [20];*

*printf ("\n Podaj lancuch:");*

*fgets(lan,15,stdin);*

Do drukowania tablic znakowych można zastosować funkcje *puts* i *fputs*.

Prototyp funkcji *puts* ma postać

int *puts*(**const char** \**s*);

Funkcja wyprowadza łańcuch na wyjście standardowe (*stdout*) i dołącza znak nowej linii. W przypadku pomyślnej realizacji funkcja zwraca wartość nieujemną, w przeciwnym przypadku EOF.

int *fputs*(const **char** \**s*, FILE \**stream)*;

Funkcja zapisuje łańcuch *s* do do strumienia wyjściowego określonego przez wskaźnik *stream.* Funkcja nie zapisuje do pliku znaku '\0'. W przypadku powodzenia operacji funkcja zwraca wartość nieujemną, w przypadku błędu wartość EOF.

**Opis funkcji operujących na łańcuchach znaków**

Poniżej przedstawiono krótkie opisy funkcji związanych z działaniami na tablicach znakowych w języku ANSI C. Poniżej będziemy zamiennie stosować terminy tablica znaków, tekst i łańcuch. Oznaczają one tutaj ten sam typ danych. Format oznacza dopuszczalną postać tablicy znakowej.

**1. Plik stdlib.h**

double **atof**(*const char \*s*)

Przekształca tekst zawarty w *s* na wartość typu *double*, funkcja rozpoznaje wartość zmiennoprzecinkową reprezentowaną przez *s*. Funkcja rozpoznaje również wartość nieskończoną, podczas działania tej funkcji pierwszy nierozpoznany znak kończy jej działanie.

Format: [biały\_znak] [znak] [ddd] [.] [ddd] [eIE[znak]ddd].

*int* **atoi** (*const char* \**s*)

Przekształca tekst zawarty w *s* na wartość typu *int* (wartość całkowitą), funkcja rozpoznaje tabulator i spację, znak oraz ciąg cyfr. Przepełnienie jest niezdefiniowane. Funkcja zwraca przekształconą wartość lub 0.

Format: [biały\_znak] [znak] [ddd].

*long* **atol**(*const char* \*s)

Przekształca tekst zawarty w *s* na wartość typu *long*. Działanie jest podobne do działania funkcji **atoi** z tą różnicą, że wartość zwracana jest typu *long*.

Format: [biały\_znak] [znak] [ddd].

*double* **strtod**(*const char* \**s*, *char* \*\**endptr*)

Przekształca tekst zawarty w *s* na wartość typu *double* (podobnie jak *atof*), jeżeli argument wywołania odpowiadający parametrowi *endptr* jest różny od NULL, funkcja wstawia pod adres wskazany przez ten argument wskaźnik do dalszej części tekstu, która nie została przekształcona. Przepełnienie zwraca HUGE\_VAL z odpowiednim znakiem. W przypadku niedomiaru zwracane jest 0.

Format: [biały\_znak] [znak] [ddd] [.] [ddd] [eIE[znak]ddd].

*long* **strtol** (*const char \*s, char \*\*endptr, int base*)

Przekształca tekst zawarty w *s* na wartość typu *long* (podobnie jak *atol*). Jeżeli argument wywołania odpowiadający parametrowi *endptr* jest różny od NULL, funkcja wstawia pod adres wskazany przez ten argument wskaźnik do dalszej części tekstu, która nie została przekształcona. Jeśli argument odpowiadający parametrowi *base* (podstawa systemu liczbowego) zawiera się w przedziale 2-36, to przekształcenia dokonuje się przy założeniu, że tekst wejściowy zawiera liczbę zapisaną przy tej podstawie. Gdy argument odpowiadający parametrowi *base* równa się 0, wówczas przyjmowana podstawa systemu liczbowego zależy od postaci liczby zapisanej w tekście *s*. Wtedy, jeżeli liczba rozpoczyna się 0, podstawą będzie 8, jeżeli 0x(lub 0X) podstawą będzie 16, a w pozostałych przypadkach 10. Przy nadmiarze zwracana jest wartość LONG\_MAX lub LONG\_MIN (zależnie od znaku).

Format: [biały\_znak] [znak] [0] [x] [ddd]

*unsigned long* **strtoul** (*const char \*s, char \*\*endptr, int base*)

Przekształca tekst zawarty w *s* na wartość typu *unsigned long* (działa podobnie do

funkcji **strtol** ) i w przypadku błędu zwraca wartość ULONG\_MAX.

Format: [biały\_znak] [znak] [0] [x] [ddd]

**2. Plik string.h**

W pliku tym zamieszczono deklaracje dwóch grup funkcji operujących na tablicach znaków. Pierwsza to funkcje, których nazwy rozpoczynają się od **str**, druga to funkcje o nazwach rozpoczynających się od **mem**. Funkcje, których nazwy rozpoczynają się od **str** to:

**strcpy, strncpy, strcat, strncat, strcmp, strncmp, strchr, strrchr, strspn, strcspn, strpbrk, strstr , strlen, strerror, strtok, strlwr, strupr.**

*char* **strcpy** *(char \*s, const char \*ct*)'

Kopiuje zawartość tablicy znaków *ct* do tablicy *s* łącznie ze znakiem '\0'; zwraca *s*. Funkcja nie sprawdza długości tablicy, do której odbywa się kopiowanie, stąd tablica docelowa musi mieć miejsce na co najmniej *n*+ l znaków, gdzie *n* to ilość znaków w tablicy *ct*.

*char* **strncpy** (*char \*s, const char \*ct, size\_t n*)

Kopiuje znaki z tablicy znaków *ct* do tablicy *s,* jednak nie więcej niż *n* znaków. Jeżeli w *ct* jest mniej niż *n* znaków, puste miejsce dopełniane jest znakami \0. Funkcja zwraca *s*.

*char* **strcat** (*char \*s, const char \*ct*)

Dopisuje znaki z tablicy znaków *ct* na koniec tablicy *s*, zwracane jest *s*.

*char* **strncat**(*char \*s, const char \*ct, size\_t n*)

Dołącza co najwyżej *n* znaków z tablicy *ct* do końca tablicy *s*, tablica *s* kończy się znakiem '\0'. Funkcja zwraca *s*.

*char* **strcmp** (*const char \*cs, const char \*ct*)

Porównuje tablice znaków *cs* i *ct*. Zwracana jest:

wartość <0 dla *cs* < *ct*

wartość 0 dla *cs* jest równego *ct*

wartość >0 dla *cs* > *ct*.

Porównywanie rozpoczyna się od pierwszego znaku obu tablic do momentu, gdy znaki będą się różnić lub do osiągnięcia końca jednej z tablic. Kryterium porównania jest wartość kodu ASCII danego znaku.

*char* **strncmp** (*const ch ar \*cs, const char \*ct*)

Funkcja ma podobne działanie jak funkcja poprzednia z tą różnicą, że porównywanych

jest co najwyżej *n* znaków.

*char* \***strchr** (*const char \*cs, int c*)

Zwraca wskaźnik do pierwszego wystąpienia znaku *c* w *cs*. w wypadku nie wystąpienia

tego znaku zwracane jest *NULL*.

*char* \**strrchr* (*const char \*cs, int c)*

Zwraca wskaźnik do ostatniego wystąpienia znaku *c* w tablicy *cs*. Jeśli znak nie występuje, zwracane jest *NULL*.

*size\_t* **strspn** (*const char \*cs, const ch ar \*ct*)

Zwraca długość przedrostka w tablicy *cs* składającego się ze znaków występujących w tablicy *ct*.

*size\_t* **strcspn** (*const char \*cs, const char \*ct*)

Działa podobnie jak poprzednia funkcja z tym, że zwraca długość przedrostka w *cs* składającego się ze znaków nie występujących w *ct*, funkcja przerywa zliczanie, gdy w *cs* napotka znak występujący w *ct*.

*char* \***strpbrk** (*const char \*cs, const char \*ct*)

Zwraca wskaźnik do pierwszego wystąpienia w tablicy *cs* któregokolwiek ze znaków z tablicy *ct*. W przypadku nie znalezienia żadnego znaku zwraca *NULL*.

*char* \***strstr** (*const char \*cs, const char \*ct*)

Zwraca wskaźnik do pierwszego wystąpienia *cs* w tablicy *ct* lub wartość *NULL.*

*size\_t* **strlen** (*const char \*cs*)

Zwraca długość tablicy *cs* bez uwzględnienia końcowego znaku '\0'.

*char* \***strerror** (*size\_t n*)

Zwraca wskaźnik do tekstu komunikatu odpowiadającemu błędowi o numerze *n*. Wywołana bez numeru wypisuje komunikat o ostatnim napotkanym błędzie.

*char* \***strtok** (*char \*s, const ch ar \*ct*)

Wyszukuje w tablicy *s* ciągi znaków przedzielone znakami z tablicy *ct*. Kolejne wywołania tej funkcji dzielą tablicę *s* na ciągi znaków rozdzielone znakami z tablicy *ct*. Funkcja po wywołaniu znajduje pierwszy ciąg znaków nie należących do *ct*. Znak następny zastępowany jest znakiem /0 i zwracany jest wskaźnik do początku tego ciągu. Każde następne wywołanie tej funkcji musi być z argumentem *s* równym *NULL*, wówczas zwracany jest kolejny ciąg znaków, przy czym szukanie rozpoczyna się za końcem poprzedniego ciągu. Zwraca *NULL*, gdy nie znajduje ciągów. Każde następne wywołanie może być z innym argumentem *ct*.

char \***strlwr**(char \*s);

Zamienia w tablicy *s* litery duże na małe. Pozostałe znaki nie są zmieniane. Zwraca wskaźnik do *s*.

char \***strupr**(char \*s);

Zamienia w tablicy *s* litery małe na duże. Pozostałe znaki nie są zmieniane. Zwraca wskaźnik do *s*.

**Funkcje mem**

Funkcje te to **memcpy, memmove, memcmp, memchr, memset**.

*void* \***memcpy** (*void \*s, const void \*ct, size\_t n*)

Kopiuje *n* znaków z obiektu *ct* do obiektu *s* i zwraca *s*.

*void* \***memmove** (*void \*s, const void \*ct, size\_t n*)

Wstawia *n* znaków z obiektu *ct* do obiektu *s* i zwraca *s*. Różni się od **memcpy** tym, że pozostawia w obiekcie *s* jego końcową część, jeżeli obiekt *s* ma większą długość niż obiekt *ct*.

*int* **memcmp** (*void const \*cs, const void \*ct, size\_t n*)

Porównuje początkowe *n* znaków zawartych w obiektach *ct* i *cs*. Zasada porównywania jest taka sama jak dla funkcji **strcmp**.

Zwracana jest:

wartość <0 dla *cs* < ct

wartość 0 dla *cs* równego *ct*

wartość >0 dla *cs* > *ct*. ~

*void* \***memchr** (*void const \*cs, int c, size\_t n*)

Zwraca wskaźnik do pierwszego wystąpienia znaku *c* w obiekcie *cs* lub wartość *NULL*, gdy znaku *c*. Sprawdza tylko *n* początkowych znaków.

void \***memset** (void \*s, int c, size\_t n)

Wstawia znak *c* do początkowych *n* znaków obiektu *s*. Zwracane jest *s*.

// Uzupełnić program o wywołania funkcji z <stdlib.h>

#include <conio.h>

#include <string.h>

#include <stdlib.h>

#include <ctype.h>

#include <stdio.h>

#include <errno.h>

int main (int argc, char\* argv[ ])

{ int i=0,w,dl,ilosc;

char \*lancuch0 = "AAAAAA";

char \*lancuch1 = "BBBBBBB";

char \*lancuch2 = "CCCCCCCAAA";

char \*lancuch3 = "DDDDDDDDDBBB";

char \*lancuch4= "EEEEEEEEEE";

char \*lancuch5="FFFFFFFFF";

char \*string1 = "123CD4567890";

char \*string2 = "123DC8";

char \*lancuch1\_1 = "AAA BBB CCC 123 456";

char \*lancuch2\_2 = " ";

char \*lancuch3\_3 = "AAABBBcCC";

char \*bufor;

char \*wskaznik , znak = 'c';

char \*wsk1 = "aaabbbccc", \*wsk2 = "bbbccc", \*wsk3 = "cccddd";

char kopia[40];

char kopia\_n[40]="abcdefghikl";

char kopia\_1[40];

char kopia\_5[40];

char zmienna1[50],zmienna2[50],zmienna3[50];

printf("\n\n\tDzialanie funkcji strcpy :");

printf("\n\n Wywolanie strcpy(kopia, lancuch0) powoduje skopiowanie tablicy znakowej");

printf("\n lancuch0 do tablicy kopia");

strcpy(kopia, lancuch0);

printf("\n\n Po skopiowaniu lancuch0=%s kopia=%s", lancuch0,kopia);

getch();

clrscr();

printf("\n\n\tDzialanie funkcji strncpy dla kopiowania 5 znakow");

printf("\n\n Wywolanie stnrcpy(kopia, lancuch0) powoduje skopiowanie 5 pierwszych");

printf ("\n znakow z lancuch0 do kopia\_5");

strncpy(kopia\_5, lancuch0,5);

kopia\_5[5]='\0';

printf("\n\n Po skopiowaniu lancuch0 =%s kopia\_5=%s", lancuch0,kopia\_5);

getch();

clrscr();

printf("\n\n\tDzialanie funkcji strcat");

printf("\n\n Funkcja skleja lancuch0 z kopia\_n");

printf ("\n\n Przed sklejeniem kopia\_n=%s lancuch0=%s",kopia\_n, lancuch0) ;

strcat(kopia\_n, lancuch0 );

printf ("\n\n Po sklejeniu kopia\_n=%s", kopia\_n) ;

getch();

clrscr();

printf("\n\n\tDzialanie funkcji strcmp");

printf("\n\n Funkcja porownuje leksykograficznie dwa lancuchy");

printf("\n\n wsk1=%s wsk2= %s",wsk1, wsk2);

w = strcmp(wsk2, wsk1);

if (w > 0)

printf("\n\n wsk2 > wsk1");

if (w==0)

printf ("\n\n wsk2 = wsk1") ;

if (w < 0)

printf("\n\n wsk2 < wsk1");

clrscr();

printf ("\n\n\tDzialanie funkcji strncmp dla 5 znakow");

printf ("\n\n Funkcja porownuje pierwsze 5 znakow swoich argumentow");

printf("\n \n wsk1 = %s wsk2 = %s",wsk1,wsk2);

w = strncmp(wsk2, wsk1, 5);

if (w > 0)

printf("\n wsk2 > wsk1");

if (w==0)

printf ("\n wsk2 = wsk1") ;

if (w < 0)

printf("\n wsk2 < wsk1");

getch();

clrscr();

printf("\n\n\tDzialanie funkcji strchr");

printf("\n\n Funkcja wyszukuje zadany znak w tablicy znakow");

wskaznik = strchr(wsk1, znak);

printf("\n\n Znak %c jest w %s na pozycji: %d", znak, wsk1, wskaznik - wsk1+1);

getch();

clrscr();

printf ("\n\n\tDzialanie funkcji strspn");

printf("\n\n Wywolanie ilosc=strspn (string1,string2)");

printf("\n Funkcja wyznacza ilosc poczatkowych znakow ze lancucha string1");

printf("\n ktore wystepuja w lancuchu string2" );

printf("\n\n Lancuch1=%s Lancuch2-%s ", string1,string2);

ilosc = strspn(string1, string2);

printf("\n\n I1osc poczatkowych znakow z lancucha2 w lancuchu 1 to %d", ilosc);

getch();

clrscr();

printf( "\n\n\tDzialanie funkcji strpbrk ");

printf("\n\n Funkcja poszukuje w jednym lancuchu pierwszego wystapienia");

printf("\n znaku z drugiego lancucha");

printf("\n\n Lancuch1=%s Lancuch2=%s ", lancuch0,lancuch2);

wskaznik =(char \*) strpbrk(lancuch0,lancuch2);

printf ("\n\n Pierwsze wystapienie znaku na pozycji %d", wskaznik-lancuch0);

getch();

clrscr();

printf ("\n\n\tDzialanie funkcji strstr ");

wskaznik = strstr(lancuch1,lancuch3);

printf("\n\n Pierwsze wystapienie tekstu =%s", wskaznik);

clrscr();

printf("\n\n\tDzialanie funkcji strlen ");

dl = strlen(lancuch1);

printf ("\n\n Dlugosc lancucha %d", dl);

clrscr();

printf("\n\n\tDzialanie funkcji strerror \n\n");

errno=ENOFILE ;

bufor = strerror(errno);

printf("\n Komunikat bledu dla nr bledu %d= %s", errno,bufor);

getch();

clrscr();

printf("\n\n\tDzialanie funkcji strtok");

printf("\n\n Wyszukuje w lancuchu1\_1 ciagi znakow rozdzielone znakami z lancucha 2\_2");

printf("\n lancuch1\_1 = %s lancuch2\_2 = spacja",lancuch1\_1);

wskaznik = strtok(lancuch1\_1,lancuch2\_2);

if(wskaznik)

printf ("\n Wywolanie nr 0=%s", wskaznik);

while(wskaznik)

{

wskaznik = strtok (NULL," ");

printf("\n Wywolanie nr %d=%s",++i,wskaznik);

if(i>5) break;

}

getch();

clrscr();

printf("\n\n\tDzialanie funkcji memcpy \n\n");

printf("\n\n Funkcja kopiuje zadana liczbe bajtow z jednej lokalizacji");

printf("\n do drugiej");

memcpy(kopia,lancuch1,strlen(lancuch1+1));

printf("\n\n Skopiowane znaki z tekstu bazowego =%s", kopia);

getch();

clrscr();

printf ("\n\n\tDzialanie funkcji mmmove \n\n");

printf("\n\n Funkcja przenosi (kopiuje) zadana liczbe bajtow z jednej lokalizacji");

printf("\n do drugiej, bloki moga sie pokrywac");

memmove(kopia,lancuch0,strlen(lancuch0)); //bufor = (char \*)

printf ("\n\n Przeniesione znaki z tekstu bazowego=%s", kopia);

getch();

clrscr();

printf ("\n\n\tDzialanie funkcji memcmp ") ;

printf("\n\n Funkcja porownuje zadana liczbe bajtow z dwoch lokalizacji");

printf("\n wsk1 = %s wsk2 = %s",wsk1,wsk2);

w = memcmp(wsk2, wsk1,strlen(wsk2));

if (w > 0)

printf("\n wsk2 > wsk1");

if (w==0)

printf ("\n wsk2 = wsk1") ;

if (w < 0)

printf("\n wsk2 < wsk1");

getch();

clrscr();

printf ("\n\n\tDzialanie funkcji memchr");

printf("\n\n Funkcja poszukuje zadanego znaku w bloku pamieci");

printf("\n zwraca wskaznik do pierwszego wystapienia znaku");

bufor = (char \*)memchr(lancuch3\_3,znak,strlen(lancuch3\_3));

printf ("\n\n Znak %c wystepuje w tekscie bazowym %s na pozycji o indeksie=%d",znak, lancuch3\_3,bufor-lancuch3\_3);

getch();

clrscr();

printf("\n\n\tDzialanie funkcji memset ");

printf("\n\n Funkcja nadaje n pierwszym znakom bloku pamieci zadana wartosc ");

printf ("\n\n Lancuch przed wstawieniem=%s", lancuch3\_3);

bufor = (char \*)memset( (char \*)lancuch3\_3,znak,strlen(lancuch3\_3)-1);

printf ("\n\n Lancuch po wstawieniu=%s", bufor);

getch();

return 0;}