Pracowania programowania

Wykład 9

Napisy (łańcuchy znaków)

Napisy

Napis - ciąg składający się z co najmniej jednego znaku.

Znaki cudzysłowu nie są częścią łańcucha.

Język C nie posiada typu `string`/łańcuchowego. Wszystkie napisy traktowane są jako tablice typu `char`. Ostatnim znakiem w tablicy jest znak `\0`.

Znak a napis

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

int main()
{
    char a = 'q';
    char b[] = "q";
    return 0;
}
```

strlen a sizeof

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

int main()
{
    char nap1[] = "Hello World";
    char nap2[50] = "Hello World";
    printf("%Iu\n",sizeof nap1);
    printf("%Iu\n",strlen(nap1));
    printf("%Iu\n",sizeof nap2);
    printf("%Iu\n",strlen(nap2));
    return 0;
}
```

Tablica a wskaźnik

```c
#include <stdio.h>
#define NAPIS "jakiś tekst"

int main()
{
    char tab[] = NAPIS;
    const char *wsk = NAPIS;
    printf("adres napisu %p\n", "jakiś tekst");
    printf("adres tab: %p\n", tab);
    printf("adres wsk: %p\n", wsk);
    printf("adres NAPIS-u: %p\n", NAPIS);
    return 0;
}
```

```c
#include <stdio.h>

int main()
{
    char nap1[] = "absddfvjskjf";
    char *nap2 = "oijefj";
    nap1[4] = 'M';
    *(nap1 + 7) = 'M';
    nap2[2]='3'; // czy to zawsze możliwe?
    return 0;
}
```

Kopiowanie napisu

```c
#include <stdio.h>

int main()
{
    char * napis = "ab6sWR";
    char * kopia;
    kopia=napis;
    printf("%s\n",napis);
    printf("%p\n",napis);
    printf("%p\n",&napis);
    printf("%s\n",kopia);
    printf("%p\n",kopia);
    printf("%p\n",&kopia);
    return 0;
}
```

czy można to zrobić notacją tablicową?

Wczytywanie napisów

- `scanf`

https://pl.wikibooks.org/wiki/C/scanf

https://en.cppreference.com/w/c/io/fscanf

- `gets`

https://pl.wikibooks.org/wiki/C/gets

https://en.cppreference.com/w/c/io/gets

- `fgets`

https://pl.wikibooks.org/wiki/C/fgets

https://en.cppreference.com/w/c/io/fgets

Wskaźnik czy tablica?

```c
#include <stdio.h>

int main()
{
    char * slowo;
    scanf("%s",slowo);
    printf("%s\n",slowo);
    return 0;
}
```

Ten kod zawiera błąd, który może prowadzić do nieprzewidywalnego zachowania programu. Program próbuje wczytać ciąg znaków do niezainicjowanego wskaźnika slowo, co jest niepoprawne. Wskaźnik nie ma przydzielonej pamięci, więc program będzie zapisywał dane w losowej lokalizacji pamięci, co może prowadzić do awarii lub naruszenia bezpieczeństwa. Poprawiona wersja powinna zadeklarować

tablicę znaków o określonym rozmiarze lub użyć dynamicznej alokacji pamięci przed próbą zapisania danych.

```c
#include <stdio.h>

int main()
{
    char slowo[20];
    scanf("%s",slowo);
    printf("%s\n",slowo);
    return 0;
}
```

Przykłas z `gets`

Program niżej wczytuje tekst (maksymalnie 4 znaki + znak końca) do tablicy ‘slowo’, a następnie wyświetla go dwukrotnie - raz funkcją printf(), a raz puts(). Uwaga: funkcja gets() jest niebezpieczna, ponieważ nie kontroluje długości wprowadzanych danych, co może prowadzić do przepełnienia bufora.

```c
#include <stdio.h>

int main()
{
    char slowo[5];
    gets(slowo);
    printf("%s\n",slowo);
    puts(slowo);
    return 0;
}
```

```c
#include <stdio.h>

int main()
{
    char slowo[5];
    gets_s(slowo, 4*sizeof(char));
    printf("%s\n", slowo);
    puts(slowo);
    return 0;
}
```

- nie działa w każdej konfiguracji

Przykład `fgets`

Program wczytuje tekst do tablicy `slowo[5]` używając bezpiecznej funkcji `fgets()` z limitem 5 znaków. Następnie wyświetla wczytany tekst na trzy różne sposoby: `printf()` z dodatkowym znakiem nowej linii, `puts()` (automatycznie dodaje nową linię) oraz `fputs()` (bez dodawania nowej linii). Wszystkie funkcje we/wy są bezpieczne i kontrolują granice bufora.

```c
#include <stdio.h>

int main()
{
    char slowo[5];
    fgets(slowo,5,stdin);
    printf("%s\n",slowo);
    puts(slowo);
    fputs(slowo,stdout);
```

Różnice?

- `scanf` - do znaku niedrukowanego, reszta do końca linii
- `gets` - mało bezpieczna przy przepełnieniu
- `fgets` - dodaje koniec linii na końcu napisu

Wyświetlanie napisów

- `printf`

https://pl.wikibooks.org/wiki/C/printf

https://en.cppreference.com/w/c/io/fprintf

- `puts`

https://pl.wikibooks.org/wiki/C/puts

https://en.cppreference.com/w/c/io/puts

- `fputs`

https://pl.wikibooks.org/wiki/C/fputs

https://en.cppreference.com/w/c/io/fputs

```c
#include <stdio.h>

int main()
{
    char tekst1[]="abc";
    char tekst2[]= {'a','b','c'};
    char tekst3[]="xyz";
    puts(tekst1);
    puts(tekst2);
    puts(tekst3);
    return 0;
}
```

```c
#include <stdio.h>

int main()
{
    char tekst1[]="abc";
    char tekst2[]= {'a','b','c'};
    char tekst3[]="xyz";
    fputs(tekst1,stdout);
    fputs(tekst2,stdout);
    fputs(tekst3,stdout);
    return 0;
}
```

```c
#include <stdio.h>

int main()
{
    char tekst1[]="abc";
    char tekst2[]= {'a','b','c'};
    char tekst3[]="xyz";
    printf("%s",tekst1);
    printf("%s",tekst2);
    printf("%s",tekst3);
    return 0;
}
```

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

int main()
{
    char buffer[20];
    int a=5;
    int b=7;
    sprintf(buffer,"%5d+%5d=%5d",a,b,a+b);
    printf("%s",buffer);
    return 0;
}
```

20

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

int main()
{
    char buffer[20];
    int a=5;
    int b=7;
    snprintf(buffer,20*sizeof(char),"%5d+%5d=%5d",a,b,a+b);
    printf("%s",buffer);
    return 0;
}
```

Formaty - p.1.

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

int main(){
    // Declare and initialize a float variable with value 4.5
    float b=4.5;

    // %a - Print float in hexadecimal notation with lowercase letters (0x1.2p+2 f
    printf("%a\n",b);

    // %A - Print float in hexadecimal notation with uppercase letters (0X1.2P+2 f
    printf("%A\n",b);

    // Declare and initialize an integer with value 87
    int a=87;
    // Declare and initialize a character with value 'r'
    char c='r';

    // %c - Print as characters: 87 will be converted to its ASCII equivalent 'W',
```

Formaty - p.2.

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

int main(){
    int a=442;

    // Basic decimal integer format (%d) - prints the number as is
    printf("%d\n",a);

    // Space flag (% d) - adds a space before positive numbers (for alignment with
    printf("% d\n",a);

    // Width specification (%2d) - minimum field width of 2 characters (right-alig
    // Since 442 is 3 digits, the width has no effect here
    printf("%2d\n",a);

    // Width specification (%7d) - minimum field width of 7 characters (right-alig
    // Will pad with spaces on the left to reach 7 characters total
```

Formaty - p.3.

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

int main(){
    double a=1234.8912;

    // Basic floating-point format (%f) - displays with 6 decimal places by defaul
    // Output: 1234.891200 (note the automatic padding to 6 decimal places)
    printf("%f\n",a);

    // Long float format (%lf) - functionally identical to %f for printf
    // (The 'l' modifier is actually unnecessary for printf, though required for s
    // Output: 1234.891200 (same as %f)
    printf("%lf\n",a);

    // Width and precision (%5.2f) - minimum width of 5 characters, exactly 2 deci
    // Width 5 is too small for "1234.89" (7 chars), so it expands as needed
    // Output: 1234.89 (value rounded to 2 decimal places)
```

Formaty - p.4.

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

int main(){
    int a=123;

    // %x format specifier - prints the integer in lowercase hexadecimal notation
    // The decimal value 123 is converted to hexadecimal, which is 7b
    // This format is commonly used for displaying memory addresses or color value
    printf("%x\n",a);

    // %X format specifier - prints the integer in UPPERCASE hexadecimal notation
    // Same as %x but uses capital letters (A-F instead of a-f) for hex digits
    // The decimal value 123 is displayed as 7B instead of 7b
    printf("%X\n",a);

    // %#x format specifier - the # is the "alternate form" flag
    // For hexadecimal format, this adds the "0x" prefix to indicate hexadecimal n
```

Formaty - p.5.

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

int main(){
    char tekst[15]="informatyka";

    // Basic string format (%s) - displays the entire string as is
    // Output: "informatyka"
    // This is the simplest way to print a string with no special formatting
    printf("%s\n",tekst);

    // Width-specified string format (%20s) - right-aligned within a field of 20 c
    // Output: "         informatyka"
    // The string "informatyka" is 11 characters long, so 9 spaces are added befor
    // to fill the 20-character field width, creating right alignment
    printf("%20s\n",tekst);

    // Left-aligned string format (%-20s) - left-aligned within a field of 20 char
```

Typ `wchar_t`

https://en.wikibooks.org/wiki/C_Programming/wchar.h

https://en.cppreference.com/w/c/language/string_literal

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

int main()
{
    wchar_t buffer[20];
    fgetws(buffer,20,stdin);
    fputws(buffer,stdout);
    return 0;
}
```

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

int main()
{
    wchar_t buffer[20];
    wscanf(L"%s",buffer);
    wprintf(L"%s",buffer);
    return 0;
}
```

Polskie znaki

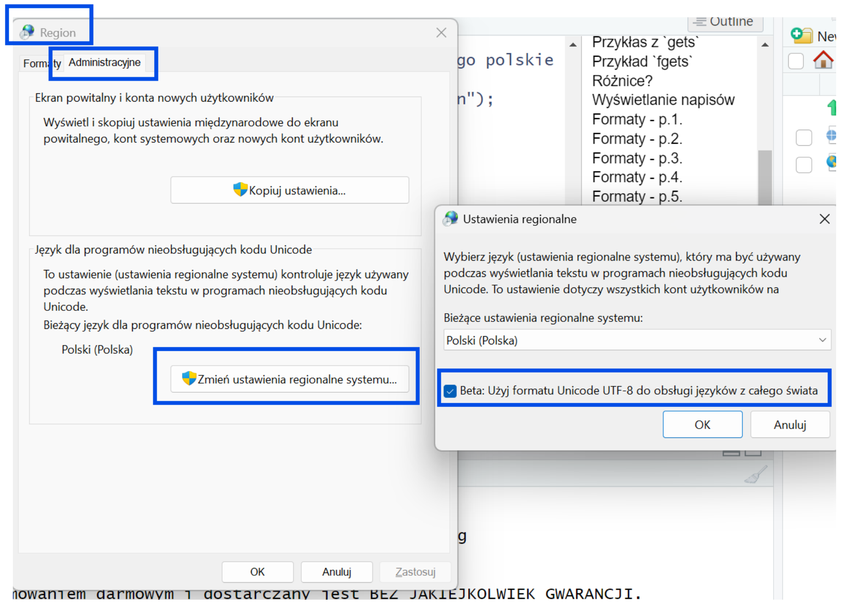
```c
#include <stdio.h>
#include <stdlib.h>
#include <io.h>      // dla _setmode
#include <fcntl.h>   // dla _O_U8TEXT
#include <locale.h>

int main(void) {
    // Ustawienie lokalizacji na polską (UTF-8)
    setlocale(LC_ALL, "pl_PL.UTF-8");

    // Przełączenie standardowych strumieni na tryb UTF-8
    _setmode(_fileno(stdout), _O_U8TEXT);
    _setmode(_fileno(stdin),  _O_U8TEXT);

    // Wyświetlenie przykładowego tekstu zawierającego polskie znaki
    wprintf(L"Witaj świecie! Zażółć gęślą jaźń.ęśół\n");

    // Pobranie danych od użytkownika
```


Region
Formaty
Administracyjne
Ekran powitalny i konta nowych użytkowników
Wyświetl i skopiuj ustawienia międzynarodowe do ekranu powitalnego, kont systemowych oraz nowych kont użytkowników.
Kopiuj ustawienia...
Język dla programów nieobsługujących kodu Unicode
To ustawienie (ustawienia regionalne systemu) kontroluje język używany podczas wyświetlania tekstu w programach nieobsługujących kodu Unicode.
Bieżący język dla programów nieobsługujących kodu Unicode:
Polski (Polska)
Zmień ustawienia regionalne systemu...
OK
Anuluj
Zastosuj
Ustawienia regionalne
Wybierz język (ustawienia regionalne systemu), który ma być używany podczas wyświetlania tekstu w programach nieobsługujących kodu Unicode. To ustawienie dotyczy wszystkich kont użytkowników na
Bieżące ustawienia regionalne systemu:
Polski (Polska)
Beta: Użyj formatu Unicode UTF-8 do obsługi języków z całego świata
OK
Anuluj
Outline
Przykłas z `gets`
Przykład `fgets`
Różnice?
Wyświetlanie napisów
Formaty - p.1.
Formaty - p.2.
Formaty - p.3.
Formaty - p.4.
Formaty - p.5.

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

int main()
{
    wchar_t buffer[20];
    int a=3;
    int b=4;
    swprintf(buffer,20*sizeof(wchar_t),L"%d+%d=%d",a,b,a+b);
    wprintf(L"%s",buffer); // na linuxie %ls
    return 0;
}
```

char	wchar_t
print	wprintf
scanf	wscnaf
"abc"	L"abc"
'k'	L'k'
%s	%ls

Napisy a funkcje

```
int dlugosc(char*napis)
{
    int temp=0;
    while(*(napis++))
    {
        temp++;
    }
    return temp;
}
```

```
int dlugosc2(char napis[])
{
    int temp=0;
    for(int i=0;napis[i]!='\0';i++)
    {
        temp++;
    }
    return temp;
}
```

czy to możliwe?

```
1 void foo(const char*napis)
2 {
3     *napis='a';
4 }
```

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

char* foo()
{
    return "abc";
}

int main()
{
    printf("%s\n",foo());
    return 0;
}
```

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

char* foo()
{
    char * temp=(char*)malloc(sizeof(char)*10);
    temp[0]='w';
    temp[1]='$';
    temp[2]='a';
    temp[3]='\0';
    return temp;
}

int main()
{
    printf("%s\n",foo());
    return 0;
}
```

Tak nie robimy (!)

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

char* foo()
{
    char temp[10];
    temp[0]='w';
    temp[1]='$';
    temp[2]='a';
    temp[3]='\0';
    return temp;
}

int main()
{
    printf("%s\n",foo());
    return 0;
}
```

Podsumowanie

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

int main()
{
    char tekst1[10]="abcde";
    printf("%Iu\n",sizeof(tekst1));
    printf("%p\n",tekst1);
    printf("%p\n",&tekst1);
    //tekst1="eee";
    //tekst1++;
    tekst1[2]='R';
    printf("%s\n",tekst1);
    return 0;
}
```

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

int main()
{
    char *tekst2="abcde";
    printf("%Iu\n",sizeof(tekst2));
    printf("%p\n",tekst2);
    printf("%p\n",&tekst2);
    tekst2="WERT";
    printf("%s\n",tekst2);
    tekst2++;
    //tekst2[2]='R';
    printf("%s\n",tekst2);
    return 0;
}
```

Funckje znakowe i łańuchowe

https://en.cppreference.com/w/c/string/byte

https://en.cppreference.com/w/c/string/wide

Porządek leksykograficzny

Mądra definicja z wikipedii:

Relację leksykograficzną $\preccurlyeq$ między ciągami $\alpha, \beta \in X^*$ ustala się następująco:

- jeśli istnieje wskaźnik j taki, że $\alpha(j) \neq \beta(j)$, to znajdujemy najmniejszy i o tej własności. Wówczas
  - $\alpha \preccurlyeq \beta$ gdy $\alpha(i) \preccurlyeq \beta(i)$ lub $\beta \preccurlyeq \alpha$ gdy $\beta(i) \preccurlyeq \alpha(i)$ (tzn. relacja między ciągami jest zgodna z relacją między odpowiednimi elementami)
- jeśli taki j nie istnieje, to
  - jeśli oba są skończone i tej samej długości, to $\alpha = \beta$
  - jeśli oba ciągi są nieskończone, to $\alpha = \beta$
  - jeśli są różnej długość np. β jest dłuższy od α (w szczególności β może być nieskończony), to $\alpha \preccurlyeq \beta$

Co do nauki?

- podstawowe znaczniki formatowania
- typ `char` i `wchar_t`
- instrukcje wejścia/wyjścia sformatowanego
- implementacja funkcji znakowych i napisowych bez użycia funkcji bibliotecznych
- porządek leksykograficzny
- właściwy sens “wycinania”
- ogólne rozeznanie tablicy znaków ASCII (pdf będzie na egzaminie)

Zamiana wybranej grupy znaków

Napisz funkcję `toLowerNew`, która przyjmuje jako argument tablicę znaków typu `char` i zamienia w niej wszystkie duże litery na małe. Nie korzystaj ze wbudowanych funkcji języka oprócz operacji wejścia/wyjścia. Stwórz przypadek testowy dla tej funkcji.

Zamiana wybranej grupy znaków - inaczej

Napisz funkcję `toLowerNew`, która przyjmuje jako argument wskaźnik do napisu typu `const char*` i zwraca wskaźnik do nowego napisu, w którym wszystkie duże litery zostały zamienione na małe. Oryginalna tablica znaków pozostaje niezmieniona. Pamiętaj o alokacji pamięci dla nowego napisu. Nie korzystaj ze wbudowanych funkcji języka oprócz operacji wejścia/wyjścia i funkcji alokacji pamięci. Stwórz przypadek testowy dla tej funkcji.

Zadanie - porządek

Napisz funkcję `lexComp` przyjmującą dwa argumenty typu `char[]` (tablice znaków) i zwracającą 1, jeśli pierwszy napis jest później w porządku leksykograficznym niż drugi, oraz 0 w pozostałych przypadkach. Nie korzystaj ze wbudowanych funkcji języka oprócz operacji wejścia/wyjścia. Stwórz przypadek testowy dla tej funkcji.

```c
#include <stdio.h>

// Funkcja porównująca dwa napisy leksykograficznie
int lexComp(char str1[], char str2[]) {
    int i = 0;
    while (str1[i] != '\0' && str2[i] != '\0') {
        if (str1[i] > str2[i]) {
            return 1; // str1 jest później w porządku leksykograficznym niż str2
        }
        if (str1[i] < str2[i]) {
            return 0; // str1 jest wcześniej w porządku leksykograficznym niż str2
```

```
        }
        i++;
    }

    // Jeśli wszystkie odpowiadające sobie znaki są takie same, ale napisy mają ró
    if (str1[i] == '\0' && str2[i] != '\0') {
        return 0; // str1 jest krótszy, więc wcześniej leksykograficznie
```

Zadanie “przepisanie”

Napisz funkcję `strCopyNew`, która otrzymuje dwa argumenty typu `char[]` (tablice znaków): źródłową i docelową. Funkcja przepisuje napis znajdujący się w tablicy źródłowej do tablicy docelowej. Zakładamy, że w tablicy docelowej jest wystarczająco dużo miejsca. Nie korzystaj ze wbudowanych funkcji języka oprócz operacji wejścia/wyjścia. Stwórz przypadek testowy dla tej funkcji.

```c
#include <stdio.h>

void strCopyNew(char source[], char destination[]) {
    int i = 0;
    while (source[i] != '\0') {
        destination[i] = source[i];
        i++;
    }
    destination[i] = '\0'; // Dodaj znak końca napisu do tablicy docelowej
```

```
}

int main() {
    char src[] = "Testowy napis";
    char dest[100]; // Zakładamy, że jest wystarczająco dużo miejsca

    strCopyNew(src, dest);

    printf("zrodlo: %s\n", src);
```

Zadanie “wycinanie”

Napisz funkcję `rmVowel`, która przyjmuje jako argument napis i usuwa z niego wszystkie znaki będące samogłoskami. Nie korzystaj ze wbudowanych funkcji języka oprócz operacji wejścia/wyjścia. Stwórz przypadek testowy dla tej funkcji.

```c
#include <stdio.h>

int isVowel(char c) {
    char vowels[] = "aeiouAEIOU";
    for (int i = 0; vowels[i] != '\0'; i++) {
        if (c == vowels[i]) {
            return 1;
        }
    }
    return 0;
}

void rmVowel(char txt[]) {
    int readIndex = 0, writeIndex = 0;
```

```
    while (txt[readIndex] != '\0') {
        if (!isVowel(txt[readIndex])) {
            txt[writeIndex++] = txt[readIndex];
```

Bibliografia

- Richard Reese, Wskaźniki w języku C, Wydawnictwo Helion 2014.
- https://pl.wikibooks.org/wiki/C/Wska%C5%BAniki, dostęp online 15.03.2020.
- http://wazniak.mimuw.edu.pl/index.php?title=Wst%C4%99p_do_programowania_w_j%C4%99zyku_C/Wska%C5%BAniki, dostęp online 15.03.2020.
- https://pl.wikibooks.org/wiki/C/Wska%C5%BAniki_-_wi%C4%99cej, dostęp online 15.03.2020.
- Stephen Prata, Język C. Szkoła programowania. Wydanie VI, Wyd. Helion, 2016.
- https://pl.wikibooks.org/wiki/C/Tablice, dostęp online 20.03.2020.
- https://pl.wikibooks.org/wiki/C/Tablice_-_wi%C4%99cej, dostęp online 20.03.2020.