Podstawy programowania

Laboratorium 11.

Cel:

- Opanowanie podstaw tworzenia i wykorzystania tablic struktur i wskaźników w C

Uwaga 1: **w każdej z dokonywanych modyfikacji kodu wprowadzenie wywołania funkcji *malloc* musi być powiązane z wprowadzeniem we właściwym miejscu wywołania funkcji *free.***

Uwaga 2: **dla każdej z rozważanych struktur danych warto, dla lepszego zrozumienia działania programu, naszkicować sobie sposób przechowywania danych w pamięci: komórki w ramkach stosu dla występujących w kodzie funkcji, zwłaszcza komórki na wskaźniki (adresy), i śledzić przesyłanie argumentów w trakcie działania programu**

Zajęcia:

1. Utworzenie katalogu roboczego *lab_11*
2. Skopiowanie za strony przedmiotu pliku *tekst_wieloliniowy.c* do nowego podkatalogu np. *napisy*
3. Analiza programu do wczytywania linii z *stdin*: funkcji *main* oraz funkcji *wczytaj_linie*, a następnie uruchomienie programu i przetestowanie jego działania
   - program wczytuje kilka linii ze standardowego wejścia posługując się funkcją *wczytaj_linie* o prototypie:
     int wczytaj_linie(char linia, int max_dlugosc_linii);*
   - definicja funkcji wraz z przykładowym opisem i kontraktem znajduje się na końcu pliku
   - w funkcji *main* sprawdzana jest poprawność działania programu poprzez wypisanie na ekranie wczytanych linii (jako tablicy znaków i jako napisu)
   - w dostarczonej, początkowej wersji programu, tekst przechowywany jest w dwuwymiarowej tablicy znaków o definicji:
     char tekst_wieloliniowy[MAX_N_L][MAX_DL_L];
   - gdzie: maksymalna liczba linii *MAX_N_L* i maksymalna długość pojedynczej linii *MAX_DL_L* są stałymi parametrami zadanymi poprzez *#define*
4. Modyfikacja programu polegająca na wykorzystaniu do przechowywania tekstu struktury danych tablicy wskaźników:
   char tekst_wieloliniowy_tab_wsk[MAX_N_L];*
   - w pierwszej wersji wskaźniki mają być alokowane w funkcji *main*, każdorazowo dla *MAX_DL_L* znaków
   - po zaalokowaniu, każda linia jest wysyłana do wypełnienia w funkcji *wczytaj_linie* i do drukowania (w celu sprawdzenia poprawności działania) w funkcji *drukuj_linie*
5. Zdefiniowanie (na początku pliku) struktury do przechowywania pojedynczej linii tekstu w postaci:
   typedef struct linia_tekstu {
   char znaki;*
   int dlugosc_linii;
   } linia_tekstu;
6. Modyfikacja programu polegająca na zamianie struktury danych przechowującej tekst na tablicę struktur zdefiniowanego typu:
   struct linia_tekstu tekst_wieloliniowy_tab_str[MAX_N_L];
   - w pierwszej wersji wskaźniki mają być alokowane w funkcji *main*, każdorazowo dla *MAX_DL_L* znaków
   - po zaalokowaniu, każda linia jest wysyłana do wypełnienia w funkcji *wczytaj_linie* i do drukowania (w celu sprawdzenia poprawności działania) w funkcji *drukuj_linie*

7. Skopiowanie zmodyfikowanego pliku *tekst_wieloliniowy.c* do nowego podkatalogu np. *napisy_dyn*
8. Zdefiniowanie w pliku nowej funkcji *wczytaj_linie_dyn*, która
   - otrzymuje jako argument wejściowy wskaźnik do wskaźnika (adres komórki pamięci, która będzie przechowywać adres początku tablicy znaków)
   - wczytuję linię tekstu korzystając z lokalnej zmiennej c*har linia_max[MAX_DL_L+1];*
     - *można także użyć lokalnej zmiennej alokowanej dynamicznie o rozmiarze max_dlugosc_linii+1 (należy pamiętać o zwolnieniu pamięci za pomocą free)*
   - alokuje (za pomocą *malloc*) obszar pamięci dla wczytanej linii
     - w tym momencie alokuje miejsce tylko dla wczytanych znaków!
   - podstawia wskaźnik zwracany przez malloc (czyli adres początku tablicy znaków) w miejsce wskazywane przez argument wejściowy, po czym przepisuje zawartość linii do zaalokowanego obszaru
9. Modyfikacja programu, tak żeby alokacja tablic do przechowywania znaków odbywała się wewnątrz funkcji *wczytaj_linie_dyn*
   - zmiany można zastosować tylko do wersji z p. 4 (tablica wskaźników)i p. 6 (tablica struktur), gdzie operuje się na wskaźnikach do tablic przechowujących znaki
   - drukowanie wprowadzonych linii tekstu odbywa się bez zmian
   - obliczanie rozmiaru struktury danych dotyczy teraz tylko wczytanych znaków - tylko one są przechowywane
   - *(do oceny 4.0 wystarczy zmiana dla p.4 - tablicy wskaźników)*
10. Napisanie kolejnej funkcji *wczytaj_linie_dyn_str*, która przyjmuje jako argument wskaźnik do struktury zawierającej jedną linię (zgodnie z typem zdefiniowanym w p. 5)
    - przykładowy prototyp:
      *int wczytaj_linie_dyn_str(struct linia_tekstu * linia_str_wsk, int max_dlugosc_linii);*
    - działanie jest podobne jak funkcji w p. 8, różni się szczegółami traktowania zwracanych danych
      - wskaźnik zwracany przez *malloc* jest podstawiany do pola struktury (po czym następuje przepisanie znaków)
      - obliczona liczba znaków jest podstawiana do pola struktury
      - pozostawia to dowolność zdefiniowania parametru zwracanego przez funkcję
11. Modyfikacja funkcji *main*, tak żeby korzystając z tablicy struktur do przechowywania tekstu (p. 6) wykorzystać funkcję *wczytaj_linie_dyn_str*
    - w pętli po strukturach (liniach) wywołuje się funkcję *wczytaj_linie_dyn_str* dla każdej z struktur, przesyłając jako argument wskaźnik do struktury
    - drukowanie wprowadzonych linii tekstu odbywa się bez zmian
    - obliczanie rozmiaru struktury danych dotyczy teraz tylko wczytanych znaków - tylko one są przechowywane

---------- 4.0 ----------

Tematy rozszerzające:

1. Skopiowanie zmodyfikowanego pliku *tekst_wieloliniowy.c* do nowego podkatalogu np. *napisy_dyn_dyn*
2. Rozważenie struktur danych dopasowanych do tekstów o zmiennej liczbie linii, zadawanej np. jako dana wejściowa *liczba_linii*
   - wszystkie stosowane dotychczas tablice będą miały zmienny rozmiar, zależny od danej wejściowej *liczba_linii* i będą alokowane w obszarze dynamicznym (na stercie)
   - w definicjach podstawowych struktur danych zamiast tablic pojawią się wskaźniki:
     *char (*tekst_wieloliniowy_tab_2D)[MAX_DL_L] = NULL;*
     *char** tekst_wieloliniowy_tab_wsk = NULL;*
     linia_tekstu tekst_wieloliniowy_tab_str=NULL;*
   - w związku z tym każda z tablic będzie musiała być odpowiednio zaalokowana:
     tekst_wieloliniowy_tab_2D jako tablica tablic znaków o rozmiarze *MAX_DL_L*
     tekst_wieloliniowy_tab_wsk jako tablica wskaźników do (tablic) znaków
     tekst_wieloliniowy_tab_str jako tablica struktur
   - poza alokacją tablic i sposobem obliczania rozmiaru struktury danych program nie wymaga innych

zmian w stosunku do wersji z katalogu *napisy_dyn*, zakładając że:
- alokacji i wczytywania znaków dla *tekst_wieloliniowy_tab_2D* dokonuje funkcja *wczytaj_linie*
- alokacji i wczytywania znaków dla *tekst_wieloliniowy_tab_wsk* dokonuje funkcja *wczytaj_linie_dyn*
- alokacji i wczytywania znaków dla *tekst_wieloliniowy_tab_wsk_str* dokonuje funkcja *wczytaj_linie_dyn_str*

3. W wersji w dostarczonym pliku *tekst_wieloliniowy.c* funkcja *wczytaj_linie* działa w specyficzny sposób w przypadku, gdy ze standardowego wejścia wczytany jest ciąg znaków o długości większej niż *max_dlugosc_linii*
   - przeanalizowanie (na przykładach) działania funkcji w takich przypadkach
   - uzupełnienie kontraktu funkcji o opis działania,
     - np. jawne wypisanie jak liczone są znaki i jak umieszczane w tablicy (m.in. *\n* i *\0*)
     - ustalenie przypadków szczególnych – np. co w przypadku kiedy dane z *stdin* do EOF nie mieszczą się w przekazanej tablicy?
   - ewentualna modyfikacja działania funkcji i kontraktu
     - dodanie możliwych zwracanych wartości będących kodami błędu
     - ustalenie działania (np. w kolejnych wywołaniach funkcji), kiedy pojawią się przypadki szczególne
       - co dzieje się ze znakami pozostającymi w buforze odczytu?
       - co w przypadku kiedy w tablicy nie ma znaku końca linii?
4. Rozważenie przypadku kiedy dopuszczamy w tekście polskie litery
5. Dla zainteresowanych: wykorzystanie tablicy struktur nie kończy możliwych wariantów zarządzania pamięcią do przechowywania tekstu
   - kolejną wersją może być użycie tablicy wskaźników do struktur
     - w stosunku do tablicy struktur zaletą może być mniejszy rozmiar samej tablicy, co może się przydać np. w momencie gdyby wystąpiła potrzeba dynamicznej zmiany rozmiaru tablicy w trakcie działania programu (dodawania i usuwania linii w tekście)
     - definicja struktury danych miałaby postać:
       *linia_tekstu** tekst_wieloliniowy_tab_wsk_str=NULL;*
     - pierwotna alokacja mogłaby przydzielić pewną liczbę linii:
       *tekst_wieloliniowy_tab_wsk_str = malloc(liczba_linii * sizeof(linia_tekstu*));*
     - każda z linii mogłaby zostać odpowiednio zaalokowana stosownie do potrzeb:
       tekst_wieloliniowy_tab_wsk_str [nr_linii] = malloc(sizeof(linia_tekstu));
     - niezaalokowane linie nie zajmują pamięci, ale są gotowe do użycia (alokacji, wypełnienia, itp.)
   - zastosowanie struktur umożliwia także rozbudowywanie pojedynczej struktury o kolejne pola
     - jednym z wariantów mogłoby być wprowadzenie pól będących wskaźnikami do innych struktur i budowanie w takim przypadku dynamicznych struktur danych takich jak lista lub drzewo struktur

Warunki zaliczenia:
1. Obecność na zajęciach i wykonanie co najmniej kroków 1-7
2. Oddanie sprawozdania o treści i formie zgodnej z regulaminem ćwiczeń laboratoryjnych, zawierającego m.in.:
   1. opis wykonanych zadań
   2. kod źródłowy podstawowych funkcji i konstrukcji sterujących
   3. wnioski