Strukturalne typy danych - łańcuchy cz.1 [1]

8 września 2021

Plan prezentacji

Bibioteka standardowa C++

Język C++

został stworzony w latach osiemdziesiątych XX wieku (pierwsza wersja pojawiła się w 1979 r.) przez Bjarne Stroustrupa jako obiektowe rozszerzenie języka C.

Standardy języka C++

Rok	Standard C++	Nazwa nieoficjalna
1998	ISO/IEC 14882:1998	C++98
2003	ISO/IEC 14882:2003	C++03
2011	ISO/IEC 14882:2011	C++11, C++0x
2014	ISO/IEC 14882:2014	C++14, C++1y
2017	ISO/IEC 14882:2017	C++17, C++1z
2020	ISO/IEC 14882:2020	C++20, C++2a

Bibioteka standardowa C++

Biblioteka standardowa w C++

nie jest tak bogata jak w innych językach, gdyż nie uwzględnia takich zagadnień jak programowanie sieciowe, graficzne, wielowątkowe, bazodanowe itd. Skupia się głównie na implementacji kolekcji (kontenerów) i algorytmów na tych kolekcjach operujących. Ta **część biblioteki standardowej** nazywana jest, ze względów historycznych, **standardową biblioteką wzorców (szablonów)**, czyli **STL (od ang. Standard Template Library)**.

Bibioteka standardowa C++

Podstawowe składniki biblioteki:

- łańcuch znaków
- realizacja funkcji wejścia-wyjścia
- kontenery (struktury danych)
- algorytmy
- wspomaganie operacji numerycznych
- wsparcie dla międzynarodowych wersji programów

Bibioteka standardowa C++

Biblioteka podzielona jest na wiele części:

- aby z nich korzystać, w tworzonym programie wystarczy użyć odpowiednich dyrektyw **#include** włączających pliki nagłówkowe opisujące funkcjonalność poszczególnych modułów biblioteki
- implementacja funkcji, klas itd. z biblioteki standardowej, w postaci plików binarnych, dostarczana jest przez producenta kompilatora C++ (można również korzystać z innych, niezależnych implementacji, zarówno komercyjnych jak i typu open source).

Bibioteka standardowa C++

Biblioteka standardowa języka C w bibliotece języka C++ (wybrane pliki):

```
#include <cstdlib>
#include <cstdio>
#include <cstring>
#include <ctime>
```

Biblioteka standardowa języka C++ (wybrane pliki):

```
#include <string>
#include <new>
#include <vector>
#include <complex>
```

Łańcuch w C/C++

Definicja w C/C++

- **łańcuch/ ciąg znaków** - to tablica znaków zakończona znakiem końca tekstu '\∅'. Znak ten jest wartością ostatniego elementu łańcucha o numerze równym jego rozmiarowi.
- **char** znak;
- znaki przechowywane są jako liczby - **kod ASCII**
- **'\n' - LF Line Feed** - nowa linia
- **'\r' – CR (Carriage Return)** - powrót karetki
- '\∅' - nul

Przypisanie wartości C/C++

```
char znak;
znak='A';
```

Łańcuch w C/C++

Deklaracja tablicy znaków

char identyfikator [rozmiar];

- znaki zawarte w tekście są numerowane od zera
- zmienna **identyfikator [rozmiar]** zawiera znak końca tekstu '\∅'

Przykład inicjalizacji zmiennej łańcuchowej

```
char s[10]={"telefon"};
char s[10]="telefon";
char s[10]={'t','e','l','e','f','o','n','\0'};
char s[]={"telefon"};
char s[]="telefon"; <=> char s[8]="telefon";
```

Łańcuch w C/C++

Elementy tablicy

s[0]='t'
s[1]='e'
s[2]='l'
s[3]='e'
s[4]='f'
s[5]='o'
s[6]='n'
s[7]='\∅'
Pozostałe wartości s[8] i s[9] są wyzerowane

Przypisanie wartości łańcuchowi C/C++

Łańcuchy mogą być wskaźnikami

Jak kompilator traktuje łańcuch?:

- zlicza ile jest znaków w łańcuchu
- rezerwuje w pamięci rozmiar łańcucha + 1
- kopiuje cały łańcuch do zarezerwowanej pamięci
- **kompilator traktuje łańcuch jak wskaźnik do zarezerowanej pamięci**

WAŻNE!

Łańcuch jest traktowany jako sekwencja znaków zakończona **nul**, lecz zachowuje się jak wskaźnik typu **char***

Przypisanie wartości łańcuchowi C/C++

Nie można !

```
char lancuch[10];
lancuch="telefon";
```

Dlaczego ?
Kompilator po lewej stronie widzi **tablicę znaków**, a po drugiej stronie łańcuch typu **char***

Jak przypisać ?

Korzystamy z funkcji **strcpy**
char * strcpy (char * destination, const char * source);

const char *

wskaźnik do stałego/niezmiennego znaku - nie możemy zmienić wartości na którą wskazuje wskaźnik, ale możemy zmienić wartość samego wskaźnika

Łańcuchy C/C++ - podsumowanie

Deklaracja 1

```
char postac[10]="Nowak";
```

- tablica 10 znaków
- następujące znaki 'N','o','w','a','k','\∅' są zapisane w tablicy
- każde użycie **postac** jest interpretowane jako wskaźnik do pierwszego elemetu tablicy - czyli **'N'**

Na co trzeba uważać używając funkcji **strcpy** ?

```
strcpy(postac,"Brzeczyszczykiewicz");
```

UWAGA !

funkcja **strcpy** nie sprawdza , czy łańcuch mieści się w tablicy !

Łańcuchy C/C++ - podsumowanie

Deklaracja 2

```
char *bohater="Jez";
```

- w pamięci kompilator rezerwuje 4 bajty (3 znaki + 1 znak pusty) i wypełnia znakami 'J','e','z','\∅'
- kompilator tworzy zmienną bohater typu **char***;
- kompilator przypisuje wskaźnik do zarezerwowanego łańcucha **Jez**

```
bohater="Wolodyjowski";
```

Łańcuchy C/C++ - wyświetlanie łańcucha

Funkcja **printf** C/C++

- **int printf(char* format,...);**
- pobiera różną ilość parametrów
- dodaje znak nowej linii na końcu
- potrzebna bibliotek *#include < cstdio >*

```
printf("%s i %s sa kolegami\n",postac,bohater);
```

Łańcuchy C/C++ - pobieranie łańcucha

Funkcja **scanf** C/C++

- **int scanf(const char *format, ...);**
- pobiera różną ilość parametrów
- wczytuje ze standardowego strumienia wejściowego ciąg znaków (łańcuch) ,formatuje go zgodnie z kodami podanymi w format i przypisuje do zmiennych, których adresy przekazujemy w liście
- potrzebna bibliotek *#include < cstdio >*

```
char imie[20];
printf("Podaj imie: ");
scanf("%s", imie);
printf("Twoje imie to %s.", imie);
```

Łańcuchy C/C++ - wczytanie i wyświetlenie linii

Funkcja **puts** i **fgets** C/C++

- **int puts(conts char* str);** - pobiera tylko jeden argument, wskaźnik do łańcucha i dodaje znak nowej linii na końcu (można ignorować zwracaną wartość int)
- **char * fgets(char * str, int num, FILE * stream);** - wczytuje tekst ze wskazanego strumienia aż do napotkania znaku przejścia do nowej linii lub do wczytania **num-1** znaków
- potrzebna bibliotek *#include < cstdio >*

Łańcuchy C/C++ - wczytanie i wyświetlenie linii

Funkcja **puts** i **fgets** C/C++

```
    puts(bohater);
    puts(postac);
    puts("Lewandowski");
    char imie[30];
    printf("Podaj imie: ");
    fgets(imie, sizeof(imie), stdin);
// read string
    printf("Imie: ");
    puts(imie);     // display string
```

Przydatne funkcje w C/C++

Funkcja **strcpy** C/C++

- **char * strcpy (char * s1, const char * s2);**
- kopiuje łańcuch s2 do łańcucha s1 (znak końca łańcucha s2 również jest kopiowany)
- wartością funkcji jest s1
- potrzebna bibliotek *#include < cstring >*

```
char   s[]="abcdef";
strcpy(s,"nowy");
\\wynik:  s="nowy"
```

Przydatne funkcje w C/C++

Funkcja **strncpy** C/C++

- **char * strncpy (char * s1, const char * s2, int n);**
- kopiuje n znaków z łańcucha s2 do łańcucha s1
- wartością funkcji jest s1
- potrzebna bibliotek *#include < cstring >*

```
char   s[]="abcdef";
strncpy(s,"nowy",3);
\\wynik:  s="nowdef"
```

Przydatne funkcje w C/C++

Funkcja **strcat** C/C++

- **char * strcat (char * s1, const char * s2);**
- dołączanie łańcucha s2 na koniec łańcucha s1
- wartością funkcji jest łańcuch utworzony w wyniku połączenia
- potrzebna bibliotek *#include < cstring >*

```
char    s[]="abcdef";
strcat(s,"nowy");
\\wynik:  s="abcdefnowy"
```

Przydatne funkcje w C/C++

Funkcja **strlen** C/C++

- **int strlen (const char * s);**
- wyznacza liczbę znaków łańcucha s
- potrzebna bibliotek *#include < cstring >*

```
char   s[]="abcdef";
int  a=strlen(s);
\\wynik:  a=6  ;
```

Przydatne funkcje w C/C++

Funkcja **strcmp** C/C++

- **int strcmp (const char * s1,const char *s2);**
- porównuje łańcuchy s1 i s2:
  - jeśli s1<s2 (czyli s1 jest alfabetycznie przed s2), to wartość funkcji jest mniejsza od 0
  - jeśli s1=s2 - to wartość jest równa 0
  - jeśli s1>s2 (czyli s1 jest alfabetycznie po s2), to wartość funkcji jest większa od 0
- potrzebna bibliotek *#include < cstring >*

```
char   s[]="abcdef";
int  a=strcmp(s,"abcdef");
\\wynik:  a=0  ;
```

Bibliografia

Korman, D., and Szabłowicz-Zawadzka, G.
Informatyka Europejczyka. Podręcznik dla szkół ponadpodstawowych. Zakres rozszerzony. Część 1.
Helion, Gliwice, 2020.

Zakończenie prezentacji

DZIĘKUJĘ ZA UWAGĘ !