Podstawy programowania

Laboratorium 9.

Cel:

- Opanowanie podstaw wykorzystania wskaźników w C

Zajęcia:

1. Utworzenie katalogu roboczego *lab_9*
2. Skopiowanie za strony przedmiotu pliku *wskazniki_simple.c* do nowego podkatalogu np. *simple*
3. Analiza kodu, uruchomienie
   - szczególna uwaga na arytmetykę wskaźników – dlaczego staramy się jej unikać?
4. Modyfikacje (testowane kolejno po wykonaniu każdej z nich; dodatkowo po każdorazowej modyfikacji należy sprawdzać jaki daje ona efekt w funkcji wywołującej *main):*
   - przesłanie do funkcji *prosta_funkcja* dwóch wskaźników: *wskaznik_do_int* i *double_p*
     - nadanie nazwy argumentom formalnym w funkcji *prosta_funkcja*: *int_ptr* i *doublePtr*
     - wydrukowanie wartości argumentów i zmiennych, na które wskazują
   - modyfikacja wartości:
     - zmiennej, na którą wskazuje *int_ptr*
     - zmiennej, na którą wskazuje *doublePtr*
     - wydrukowanie wartości argumentów i zmiennych, na które wskazują wewnątrz funkcji *prosta_funkcja*
   - modyfikacja wartości wskaźników (*int_ptr++, doublePtr++)*
     - wydrukowanie wartości argumentów i zmiennych, na które wskazują wewnątrz funkcji *prosta_funkcja*
5. Skopiowanie za strony przedmiotu (lub z odpowiedniego katalogu z laboratorium 5) pliku *rownanie_kwadratowe.c* do nowego podkatalogu np. *rownanie_kwadratowe*
6. Analiza programu rozwiązywania równania kwadratowego, uruchomienie
7. Modyfikacja programu polegająca na przeniesieniu kodu rozwiązania równania do osobnej funkcji, np *rownanie_kwadratowe*
   - stworzenie prototypu (nagłówka):
     - zwracana wartość (np. kod sukcesu lub błędu-niepowodzenia) – krótki opis w komentarzu,
     - argumenty: wejściowe i wyjściowe – krótki opis w komentarzu
   - umieszczenie prototypu na początku pliku jako deklaracji
   - skopiowanie prototypu na koniec pliku i dalsze modyfikacje w celu uzyskania poprawnej definicji funkcji
     - napisanie prostej, krótkiej specyfikacji (co robi funkcja)
     - przepisanie (przeklejenie) treści funkcji - kodu obliczania pierwiastków - z funkcji *main*
       - należy wprowadzić odpowiednie zmienne lokalne i właściwie używać przekazanych argumentów
   - w pierwszej wersji nie należy zmieniać istniejącej funkcjonalności, funkcja ma obliczać dwa pierwiastki zakładając poprawne dane wejściowe
     - wszelkie przypadki nie pasujących danych (np. delta<0) mogą się kończyć przerwaniem działania (exit(0))
   - istotą zadania jest poprawne przekazanie wyniku do funkcji *main*
     - w funkcji *main* wywołanie funkcji *rownanie_kwadratowe* ma nastąpić bezpośrednio po wczytaniu współczynników równania
     - wydruk obliczonych wartości ma nastąpić także w funkcji main
       - w funkcji r*ownanie_kwadratowe* można zostawić pomocnicze wydruki kontrolne i komunikaty
8. Uruchomienie programu z funkcją r*ownanie_kwadratowe* i przetestowanie poprawności działania

9. Na podstawie slajdów z wykładu napisanie programu zawierającego funkcję sortowania bąbelkowego o prototypie:
    1. */*----------*
    2. *sortowanie_babelkowe - funkcja sortuje tablicę A o rozmiarze n*
    3. *----------*/*
    4. *void sortowanie_babelkowe(*
    5. *int* A, /* tablica do posortowania */*
    6. *int n /* rozmiar tablicy, A[0], ..., A[N-1] */*
    7. *);*
10. Program powinien zaczynać się w funkcji *main* od stworzenia tablicy o zadanym rozmiarze i wypełnieniu jej wartościami losowymi (podobnie jak w przypadku programów dokonujących operacji na tablicach w ramach laboratorium 6)
11. Wewnątrz funkcji sortowania zastosowanie funkcji *zamien_wyrazy* o prototypie:
    - *void zamien_wyrazy (int *int_1_p, int *int_2_p);*
12. Przetestowanie działania dla tablic o różnych rozmiarach

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

13. Dalsze modyfikacje funkcji *rownanie_kwadratowe*
    - napisanie prostej, krótkiej specyfikacji (co robi funkcja)
    - rozbudowanie treści funkcji o szereg komentarzy wyjaśniających kontrakt:
        - zwracana wartość - kiedy jakie wartości są zwracane (jeden czy wiele kodów błędu-niepowodzenia)
        - argumenty: wejściowe i wyjściowe
            - warunki początkowe (czy są stawiane warunki parametrom wejściowym?)
            - warunki końcowe (np. czy funkcja zawsze zwraca dwa argumenty?, jeśli tak, to jakie?)
        - uzupełnienie opisu funkcji i warunków wejściowych i wyjściowych o jak największą liczbę przypadków, kiedy wykonanie funkcji kończy się niepowodzeniem
            - wersja początkowa: wszystkie takie przypadki umieszczone są w warunkach początkowych – użytkownik nie może wprowadzać danych prowadzących do niepowodzenia (odpowiedzialność po stronie użytkownika, program może wyłącznie wypisać komunikat i zakończyć działanie)
            - testowanie działania funkcji – szczególnie wszelkich sytuacji nietypowych i prowadzących do trudności i niepowodzeń
        - modyfikacja kodu – zamiana kolejnych warunków początkowych na warunki końcowe, czyli zwracane kody błędu
            - funkcja musi sama wykrywać sytuacje prowadzące do niepowodzenia i odpowiednio reagować
            - testowanie działania funkcji – szczególnie wszelkich sytuacji nietypowych i prowadzących do trudności i niepowodzeń
14. Dalsze modyfikacje funkcji *sortowanie_babelkowe*
    - rozważenie wariantu, w którym funkcja *zamien_wyrazy* otrzymuje jako argumenty nie wskaźniki, ale indeksy wyrazów
15. Rozważenie innych algorytmów sortowania (np. przez wstawianie)

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

Warunki zaliczenia:

1. Obecność na zajęciach i wykonanie co najmniej kroków 1-8
2. Oddanie sprawozdania o treści i formie zgodnej z regulaminem ćwiczeń laboratoryjnych zawierającego m.in.:
    - opis wykonanych zadań
    - kod źródłowy podstawowych funkcji i konstrukcji sterujących
    - wnioski