Podstawy programowania

Laboratorium 6.

Cel:

- Opanowanie podstaw wykorzystania tablic i pętli w C.

Zajęcia:

1. Utworzenie katalogu roboczego *lab_6*.
2. Utworzenie podkatalogu *loop*
3. Skopiowanie ze strony przedmiotu do podkatalogu programu *simple_loop.c*
4. Kompilacja w terminalu i sprawdzenie działania
5. Modyfikacje programu *simple_loop.c* (modyfikacji należy dokonywać w dostarczonym pliku, zgodnie z punktami zapisanymi w komentarzach w pliku, najlepiej dla każdej kolejnej modyfikacji wykomentowywać kod dotychczasowy (w odpowiednim punkcie np. 1a) i wpisywać nowy kod w odpowiednim punkcie, np. 1b – dzięki temu cała praca podczas laboratorium będzie znajdowała się w jednym pliku, co ułatwi pisanie sprawozdania i ocenę)*:*
   - inicjowanie tablicy w momencie definiowania:
     - zestaw wartości, pełny i częściowy (uzupełnianie zerami)
   - nadawanie w nowej pętli (poprzedzającej pętlę wypisującą w terminalu) wartości tablicy, jako funkcji indeksu pętli (wartości zmiennej sterującej), np. *2*i*
   - wczytywanie wartości w pętli funkcją *scanf*
     - uzupełnienie o prośbę o podanie każdego kolejnego wyrazu
       - *printf("... tab[%d]....", ..., indeks, ...);*
6. Zamiana jednej z pętli *for* na:
   - pętle *while(...){ ... }* - (np. pierwsza pętla) - testowanie działania
   - pętlę *do{ ... }while(...)* - (np. druga pętla) - testowanie działania
7. Uzupełnienie programu o dodatkowe pętle wykonujące podstawowe operacje na tablicach:
   - znajdowanie elementu maksymalnego
   - obliczanie sumy elementów

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

Tematy rozszerzające:

1. Nadawanie elementom tablicy wartości losowych i dalsze operacje na tablicach:
   - inicjowanie generatora liczb losowych
     - np. *srand(time(NULL)); // info: man srand, man 3 time*
   - generowanie wartości i rzutowanie ich do zadanego przedziału [a, b]
     - np. *int liczba_losowa = a + rand()%(b-a+1); // pseudo-losowość*
     - *man 3 rand podaje inne sposoby rzutowania, bardziej losowe:*
       - *a+(int) ((double)(b-a+1.0)*rand()/(RAND_MAX+1.0));*
   - sprawdzenie poprawności działania – obliczenie wartości średniej elementów tablicy
     - do jakiej wartości powinna zmierzać średnia dla rosnącego w nieskończoność rozmiaru tablicy (przetestowanie rozmiarów tablicy do co najmniej 1000000)?
   - wyszukiwanie elementu o zadanej wartości – należy zwrócić uwagę na dwa różne możliwe warianty (w ramach laboratorium wystarczy uwzględnić jeden z przypadków):
     - tablica z wartościami całkowitymi – porównywanie wartości
     - tablica z wartościami zmiennoprzecinkowymi – zadanie granicznej wartości różnicy między liczbami, poniżej której dwie liczby są traktowane jako identyczne
   - rozważenie wyszukiwania:
     - pierwszej napotkanej wartości (zastosowanie pętli *while*, jako alternatywa pętla *for* z instrukcją *break*)
     - wszystkich wartości zawartych w tablicy

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

(zadanie na ocenę 5.0 zakłada samodzielne badanie funkcjonowania kodu – punkty poniższe są sugestiami działań, sprawozdanie powinno zawierać opis wykonanych kroków, fragmenty kodu, ewentualnie zrzuty ekranu ilustrujące sposób funkcjonowania i wyniki uzyskiwane przez program)

1. Skopiowanie za strony przedmiotu pliku *oblicz_PI.c* do nowego podkatalogu np. *szeregi*
2. Napisanie własnego miniskryptu kompilującego (i ewentualnie uruchamiającego) program *oblicz_PI.c*
3. Analiza programu, uruchomienie, zadanie odpowiednich parametrów sterujących wykonaniem pętli
   - wykorzystanie alternatywne zmiennych typu *double i float* ("programowanie generyczne")
4. Rozważenie:
   - różnych typów pętli (*for, while, do*) – zalety i wady
   - różnych wariantów zakończenia pętli (np. w połączeniu ze zmianami rodzaju pętli):
     1. wykorzystanie skończonej liczby wyrazów (np. pętla *for*)
     2. przerwanie obliczeń gdy kolejne wyrazy dodawane w pojedynczej iteracji osiągną wartość mniejszą od zadanej granicy bliskiej 0 – (np. pętla *while*)
     3. przerwanie obliczeń kiedy przybliżane wartości PI w kolejnych iteracjach przestaną się zmieniać (ich różnica zmaleje poniżej zadanej granicy bliskiej 0) – (np. pętla *do*)
   - pokazanie, że przypadek 3 jest różny od przypadku 2 (w tym celu lepiej wykorzystać zmienne typu *float*)

   (*rozmaite elementy, które można wykorzystać w celu realizacji powyższych zadań znajdują się już w kodzie programu* oblicz_PI.c *odpowiednio wykomentowane; tworząc własne warianty sterowania przebiegiem pętli, można rozważyć zasadę: warunek zadany przez użytkownika umieszczany jest w warunku zakończenia pętli – jako domyślny, nie wymagający komentarza, warunki dodatkowe, np. ograniczenie całkowitej liczby wyrazów, żeby obliczenia zawsze były skończone i nie trwały zbyt długo, ewentualnie ograniczenie wynikające ze zbyt małej precyzji liczb* float*, dla których relatywnie szybko poprawki przestają zmieniać obliczaną wartość PI (poprawki same nie są zbyt małe dla typu* float*, ale relatywnie małe w stosunku do liczonej wartości) – jako warunki dodatkowe wewnątrz pętli, których spełnienie powoduje oprócz przerwania wykonania także dodatkowy komentarz lub inna reakcję (można rozważyć zwracanie kodu błędu przez program))*
5. Szczegółowa analiza realizowanego kontraktu – jak sprawdzane są zadane parametry obliczeń, jak przebiegają obliczenia i jaki jest wynik dla rozmaitych wartości parametrów, w tym wartości powodujących odejście od standardowego kontraktu (np. zbyt duża wymagana dokładność obliczeń)
6. Napisanie własnego programu obliczania sumy częściowej dla szeregu liczbowego (np. obliczanie przybliżenia wartości ln 2, *e*, liczby PI z wzoru Eulera lub wartości wyrażenia 1/(1-x) dla 0<x<1)
   - rozważenie alternatyw podobnych jak w poprzednim programie: typy zmiennych, różne warunki zakończenia pętli, różne rodzaje pętli
   - wprowadzenie do pisanego programu szeregu wydruków kontrolnych pozwalających śledzić na bieżąco:
     - poprawność wprowadzanych danych
     - poprawność przeprowadzanych obliczeń
   - poprawność końcowego wyniku (spełnienie warunku końcowego)

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

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