C - 7 - Operacje na plikach

7.1 Odczyt z pliku

Ścieżkę do pliku normalnie zapisalibyśmy jako **c:\mojePliki\dane\pliczek.txt**. Ale znak **** ma specjalne znaczenie w języku C, należy więc zapisać: **char *nazwaPliku ="c:\\mojePliki\\dane\\pliczek.txt";** Jeżeli jednak wprowadzasz nazwę pliku z klawiatury, to zawsze podawaj pojedyńcze ukośniki !!! Komputer sam zamieni je na podwójne.

Utwórz na dysku **C** katalog **mojeDane**, w tym katalogu utwórz plik **dane.txt** i zapisz do niego w dwóch linijkach:

Co komu do tego,
skoro i tak mniejsza o to?

Następnie napisz program:

```
#include "stdafx.h"

int main(int argc, char* argv[]){

        char *nazwaPliku = "c:\\mojeDane\\dane.txt";
        char znaczek;
        FILE *plik;

        plik = fopen(nazwaPliku,"r");

        if(plik==NULL){
                printf("Nie znaleziono pliku lub plik jest uszkodzony");
                return 1;
        }

        while( !feof(plik) ){
                znaczek = fgetc(plik);
                printf("%c",znaczek);
        }
        fclose(plik);
        return 0;
}
```

Jeżeli ścieżka do pliku jest wpisana „na sztywno", to zawsze dajemy dwa ukośniki!!! Jeżeli podajemy ścieżkę do pliku z klawiatury, to podajemy jeden ukośnik.
FILE, to typ plikowy. Typ, czyli coś takiego jak int, char, ...

Jeżeli otwarcie pliku nie powiodło się ...

fopen otwiera plik. Funkcja ta pobiera dwa parametry – ścieżkę dostępu oraz specyfikator.

Wychodzimy z funkcji main i zgłaszamy błąd.

eof to skrót od *End of File*.

Elegancja nakazuje zamknąć plik.

fgetc pobiera jeden znaczek, po czym wskaźnik plikowy sam przesuwa się dalej.

Mamy następujące specyfikatory dostępu do pliku:
„**r**" – otwarcie pliku tylko do odczytu, jeżeli plik nie istnieje, to **fopen** zwraca **NULL**.
„**w**" – otwarcie pliku do zapisu. Jeżeli plik nie istnieje, to zostaje stworzony. Jeżeli już istnieje, to zostaje wykasowany, a na jego miejsce tworzony jest nowy pusty plik.
„**a**" – otwarcie pliku do zapisu, nowa treść będzie doklejona na końcu pliku. Jeżeli plik nie istnieje, to zostaje stworzony.
„**r+**" – otwarcie pliku do zapisu i do odczytu. Jeżeli plik nie istnieje, to zostaje stworzony. Jeżeli istnieje, to nowe dane zostaną zapisane na początku pliku, nadpisując stare dane.
„**w+**" - otwarcie pliku do zapisu i do odczytu. Jeżeli plik nie istnieje, to zostaje stworzony. Jeżeli już istnieje, to zostaje wykasowany, a na jego miejsce tworzony jest nowy pusty plik.
„**a+**" – otwarcie pliku do odczytu i zapisu. Nowe dane zostaną dopisane na końcu pliku.

Zadanie 0. Sprawdź, że w pętli **while** wystarczy wpisać **printf("%c",fgetc(plik));**

Zadanie 1. Wypisz tylko co drugą literę z pliku, zawartość pętli **while** może zawierać tylko dwie linijki.

Zadanie 2. Przerób powyższy program – niech użytkownik podaje nazwę pliku (całą ścieżkę) z klawiatury.

Zadanie 3. Przerób powyższy program – niech użytkownik podaje nazwę pliku w linii poleceń (o parametrach z linii poleceń było przy okazji tablic). Jeżeli nie podano parametru z linii poleceń, wypisz komunikat o błędzie.

Teraz w pliku dane.txt wpisz tylko liczby zmiennoprzecinkowe, każdą w osobnej linijce. Np:
2.3
-32.4
123.4
5444.2
123.2
4343.2

Poniższy kod wczytuje do tablicy liczby z pliku, korzystając z osobnej funkcji:

```
#include "stdafx.h"

int wczytajLiczby(char *nazwaPliku,double *tablica,int MAX){

	FILE *plik = fopen(nazwaPliku,"r");
	if(plik==NULL){
		printf("Nie znaleziono pliku lub plik jest uszkodzony");
		return 0;
	}

	int licznik=0;

	while(! feof(plik)){
		fscanf(plik,"%lf",tablica++);
		if(++licznik==MAX) break;
	}

	fclose(plik);
	return licznik;
}


int main(int argc, char* argv[]){

	double tab[20];

	int ileLiczb = wczytajLiczby("c:\\mojeDane\\dane.txt",&tab[0],20);

	if(ileLiczb > 0){
		printf("Podales liczby:\n");
		for(int i=0;i<ileLiczb;i++)	printf("%lf\n",tab[i]);
	}

	return 0;
}
```

Przekazujemy ścieżkę do pliku, wskaźnik do pierwszego elementu tablicy oraz maksymalną liczbę elementów. Zwracamy liczbe wczytanych liczb.

Tu pobieramy ze zmiennej plikowej plik liczbę typu double, czyli %lf i od razu przesuwamy się na kolejne miejsce w tablicy. Możnaby napisać:

```
fscanf(plik,"%lf",tablica);
tablica ++;
```

Ale to zajęłoby aż dwie linijki :)

Zwiększamy licznik liczb o jeden za pomocą ++ i sprawdzamy, czy osiągnęliśmy już maksymalną liczbę liczb. Możnaby napisać:

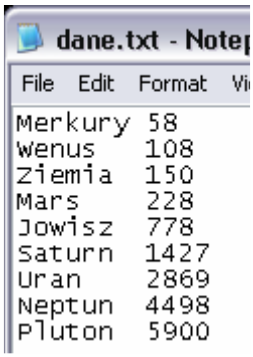
```
licznik++;
if(licznik==MAX) break;
```

Ale to znowu zajęłoby dwie linijki.

Zadanie 4. Napisz program, który pobiera z pliku co najwyżej 20 liczb, a następnie sortuje je malejąco i znajduje średnią i medianę.

Poniższy przykład wczytuje z pliku dwa rodzaje danych oddzielonych spacją – napisy i liczby:

```
#include "stdafx.h"

int wczytajLiczby(char *nazwaPliku,double *odleglosci,char planety[20][20],int MAX){

	FILE *plik = fopen(nazwaPliku,"r");
	if(plik==NULL){
		printf("Nie znaleziono pliku lub plik jest uszkodzony");
		return 0;
	}

	int licznik=0;

	while(! feof(plik)){
		fscanf(plik,"%s %lf",*(planety++),odleglosci++);
		if(++licznik==MAX) break;
	}

	fclose(plik);
	return licznik;
}

int main(int argc, char* argv[]){

	double odl[20];
	char pl[20][20];

	int ileDanych = wczytajLiczby("c:\\mojeDane\\dane.txt",&odl[0],pl,20);

	if(ileDanych > 0){
		printf("Planeta i odleglosc od slonca:\n");
		for(int i=0;i<ileDanych;i++)  printf("%s\t%.2lf\n",*(pl+i),odl[i]);
	}

	return 0;
}
```

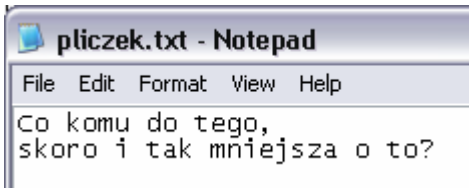


Oddzielamy tabulacją, żeby ładniej wyrównać w pionie.

Zadanie 5. Napisz program, który pobiera z pliku dane w trzech formatach: napis, liczba zmiennoprzecinkowa i liczba całkowita, np: marka samochodu, pojemność silnika i maksymalna prędkość. Dane te powinny być zapisane do trzech tablic. Napisz trzy funkcje, które będą sortować dane – użytkownik powinien wskazać, po czym ma się odbywać to sortowanie. Czwarta funkcja może wyświetlać dane w jakimś eleganckim formacie, np. w tabelce.

Do wczytywania całych linijek z pliku służy funkcja **fgets**. Jeżeli poniższy program nie działa, to znaczy że pewnie **zapomniałeś stworzyć plik z danymi** – jest to najczęstszy błąd.

```
#include "stdafx.h"

int main(int argc, char* argv[]){

	FILE *plik;
	int rozmiar;
	char wyraz[256];

	plik=fopen("c:\\pliczek.txt","r");
	if (plik==NULL) return 1;

	fseek(plik,0,SEEK_END);
	rozmiar = ftell (plik);
	printf("Plik zawiera %d bajtow\n\n",rozmiar);
	rewind(plik);

	int linijki=0;

	while ( fgets(wyraz, 256, plik) != NULL){
		linijki++;
		printf("%s",wyraz);
	}

	printf("\n\nPlik zawiera %d linijek.\n\n", linijki);

	fclose(plik);

	return 0;
}
```



Funkcja fseek przesuwa wskaźnik pliku. Jest wywoływana przed funkcją ftell, żeby ta mogła zwrócić rozmiar pliku. Sprawdź, że po zakomentowaniu linijki z fseek funkcja ftell nie zwróci poprawnie rozmiaru pliku.

Funkcja rewind „przewija" plik do początku. Sprawdź, że po zakomentowaniu tej linijki zawartość pliku nie zostanie wypisana.

Funkcja fgets pobiera:
1) zmienną, do której należy wpisać tekst
2) liczbę bajtów do przeczytania (jeżeli w pliku będzie \n, to czytanie się skończy)
3) wskaźnik do pliku

Dwa ostatnie parametry funkcji **fseek** oznaczają, że należy przesunąć wskaźnik pliku o **zero** bajtów od końca pliku (**SEEK_END**) – oznacza to, że wskaźnik ma się znaleźć na końcu pliku. Oprócz **SEEK_END** mamy też **SEEK_SET** (początek pliku) oraz **SEEK_CUR** (bieżąca pozycja).

Zadanie 6. Utwórz plik z dużą (więcej niż 2) liczbą linijek. Wczytaj wszystkie linijki - te nieparzyste poprzedź tabulacją podczas wyświetlania na ekranie.

7.2 Zapis do pliku

Przykład zapisu do pliku – efektów nie będzie widać na ekranie, lecz w pliku. Po zakończeniu działania programu sprawdź, czy na dysku C utworzono plik i zapisano do niego dane.

```
#include "stdafx.h"

int main(int argc, char* argv[]){

	int tab[20];
	tab[0] = 1;
	tab[1] = 1;
	for(int i=2;i<20;i++) tab[i] = tab[i-1]+tab[i-2];

	FILE *plik;
	plik=fopen("c:\\fibonacci.txt","w");
	if (plik==NULL) return 1;

	fprintf(plik,"Pierwsze 20 liczb Fibonacciego:\n");
	for(i=0;i<20;i++)
		fprintf(plik,"%d\n",tab[i]);

	fclose(plik);

	printf("Zapisano dane.\n");

	return 0;
}
```

Funkcja fprintf pobiera:
1) wskaźnik do pliku
2) zawartość identyczną jak w funkcji printf

Jeżeli program zapisuje coś do pliku, to studenci prawie zawsze są przekonani, że program nie działa, ponieważ efektów nie widać na ekranie. Trzeba im ciągle powtarzać, żeby sprawdzili plik :(

Zadanie 7. W programie robiącym cokolwiek ponownie otwórz plik **fibonacci.txt** do zapisu i zapisz w nim coś. Zauważ, że poprzednie dane zostały nadpisane. W jeszcze kolejnym programie otwórz ten sam plik, ale z opcją **a**, sprawdź, że dane zostaną dopisane na końcu pliku.

Zadanie 8. Sprawdź, co się stanie, jeżeli ze ścieżki pliku w funkcji **fopen** wykasujesz fragment **c:**. Wskazówka: plik znajdzie się gdzieś w katalogu z Twoim projektem.

Studenci często podają tylko nazwę pliku (bez ścieżki) i oczekują, że komputer domyśli się, że plik miał być np. na dysku D. Potem są oburzeni, że pliku tam nie ma :P

Zadanie 9. Utwórz plik macierz.txt z macierzą 3x3. Otwórz go do odczytu i zapisu na końcu pliku (jaka będzie opcja otwarcia pliku?). Oblicz wyznacznik macierzy i dopisz go w pliku za macierzą nie kasując żadnych danych.

Zadanie 10. Poczytaj w internecie o funkcjach **fread** i **fwrite**. Napisz program ilustrujący ich działanie.

7.3 Inne operacje na plikach/katalogach

```
#include "stdafx.h"

int main(int argc, char* argv[]){

        if ( remove("c:\\fibonacci.txt") == 0)
                printf("Usuniecie pliku powiodlo sie\n");
        else
                printf("Blad usuniecia pliku\n");

        return 0;
}
```

Do usunięcia pliku służy funkcja **remove** – należy z niej korzystać bardzo ostrożnie.

Zadanie 11. Sprawdź, co się stanie przy próbie usunięcia pliku, który nie istnieje.

```
#include "stdafx.h"

int main(int argc, char* argv[]){

        if ( rename("c:\\fibonacci.txt","c:\\lucas.txt") == 0)
                printf("Przenazwowanie pliku powiodlo sie\n");
        else
                printf("Blad przenazwowania pliku\n");

        return 0;
}
```

Aby zmienić nazwę pliku, skorzystamy z funkcji **rename**. Utwórz plik fibonacci.txt na dysku C i wpisz do niego cokolwiek. Następnie wykonaj przykładowy program.

Zadanie 12. Sprawdź, co się stanie przy próbie przenazwowania pliku, który nie istnieje.

Zadanie 13. Napisz program, w którym użytkownik poda z klawiatury starą i nową nazwę pliku.

Zadanie 14*. Napisz program, w którym użytkownik poda nazwę istniejącego pliku oraz nowego pliku. Program ma przekopiować zawartość ze starego do nowego pliku. Żeby było bardziej elegancko, napisz osobną funkcję, która będzie przyjmować dwa parametry (ścieżki do plików) i kopiować jeden plik do drugiego.

Visual C++ umożliwia też działanie na katalogach:

```
#include "stdafx.h"
#include "direct.h"

int main(int argc, char* argv[]){

        if ( _mkdir("c:\\moj_wlasny_katalog") == 0){
                printf("Utworzylem katalog\n");
                if ( _mkdir("c:\\moj_wlasny_katalog\\moj_podkatalog") == 0)  printf("Utworzylem podkatalog\n");
                else printf("Nie utworzylem podkatalogu - moze on juz istnieje?\n");
        }
        else  printf("Nie utworzylem katalogu - moze on juz istnieje?\n");

        printf("\n\nTeraz jestes na dysku %c \n",_getdrive() + 'A' -1);

        char gdzieJestes[256];
         _getcwd(gdzieJestes, 256);
        printf ("... a dokladniej mowiac, w katalogu:\n %s\n", gdzieJestes);
        printf("\nSprawdze jakie masz dyski (oprocz stacji dyskietek):\n");

        for(int i='C'-'A'+1;i<'Z'-'A'+1;i++){
          if (_chdrive(i) == 0)   printf ("Masz dysk %c\n",_getdrive()+'A'-1);
        }
```

Funkcja _mkdir tworzy katalog. W razie sukcesu zwraca zero. Sprawdź, czy da się od razu utworzyć katalog c:\\moj_wlasny_katalog\\moj_podkatalog zamiast tworzyć go stopniowo.

Funkcja _getdrive zwraca numer dysku, na którym działamy w danej chwili. Dla dysku A będzie to 1, dla B 2, dla C 3, itd ... Trzeba więc przerobić ten numerek na literkę podczas wyświetlania.

Funkcja _getcwd zwraca bieżący katalog. Pobiera zmienną tekstową, do której zapisze ścieżkę katalogu oraz długość tej zmiennej tekstowej.

Funkcja _chdrive zmienia bieżący dysk na inny (pobiera jego numer). W razie powodzenia zwraca zero.

```
        printf("\nTeraz zmienie biezacy katalog i utworze tam plik\n");
        if(_chdir("c:\\moj_wlasny_katalog\\moj_podkatalog")==0){
                FILE *plik = fopen("aqq.txt","w");
                fprintf(plik,"A kuku !!!");
                fclose(plik);
        }

        return 0;
}
```

Funkcja _chdir zmienia bieżący katalog na inny. W razie powodzenia zwraca zero.

Po wykonaniu tego przykładu sprawdź, czy na dysku **c** w katalogu **c:\moj_wlasny_katalog\moj_podkatalog** znajduje się plik **aqq.txt** z odpowiednią treścią.

Zadanie 15. Napisz program, który usunie katalogi stworzone w powyższym programie za pomocą funkcji **_rmdir** – sprawdź, czy da się usunąć katalog i podkatalog od razu, czy też trzeba to robić stopniowo. Korzystaj z funkcji **_rmdir** z ostrożnością – nie usuń np. całej zawartości dysku C.

Zadanie 16. Napisz własną funkcję **czyIstniejeDysk**, która pobiera zmienną typu **char** i zwraca wartość typu **bool**. Funkcja ma orzekać, czy na komputerze istnieje dysk reprezentowany przez podaną literkę. Wewnątrz tej funkcji należy skorzystać z funkcji **_chdrive**.

7.4 Najczęstsze błędy

Podawanie pojedyńczych ukośników w ścieżkach plików podawanych "na sztywno".

Podawanie podwójnych ukośników w ścieżkach plików podawanych z klawiatury.

Najczęstszy błąd: zapominanie o stworzeniu pliku z danymi **oraz** podawanie tylko nazwy pliku, bez pełnej ścieżki.

Brak obsługi błędów plikowych (np. nieistniejący plik).

Niezamykanie plików za pomocą **fclose**.

Oczekiwanie, że zapis do pliku będzie zawsze widoczny na ekranie.

Nieostrożne korzystanie z funkcji usuwających pliki i katalogi.

7.5 Quiz

1. Jakiej opcji otwarcia pliku użyjesz, jeżeli:
   a) chcesz otworzyć plik do odczytu i do zapisu, ale w taki sposób, by nie utracić danych w pliku? _____
   b) chcesz otworzyć plik do odczytu i do zapisu, ale w taki sposób, by wykasować stare dane? _____

2. Wypełnij tabelkę:

funkcja	liczba pobieranych parametrów – co one oznaczają?	zwracana wartość – co oznacza?
fgets		
fprintf		
feof		
ftell		

3. W jakiej bibliotece znajdują się funkcje operujące na katalogach? _ _ _ _ _ _ . h
   Wymień trzy funkcje zaczynające się od znaczka _ i napisz, co one robią:

_____
_____
_____

_____
_____
_____

_____
_____
_____