4.03.2022

- Napisz (albo odszukaj w archiwum zadań domowych z pierwszego roku studiów) program w C deklarujący tablicę `int liczby[50]` i wczytujący do niej z klawiatury kolejne liczby. Wczytywanie należy przerwać gdy użytkownik wpisze zero lub po wczytaniu 50 liczb.

- Dodaj do powyższego programu pętlę for idącą przez wszystkie wczytane liczby i drukującą te z nich, które są większe od 10 i mniejsze od 100.

- Powyższy program prawdopodobnie używał indeksowania tablic. Jeśli tak, to przerób go aby korzystał ze wskaźników i aby nigdzie w kodzie nie występowało `liczby[i]`.

- Przerób program tak, aby kończył działanie również wtedy, kiedy zabraknie na standardowym wejściu liczb do wczytania (wystąpi warunek "end-of-file", `EOF`).

  `EOF` wystąpi kiedy np. przekierujemy na standardowe wejście naszego programu liczby zapisane wcześniej w pliku tekstowym, np.

  ```
  ./moj_program.x < plik_z_liczbami.txt
  ```

  lub kiedy wejście do programu stanowi potok (*pipe*) z innego programu, np.

  ```
  printf "1 2 3 4 5" | ./moj_program.x
  ```

  Jak zasymulować `EOF` wpisując liczby z klawiatury?

  Zwróc uwagę na dokumentację odpowiednich funkcji jak `scanf` czy `fgets`. Co robią w przypadku wystąpienia warunku EOF?

- Opracuj funkcję testującą czy przekazany jej bufor zawiera tylko i wyłącznie drukowalne znaki ASCII, tzn. bajty o wartościach z przedziału domkniętego [32, 126]. Funkcja ma mieć następującą sygnaturę: `bool drukowalne(const void * buf, int len)`.

- Pamiętaj o włączeniu nagłówka `<stdbool.h>`, bez niego kompilator nie rozpozna ani nazwy typu `bool`, ani nazw stałych `true` i `false`.

- Zaimplementuj dwa warianty tej funkcji, w pierwszym wariancie funkcja ma pobierać kolejne bajty z bufora przy pomocy operacji indeksowania tablic, w drugim ma używać wskaźnika przesuwającego się z bajtu na bajt.

- A teraz opracuj wersję, która jako argument dostaje łańcuch w sensie języka C, czyli ciąg niezerowych bajtów zakończony bajtem równym zero (ten końcowy bajt nie jest uznawany za należący do łańcucha). Ta wersja funkcji powinna mieć taką sygnaturę: `bool drukowalne(const char * buf)`.

- Tu też zaimplementuj dwa warianty: używający indeksowania i używający przesuwającego się wskaźnika.

- W dokumentacji POSIX API znajdź opisy czterech podstawowych funkcji plikowego wejścia-wyjścia, tzn. `open`, `read`, `write` i `close`. Czy zgadzają się one z tym, co pamiętasz z przedmiotu „Systemy operacyjne”? Jakie znaczenie ma wartość 0 zwrócona jako wynik funkcji read?

- Zaimplementuj program kopiujący dane z pliku do pliku przy pomocy powyższych funkcji. Zakładamy, że nazwy plików są podawane przez użytkownika jako argumenty programu (tzn. będą dostępne w tablicy argv). Zwróć szczególną uwagę na obsługę błędów — każde wywołanie funkcji we-wy musi być opatrzone testem sprawdzającym, czy zakończyło się ono sukcesem, czy porażką.

Funkcje POSIX zwracają -1 aby zasygnalizować wystąpienie błędu, i dodatkowo zapisują w globalnej zmiennej `errno` kod wskazujący przyczynę wystąpienia błędu (na dysku nie ma pliku o takiej nazwie, brak wystarczających praw dostępu, itd.). Polecam Państwa uwadze pomocniczą funkcję `perror`, która potrafi przetłumaczyć ten kod na zrozumiały dla człowieka komunikat i wypisać go na ekranie.

- Modyfikacja powyższego zadania. Zakładamy, że kopiowany plik jest plikiem tekstowym. Linie są zakończone bajtami o wartości 10 (znaki LF, w języku C zapisywane jako `'\n'`). Podczas kopiowania należy pomijać parzyste linie (tzn. w pliku wynikowym mają się znaleźć pierwsza, trzecia, piąta linia, a druga, czwarta, szósta nie).

- (nieobowiązkowe) Popraw powyższy program tak, aby i znaki `'\n'`, i dwubajtowe sekwencje złożone ze znaku `'\r'` i następującego po nim znaku `'\n'` były traktowane jako terminatory linii.