Laboratorium Informatyki
Programowanie w języku C
Ćwiczenie 4

1 Wstęp

Tablice znakowe stosowane są w języku C do przechowywania i przetwarzania ciągów znakowych typu *string*. Narzędzia wspomagające przetwarzanie tablic znakowych dostarcza biblioteka `string.h`.

2 Tablica znakowa

W języku C nie ma osobnego typu dedykowanego dla ciągu znakowego, tak jak w C++ gdzie można korzystać ze zmiennej typu string. W języku C stosuje się tablicę znakową `char[]`.

Aby tablica znakowa była prawidłowo przetwarzana ciąg znakowy w niej przechowywany musi kończyć się znakiem końca ciągu `'\0'`. tablica znakowa podlega tym samym zasadom przy deklaracji i inicjacji co tablice liczbowe.

- `char tablica[30];` - deklaruje tablicę znakową pozwalającą na przechowanie 29 znaków, ostatni 30 znak musi zostać przeznaczony na znak końcowy `'\0'`
- `char tablica[] = {'S','t','r','i','n','g'};` - zostanie utworzona tablica znakowa sześcioelementowa. Mogą pojawić się problemy z przetwarzaniem tablicy ponieważ nie zostanie w niej zapisany znak końca ciągu tekstowego `'\0'`.
- `char tab[] = "String";` - zostanie utworzona tablica znakowa siedmioelementowa, sześć komórek zajmą znaki tekstu, a siódmy zostanie zajęty przez znak końca `'\0'`.
- `char tab[20] = "String";` - zostanie utworzona tablica dwudziestoelementowa. Pierwsze sześć elementów zajmą znaki podane przy inicjalizacji. Pozostałe komórki tablicy zostaną zapełnione znakami końca ciągu tekstowego `'\0'`.
- `char tablica[20] = {'S','t','r','i','n','g'};` - tablica zostanie utworzona w identyczny sposób jak w poprzednim przypadku.

2.1 Odczytywanie danych z tablicy znakowej

Dane z tablicy znakowej odczytuje się analogicznie jak z każdej tablicy za pomocą odpowiedniego indeksowania. Wyświetlanie ciągu tekstowego na ekranie może być realizowane za pomocą funkcji `printf()` i `puts()`. W przypadku pierwszej z funkcji wstawienie tablicy znakowej w edytowany ciąg tekstowy realizowany jest za pomocą znaku formatującego %**s**. W obu przypadkach wystarczy podać nazwę tablicy. W obu przypadkach zostaną wyświetlone wszystkie znaki od początku tablicy aż do natrafienia na znak końca ciągu tekstowego `'\0'`.

Dane z tablicy można wyświetlać także znak po znaku za pomocą odpowiedniej pętli i funkcji `putchar()` i `printf()` z znakiem formatującym %**c**.

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
   char tab[] = "Testowy zestaw danych dla analizy ciagu tekstowego.";
   printf("%s\n",tab);
   puts(tab);
   for(int i=0;i<sizeof(tab);i++)
      putchar(tab[i]);
   putchar('\n');
   for(int i=0;i<sizeof(tab);i++)
      printf("%c",tab[i]);
}
```

2.2 Przypisywanie danych do tablicy znakowej

Dane z bufora wejściowego (odczytywane z klawiatury) przypisuje się do tablicy za pomocą funkcji bufora wejściowego. Funkcja `scanf("%s",tab)` z znakiem formatującym `%s` odczyta z bufora wejściowego wszystkie znaki do momentu natrafienia na znak pusty (spacja, tabulacja) lub znak przejścia do nowego wiersza i przypisze te znaki do tablicy. Ponieważ tablica jest także wskaźnikiem nie należy dodawać znaku `&` przed nazwą tablicy. Nie zaleca się stosowania funkcji `scanf()` do odczytywania ciągów tekstowych, ze względu na pomijanie przy odczycie pustych znaków (spacja, tabulacja) i problemem przy identyfikacji końca danych.

Funkcja `gets(tab)` przypisuje do wskazanej tablicy wszystkie znaki z bufora wejściowego do momentu natrafienia na znak przejścia do nowego wiersza. Funkcja ma też parametr opcjonalny którym możemy ograniczyć liczbę znaków odczytywanych z bufora wejściowego. `gets(tab,50)` odczyta i przypisze do tablicy `tab[]` 50 znaków z bufora wejściowego, o ile wcześniej nie pojawi się znak przejścia do nowego wiersza.

Żadna z tych funkcji nie dodaje na końcu wstawianego to tablicy ciągu znakowego, znaku specjalnego - końca ciągu tekstowego `'\0'`.

Do tablicy znakowej można przypisywać dane element po elemencie, za pomocą pętli i odpowiedniego indeksowania za pomocą funkcji `getchar()` i `scanf(()` ze znakiem formatującym %**c**. Można także użyć dedykowanych funkcji kopiowania lub przenoszenia danych (biblioteka `string.h`).

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
   char tab[50];
   char c;
   int i=0;
   printf("Podaj dane: ");
   gets(tab);
   printf("Weryfikacja: %s\n\n",tab);
   printf("Podaj dane: ");
   do{
      c=getchar();
      tab[i]=c;
      i++;
      tab[i]='\0';
   }while(c!='\n');
   printf("Weryfikacja: %s",tab);
}
```

3 Konwersja ciągu znakowego do liczby

Dane przekazywane w postaci ciągów znakowych często zawierają dane liczbowe. Konieczne więc staje się przekonwertowanie wartości liczbowej z postaci znakowej do właściwego formatu - liczby całkowitej (`int`, `long`) lub rzeczywistej (`float`). W języku C dostępne są dedykowane funkcje realizujące to zadanie:

- `atoi(tablica)` - funkcja konwertuje ciąg znakowy do liczby całkowitej typu `int`,
- `atol(tablica)` - funkcja konwertuje ciąg znakowy do liczby całkowitej typu `long int`,
- `atof(tablica)` - funkcja konwertuje ciąg znakowy do liczby rzeczywistej `float`.

Funkcje dostępne są w bibliotece `<stdlib.h>`. Funkcje próbują Przekonwertować znaki z tablicy do liczby wybranego typu począwszy od pierwszego znaku. Jeżeli pierwszy znak tablicy nie jest cyfrą funkcje zwrócą jako wynik wartość 0. Funkcje są w stanie wykryć liczbę zapisaną w systemie dziesiętnym.

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

```
int main()
{
   char buf[] = "1234.34qwe";
   int x= atoi(buf);
   float y = atof(buf);
   printf("Ciąg znakowy \"%s\" -> liczba (int): %i\n",buf,x);
   printf("Ciąg znakowy \"%s\" -> liczba (float): %.3f\n",buf,y);
}
```

Funkcje `strtol()` i `strtoul()` pozwalają na przekonwertowanie liczby w dowolnym (2-36) systemie liczbowym zapisaną w postaci znakowej do liczby całkowitej.

```
long strtol(char *dane, char **reszta, int baza);
```

- `dane` - tablica w której poszukiwana będzie wartość liczbowa danego typu. Ciąg tekstowy musi zaczynać się od możliwych do przekonwertowania danych.
- `reszta` - elementy ciągu tekstowego `dane` które zostaną po wycięciu przekonwertowanych danych.
- `baza` - podstawa systemu liczbowego 2-36

Funkcje są zazwyczaj stosowane do konwersji zestawów zakodowanych danych liczbowych otrzymanych z transmisji lub odczytanych z pliku danych.

```
#include <stdio.h>
#include <stdlib.h>
void cpy(char*,char*);
int main()
{
   char dane[] = "0xa cc 0Xd34 dcf3 afe23";
   char *reszta;
   while(1){
      long x = strtol( dane, &reszta, 16);
      if(reszta==dane) break;
      cpy(reszta,dane);
      printf("Znaleziono liczbę: %li\n",x);
   }
}
void cpy(char *A, char *B)
{
   int i;
   for(i=0;A[i]!='\0';i++)
      B[i] = A[i];
   B[i] = '\0';
}
```

4 Biblioteka string.h

Bibliotek `<string.h>` dostarcza wielu narzędzi ułatwiających przetwarzanie ciągów tekstowych zapisanych w tablicach znakowych. Funkcje te można podzielić na pięć grup:

- kopiowania,
- łączenia,
- porównania,
- wyszukiwania,
- i inne.

Ostatnią grupę stanowią funkcje niezwiązane z żadną z pozostałych grup jak funkcja określająca liczbę znaków w tablicy `int strlen(char*)`. Funkcja ta zlicza znaki w tablicy aż do natrafienia na znak końca ciągu tekstowego `'\0'`. Drugą funkcją z tej grupy jest funkcja `memset(start,znak,liczba)` ustawiająca określoną liczbę komórek tablicy zadaną wartością (znakiem), począwszy od wskazanej komórki tablicy. Funkcja odwołuje się bezpośrednio do komórek pamięć, wymaga więc odpowiedniego adresowania (wskaźniki).

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
   char dane[] = "Zestaw danych testowych.";
   printf("%s\n",dane);
   memset(dane+7,'*',6);
   printf("%s\n",dane);
}
```

Funkcje kopiowania pozwalają na przeniesienie danych z jednej tablicy znakowej do drugiej. Funkcje te dzielimy na dwie grupy, w pierwszej kopiowanie realizowane jest na komórkach pamięci (`memcpy()` i `memmove()`), a w drugiej na elementach tablic (`strcpy()` i `strncpy()`). We wszystkich funkcjach poza `strcpy()` konieczne jest określenie liczby kopiowanych elementów. Przy funkcjach operujących bezpośrednio ka komórkach pamięci należy podać rozmiar kopiowanej pamięci w bajtach.

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
   char dane[] = "Zestaw danych testowych.";
   char tekst[100];
   memcpy(tekst, dane, 13*sizeof(char));
   //memcpy(tekst, dane, sizeof(dane)); //kopiuje całą tablice
```

```
   printf("%s\n",dane);
   printf("%s\n",tekst);
}
```

Kolejną grupę stanową funkcje łączenia ciągów tekstowych (`strcat()` i `strncat()`). Funkcje te na koniec ciągu z tablicy celu (`'\0'`) dodają zwartość drugiego ciągu (w drugiej z funkcji można określić maksymalną liczbę dodawanych znaków). Ważne jest aby kontrolować ilość wolnych komórek w tablicy celu aby nie przekroczyć jej rozmiaru.

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
   char dane[] = "Zestaw danych testowych.";
   char tekst[100] = "Wstawiono: ";
   strncat(tekst, dane, strlen(tekst)-sizeof(dane)-1);
   printf("%s\n",tekst);
}
```

Funkcje z grupy porównania (`strcmp()`, `strncmp()` i `memcmp()`) analizują dwie tablice lub bloki pamięci i zwracają **0** jeżeli obiekty są takie same lub wartość liczbową różnicy pomiędzy pierwszymi znakami jakimi się różnią.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
int main()
{
   char dane[] = "Morze";
   char tekst[] = "Morski";
   printf("Wynik porównania tablic %i\n",strcmp(dane,tekst));
   for(int i=0;i<sizeof(tekst);i++){
      printf("dane[%i]=%i : %c\t",i,dane[i],dane[i]);
      printf("tekst[%i]=%i : %c\t\n",i,tekst[i],tekst[i]);
   }
}
```

Ostatnią grupę stanowią funkcje wyszukiwania znaku, ciągu znaków lub elementu z zestawu znaków. Funkcje zwracają adres (wskaźnik) na odnaleziony element.

Funkcje wyszukiwania znaku zwracają wskaźnik na pierwsze wykrycie znaku w tablicy. Funkcje występują w dwóch rodzajach, przeszukiwania określonego obszaru pamięci:

```
   char* memchr(char *str, char z, int s);
```

i przeszukiwania tablicy:

```
char* strchr(char *str, char z);
char* strrchr(char *str, char z);
```

W przypadku funkcji odwołującej się do pamięci określa się początek obszaru przeszukania (wskaźnik na tablicę) oraz liczbę bloków do przeszukania (liczba elementów tablicy). Funkcje przeszukują tablicę umożliwiają poszukiwanie znaku od początku lub końca tablicy.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
int main()
{
   char tekst[] = "Przykladowy tekst do analizy funkcji wyszukiwania.";
   char znak = 'a';
   char *ch = 0;
   printf("Analizowany tekst:\"%s\"\n",tekst);
   printf("Samogloske \'a\' znaleziono na pozycjach: ");
   ch = strchr(tekst,znak);
   while (ch!=NULL)
   {
       printf("%i ",ch-tekst+1);
       ch = strchr(ch+1,znak);
   }
}
```

Funkcje wyszukiwania znaku z zestawu działają analogicznie jak powyższe, ale zwracają informacje o położeniu jednego znaku z podanego zestawu. Niestety nie zwracana jest informacja który z znaków z zestawu został odnaleziony.

- `char* strpbrk(char *tab, char *zest);` - Zwraca wskaźnik do odnalezionego znaku
- `int strcspn(char *tab, char *zest);` - Zwraca liczbę znaków od początku ciągu do odnalezionego znaku
- `int strspn(char *tab, char *zest);` - Zwraca liczbę odnalezionych znaków od początku ciągu do pierwszego znaku który nie występuje w zestawie

Funkcja wyszukania ciągu tekstowego zwraca wskaźnik na pierwszy z znaków szukanego ciągu w analizowanej tablicy - `char * strstr(char *str, char *dat);`.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
int main()
{
   char tekst[] = "Przykladowy tekst do analizy funkcji wyszukiwania.";
   char ciag[] = "funkcji";
   printf("Analizowany tekst:\"%s\"\n",tekst);
   char * poz = strstr(tekst,ciag);
   printf("Poszukiwany ciag \"%s\" odnaleziono na pozycji %i."
      ,ciag, poz-tekst+1);
}
```

5 Parametry wejściowe funkcji main()

Funkcja `main()` tak jak każda inna funkcja w programie może przyjmować parametry wejściowe. Jednakże ze względu na jej wyjątkowe zastosowanie, składnia jest w formie standaryzowanej. Nazwy i typy parametrów wejściowych są zastrzeżone.

```
int main(int argc, char **argv)
```

Pierwszy parametr `argv` przechowuje informację o liczbie podanych przy wywołaniu programu parametrów. Drugi parametr `argv` w tablicy znakowej przechowuje poszczególne parametry wywołania. Pod indeksem `0` przechowywana jest nazwa programu. Należy pamiętać, że pomimo parametry liczbowe muszą zostać skonwertowane z postaci ciągu tekstowego do postaci liczbowej (`int` lub `float`) przed użyciem.

```
#include <stdio.h>
#include <stdlib.h>
int main(int argc, char **argv)
{
   if(argc>1){
       for(int i=0;i<argc;i++)
           printf("Parametr[%i]: %s\n",i,argv[i]);
       }
   else
      printf("Nie podano zadnych parametrow wywolania programu.\n");
}
```

Zadania

Na podstawie materiału omówionego w instrukcji napisać programy:

Zad 1. Napisać programy pobierający od użytkownika ciąg tekstowy o dowolnej długości. Program analizuje podany ciąg znakowy i wyświetla informację o:

- liczbie znaków
- liczbie wyrazów
- zestawienie informacji o każdym wyrazie:
  - liczbie znaków
  - liczbie samogłosek
  - liczbie spółgłosek
- liczbie znaków interpunkcyjnych (,.? itd),

Zad. 2 Napisać program losujący `n` liczb całkowitych z przedziału (`A`,`B`). Dane można podać w trakcie działania programu lub jako parametry jego wywołania. Program analizuje podane dane sprawdzając ich kompatybilność z programem (zabezpieczenie przed błędnym wprowadzeniem danych).

Programy napisać wykorzystując dynamiczną alokację pamięci, wydzielając do funkcji poszczególne zadania realizowane przez program.