Podstawy programowania

Laboratorium 13.

Cel:

- Opanowanie programowania z użyciem funkcji rekurencyjnych, wskaźników do funkcji oraz plików

Zajęcia:

1. Utworzenie katalogu roboczego, np. *lab_13*
2. Skopiowanie pliku *potega.c* do podkatalogu np. *rekurencje*
3. Kompilacja i uruchomienie programu (np. tworząc prosty plik *Makefile*, jako modyfikację pliku *Makefile* z poprzedniego laboratorium)
4. Modyfikacja programu polegająca na:
   - napisaniu (np. na podstawie wykładu) funkcji *power_int_rekur* obliczającej całkowitą potęgę liczby całkowitej w sposób rekurencyjny
   - uzupełnienie kodu o makro *assert* sprawdzające w funkcji *power_int_rekur* czy wykładnik jest liczbą dodatnią
     - uruchomienie programu z błędnymi danymi – obserwacja zwracanego komunikatu
     - kompilacja z opcją *-DNDEBUG* , uruchomienie z błędnymi danymi – obserwacja działania
5. Modyfikacja programu polegająca na przekazywaniu parametrów z linii wywołania programu, np. przekazanie podstawy i wykładnika – program drukuje *podstawa^wykładnik*
   - do przekształcenia w liczby całkowite napisów przekazywanych przez środowisko wywołania i wskazywanych przez elementy tablicy *argv[]* można użyć dostarczonej funkcji *my_atoi*
   - wywołanie *my_atoi* można następnie zamienić na wywołanie funkcji bibliotecznej *atoi* (ewentualnie *strtol*) z biblioteki standardowej
6. Modyfikacja kodu poprzez wprowadzenie **zmiennej statycznej** *liczba_wywolan* przechowującej liczbę wywołań funkcji *power_int_rekur*
   - *liczba_wywolan* powinna być inicjowana wartością 0 (takie inicjowanie zmiennej statycznej dokonywane jest raz na początku wykonywania programu)
   - *liczba_wywolan* ma być modyfikowana przy każdorazowym wywołaniu (i ewentualnie drukowana)
   - *power_int_rekur* może zwracać komunikat o błędzie w przypadku przekroczenia zadanej (np. przez *#define* lub s*tatic const int*) wartości granicznej
7. Modyfikacja kodu poprzez wczytywanie argumentów funkcji potęgowania z pliku tekstowego
   - definicja wskaźnika do pliku
   - otwarcie pliku w trybie odczytu
   - wczytanie argumentów – użycie *fscanf*
   - zamknięcie pliku
   - wykonanie operacji i wydruk sprawdzający
   - ewentualnie: otwarcie pliku (w trybie dopisywania), dopisanie wyniku do pliku, zamknięcie pliku

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

8. Skopiowanie pliku *loop_table.c* do podkatalogu np. *tablice* (zamiast pliku *loop_table.c* można skopiować plik z kodem zawierającym tworzenie tablic o losowych wartościach napisanym w ramach poprzednich laboratoriów)
   - uruchomienie, sprawdzenie poprawności działania (wydruki)
9. Rozszerzenie programu z p.8 o wywołanie funkcji bibliotecznej *qsort* sortującej utworzoną tablicę
   - włączenie odpowiednich plików nagłówkowych z prototypami
   - napisanie funkcji porównywania elementów o odpowiednim prototypie zgodnym z prototypem funkcji *qsort*
     - umieszczenie w kodzie deklaracji i definicji funkcji
       - z określeniem *inline* dla zasugerowania optymalizacji wywołania funkcji (jeśli obsługiwane przez kompilator)
   - napisanie wywołania *qsort* w funkcji *main*
   - sprawdzenie działania (kod sprawdzenia wewnątrz dyrektywy kompilacji warunkowej):
     - poprzez wydruki
     - poprzez pętlę porównującą dwa kolejne wyrazy dla całej tablicy
       - do porównania można użyć makra *assert*
     - poprzez porównanie z wynikiem innego algorytmu sortowania, np. napisanym w ramach poprzednich laboratoriów sortowaniem bąbelkowym

10. Dodanie do programu z p.8 i9 zapisu i odczytu tablic liczbowych do i z pliku tekstowego o wybranej nazwie
    - zapis tablicy zaraz po wygenerowaniu – użycie *fprintf*
    - odczyt do innej tablicy – użycie *fscanf* – porównanie wartości w tablicach
        - poprawność zapisu można także sprawdzić poprzez przeglądanie zawartości pliku w dowolnym edytorze tekstu
    - modyfikacja pliku poprzez zmianę wprowadzoną w edytorze tekstu powodującą błąd odczytu
        - wypisanie informacji o błędzie w trakcie odczytu do standardowego strumienia błędu *stderr*
    - wersja początkowa programu może zawierać nazwy plików bezpośrednio wpisane do kodu
    - wersja ostateczna powinna pobierać nazwę pliku jako argument linii wywołania
        - nazwa powinna określać pełną ścieżkę do pliku
        - katalog, w którym mają znajdować się pliki może zostać przekazany jako inny argument linii wywołania i połączony w jedną nazwę za pomocą odpowiedniej funkcji obsługi napisów, np. *sprintf*

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

Tematy rozszerzające:

1. Rozszerzenie programu z p.8,9,10 o wywołanie funkcji wyszukującej wartości w posortowanej tablicy
    1. odkomentowanie fragmentów funkcji *main* zawierających nieskończoną pętlę wyszukiwania wartości w tablicy (pętlę można uzupełnić o warunek zakończenie obliczeń, np. przez podanie -1)
    2. odkomentowanie poprawnego wywołania funkcji *binsearch_iter* wyszukującej wartość w tablicy
    3. uruchomienie – sprawdzenie działania poprzez porównanie wydruków
    4. dodanie porównania wyniku zwracanego przez funkcję *binsearch_iter* z wynikiem zwracanym przez funkcję biblioteczną *bsearch*
        1. sprawdzenie powinno być wewnątrz obszaru dyrektywy kompilacji warunkowej
        2. sprawdzenie może wykorzystywać makro *assert* lub własną obsługę błędu
2. Zamiana funkcji *binsearch_iter* na funkcję *binsearch_rekur* realizującą algorytm rekurencyjny
    1. wykorzystanie dostarczonego prototypu, umożliwiającego wyszukiwanie we fragmencie tablicy
    2. implementacja algorytmu (np. na podstawie pseudokodu z wykładu)
    3. wpisanie poprawnego wywołania w funkcji *main* zamiast *binsearch_iter*
        1. uwaga na poprawne określenie indeksów tablicy
    4. uruchomienie, sprawdzenie poprawności działania
        1. np. poprzez wielokrotne (w pętli o zadanej liczbie iteracji) automatyczne generowanie wyszukiwanej liczby losowej i sprawdzanie czy rzeczywiście znajduje się na znalezionej pozycji
3. Rozszerzenie programu sortowania tablic o generowanie tablic liczb podwójnej precyzji o losowych wartościach (np. wykorzystując funkcję *drand48* lub prostą konwersję *r=a+rand()*(b-a)/RAND_MAX)*
    1. zapis liczb do plików tekstowych z wykorzystaniem różnych sposobów formatowania
        1. *%f, %e, %g* – różne warianty dodatkowych parametrów
    2. sprawdzenie poprzez przeglądanie plików w edytorach tekstu
4. Zamiana zapisu do pliku i odczytu z pliku w postaci formatowanej w programie sortowania na zapis i odczyt binarny
    - wykorzystanie niskopoziomowych funkcji z interfejsu systemowego *open*, *read* i *write (man 3 open, man 3 read, man 3 write)*

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 utworzonych i zmodyfikowanych plików źródłowych
    3. zrzuty ekranu z wynikami działania programów
    4. wnioski