Podstawy programowania

Laboratorium 10.

Cel:

- Opanowanie podstaw tworzenia i wykorzystania struktur w C

Zajęcia:

1. Utworzenie katalogu roboczego *lab_10*
2. Skopiowanie pliku *struktury_szablon.c*, jako pomocniczego przy realizacji ćwiczenia
   - całość zadań można wykonywać pracując na pliku *struktury_szablon.c* lub na pliku stworzonym przez siebie
3. Na podstawie wykładu zaprojektowanie własnej struktury, zawierającej kilka parametrów opisujących obiekt z wybranej dziedziny (np. własnej dziedziny zainteresowań)
   - parametry mają być zmiennymi różnych typów, w tym *char[], int* i *double*
   - zmienne zaprojektowanego typu będą przekazywane pomiędzy funkcjami zdefiniowanymi w pliku, dlatego definicja typu powinna być umieszczona poza definicjami funkcji, najlepiej na początku pliku
     - można wykorzystać funkcjonalność *typedef* lub nie, należy później konsekwentnie stosować nazwy typu w programie - ze słowem *struct* lub bez
4. Napisanie programu z funkcją *main*, która kolejno:
   1. definiuje *obiekt_1* będący strukturą zaprojektowanego typu
   2. nadaje wartości parametrom *obiektu_1* posługując się operatorem składowych **.**
   3. wypisuje na ekranie wartości parametrów posługując się operatorem składowych **.**
   4. definiuje *obiekt_2* będący strukturą, inicjując jednocześnie wartości składowych
   5. definiuje wskaźnik do struktury inicjując adresem *obiektu_2*
   6. wypisuje na ekranie zawartość struktury posługując się wskaźnikiem i operatorem –>
   7. definiuje *obiekt_3* będący strukturą, inicjując jednocześnie wartości składowych poprzez przypisanie (skopiowanie) wartości pól z *obiektu_2*
   8. wypisuje na ekranie wartości pól *obiektu_3* posługując się operatorem składowych **.**
5. Modyfikacje programu polegające na napisaniu kolejnych funkcji, wywoływanych następnie przez funkcje *main* (prototypy funkcji, stanowiące ich deklaracje, powinny być wszystkie zgrupowane na początku pliku źródłowego, zaraz po dyrektywach *#include* i definicjach typów):
   1. napisanie funkcji (np. *fun_strukt*), która przyjmuje jako argument wejściowy obiekt będący strukturą
      - funkcja dokonuje modyfikacji składowych struktury i wypisuje na ekranie nowe wartości
   2. uzupełnienie funkcji *main* o:
      1. wywołanie funkcji *fun_strukt* z *obiektem_1* jako argumentem, zaraz po wypisaniu wartości składowych *obiektu_1*
      2. ponowne wypisanie wartości składowych *obiektu_1* po powrocie z funkcji *fun_strukt*
   3. napisanie funkcji (np. *fun_strukt_out*), która przyjmuje jako argument wejściowy obiekt będący strukturą i zwraca obiekt będący strukturą tego samego typu
      - funkcja dokonuje modyfikacji składowych struktury i wypisuje na ekranie nowe wartości

- funkcja zwraca obiekt ze zmodyfikowanymi wartościami

4. uzupełnienie funkcji *main* o:
    1. wywołanie funkcji *fun_strukt_out* z *obiektem_1* jako argumentem, zaraz po wypisaniu wartości składowych *obiektu_1*
    2. przypisaniu (skopiowaniu) zwracanej przez *fun_strukt_out* struktury (z całą zawartością) z powrotem do zmiennej *obiekt_1* i wypisanie wartości pól *obiekt_1* po skopiowaniu
5. napisanie funkcji (np. *fun_strukt_wsk*), która przyjmuje jako argument wejściowy wskaźnik do obiektu będącego strukturą
    1. funkcja dokonuje modyfikacji składowych struktury (uzyskując dostęp poprzez wskaźnik i operator ->) i wypisuje na ekranie nowe wartości
6. uzupełnienie funkcji *main* o:
    1. wywołanie funkcji *fun_strukt_wsk* z *obiektem_2* jako argumentem, zaraz po wypisaniu wartości składowych *obiektu_2*
    2. ponowne wypisanie wartości składowych *obiektu_2* po powrocie z funkcji *fun_strukt_wsk*

-> fun_strukt_out *i* fun_strukt_wsk *to przykłady możliwości modyfikacji zawartości struktury z wykorzystaniem funkcji: w pierwszym przypadku można zmodyfikować pola struktury poprzez przesłanie jej jako argumentu do funkcji i przepisanie zawartości zwracanej struktury do tego samego obiektu (którego zawartość przed chwilą była przesłana - czyli skopiowana na stos - za pomocą argumentu wywołania); w drugim przypadku przekazujemy funkcji wskaźnik do struktury i pozwalamy jej bezpośrednio operować na polach struktury. Opisz w sprawozdaniu działanie zaprojektowanych funkcji dla przykładowych argumentów i struktur z funkcji main*

---------- 3.0 ----------

1. Napisanie funkcji (np. *fun_strukt_wsk_kopia*), która przyjmuje jako argument wejściowy wskaźnik do obiektu będącego strukturą i zwraca strukturę tego samego typu
    - funkcja na samym początku dokonuje przepisania zawartości struktury do zmiennej lokalnej (w operacji inicjowania, posługując się operatorami przypisania i wyłuskania)
    - funkcja dokonuje modyfikacji składowych struktury będącej zmienną lokalną i wypisuje na ekranie nowe wartości
    - funkcja zwraca strukturę będącą zmienną lokalną
1. uzupełnienie funkcji *main* o:
    - wywołanie funkcji *fun_strukt_wsk_kopia* z adresem *obiektu_3* jako argumentem, zaraz po wypisaniu wartości składowych *obiektu_3*
    - przypisaniu (skopiowaniu) zwracanej przez *fun_strukt_wsk_kopia* struktury (z całą zawartością) do nowej zmiennej *obiekt_4* zaprojektowanego typu i wypisanie wartości pól zwróconej struktury w funkcji *main*
2. Napisanie funkcji (np. *fun_strukt_wsk_inout*), która przyjmuje jako argument wejściowy wskaźnik do obiektu będącego strukturą
    - funkcja na samym początku dokonuje przepisania zawartości struktury do zmiennej lokalnej posługując się operatorami przypisania i wyłuskania
    - funkcja dokonuje modyfikacji składowych struktury będącej zmienną lokalną i wypisuje na ekranie nowe wartości
    - funkcja przed zakończeniem przepisuje zawartość struktury ze zmiennej lokalnej do struktury w funkcji wywołującej (*main*)
        - można przetestować rozmaite metody z operatorami *, . i ->

    (funkcja różni się od fun_strukt_wsk tylko tym, że wewnątrz pracuje na lokalnej kopii struktury i modyfikuje wartości z pomocą wskaźnika tylko raz, bezpośrednio przed końcem - takie działanie jest bezpieczniejsze niż wielokrotne operowanie z pomocą wskaźnika)

1. uzupełnienie funkcji *main* o:
   - wywołanie funkcji *fun_strukt_wsk_inout* z adresem *obiektu_4* jako argumentem, zaraz po wypisaniu wartości składowych *obiektu_4*
   - ponowne wypisanie wartości składowych *obiektu_4* po powrocie z funkcji *fun_strukt_wsk_inout*

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

Tematy rozszerzające:

1. Napisanie funkcji, np. *fun_strukt_wsk_out,* która przyjmuje jako argument strukturę zaprojektowanego typu i zwraca wskaźnik do struktury tego typu
   - funkcja ma dokonać alokacji pamięci dla nowej struktury zaprojektowanego typu posługując się funkcją *malloc* i operatorem *sizeof,* następnie przepisać zawartość ze struktury przesłanej jako argument funkcji *fun_strukt_wsk_out* do nowej struktury w obszarze dynamicznym (na stercie) i zwrócić wskaźnik do zaalokowanej struktury (**uwaga: funkcja nie może tylko utworzyć zmiennej lokalnej i zwrócić wskaźnik do zmiennej – taki wskaźnik pokazuje miejsce na stosie, które po zakończeniu działania funkcji jest odzyskiwane do ponownego wykorzystania i po pewnym czasie może zawierać śmieci)**
     - po wywołaniu funkcji *malloc* należy sprawdzić czy nie została zwrócona wartość *NULL* i zaprojektować odpowiednią obsługę błędu (z wykorzystaniem *return* lub *exit* i odpowiednimi wydrukami lub zwracaniem wartości kodów błędu)
     - przed zakończeniem programu pamięć przydzielona przez *malloc* powinna zostać zwolniona za pomocą wywołania funkcji *free* w funkcji *main*
     - wartość zwracaną przez operator *sizeof* można wydrukować
2. Zaprojektowanie nowego typu strukturalnego zawierającej pola, pośród których znajduje się tablica znaków, np. : *{double d, char tab_c[N], int m}*, gdzie N jest np. stałą definiowaną przez #define
   - zbadanie za pomocą operatora *sizeof* rozmiaru pojedynczej zmiennej zaprojektowanego typu dla N równych 1,2,3,4,5 itd.
   - uzupełnienie, na podstawie materiałów internetowych, wiadomości o wyrównaniu adresów (*address alignment*) początków zmiennych do wartości podzielnych przez konkretną liczbę i wytłumaczenie na tej podstawie otrzymanych wartości
3. Zaprojektowanie nowego typu strukturalnego, którego jednym z pól jest wskaźnik do zmiennych tego właśnie typu
   - problem polega na tym, że definiujemy wskaźnik do zmiennej typu, który jeszcze nie został zdefiniowany (bo właśnie go definiujemy)
   - wytłumaczenie dlaczego w zmiennej określonego typu strukturalnego polem nie może być zmienna tego samego typu, a może być wskaźnik do niej

---------- 5.0 ----------

Warunki zaliczenia:

1. Obecność na zajęciach i wykonanie co najmniej kroków 1-5
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