wcztaj znak i wypisz jego kod dwójkowo

- kolejne cyfry w układzie pozycyjnym to reszta z dzielenia liczby przez podstawę układu
- drobny problem, że w/w kolejność jest od prawej do lewej
- trzeba jakoś odwrócić kolejność wypisywania cyfr

wcztaj znak i wypisz jego kod dwójkowo

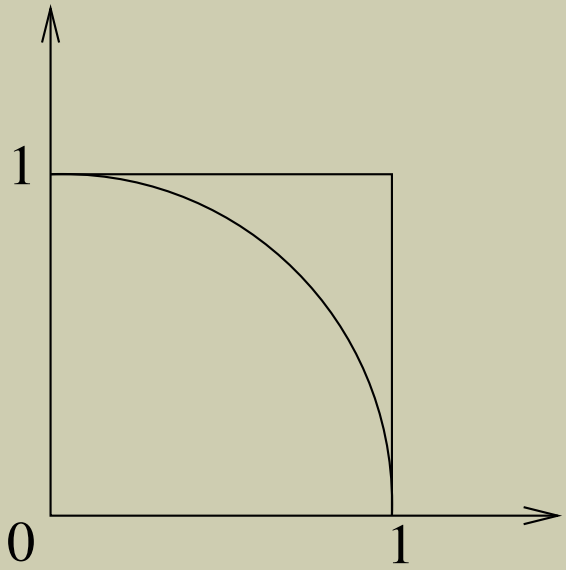
- kolejne cyfry w układzie pozycyjnym to reszta z dzielenia liczby przez podstawę układu
- drobny problem, że w/w kolejność jest od prawej do lewej
- trzeba jakoś odwrócić kolejność wypisywania cyfr
- można zapamiętać kolejne cyfry w tablicy znaków
- wpisywać je od ostatniej pozycji (7) do pierwszej (0)
- znak ma wartośc mieszczącą się w 8 bitach
- jeśli mamy taką kolejność cyfr dwójkowych to wystarczy wypisać gotowy napis

wcztaj znak i wypisz jego kod dwójkowo

- kolejne cyfry w układzie pozycyjnym to reszta z dzielenia liczby przez podstawę układu
- drobny problem, że w/w kolejność jest od prawej do lewej
- trzeba jakoś odwrócić kolejność wypisywania cyfr
- można zapamiętać kolejne cyfry w tablicy znaków
- wpisywać je od ostatniej pozycji (7) do pierwszej (0)
- znak ma wartośc mieszczącą się w 8 bitach
- jeśli mamy taką kolejność cyfr dwójkowych to wystarczy wypisać gotowy napis
- przykładowe rozwiązanie:
  `http://www.oa.uj.edu.pl/~soida/wyklady/C/progs/10to2.c`

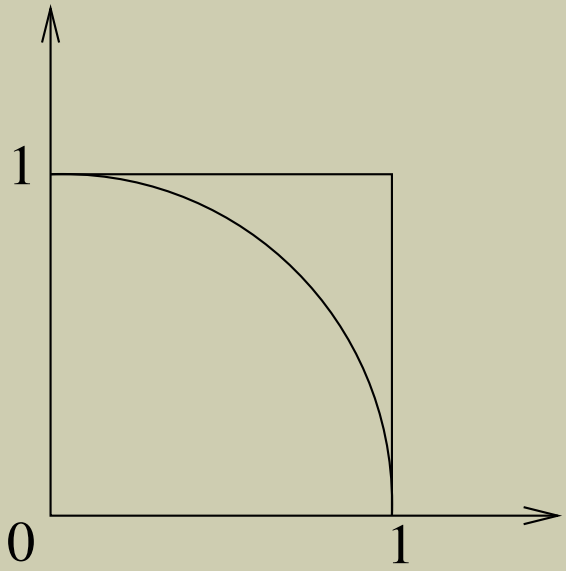
wyznaczenie π za pomocą generatora liczb pseudolosowych

- $\pi/4$ to (m.in.) stosunek pola koła do pola opisanego na nim kwadratu
- generator liczb pseudolosowych może generować przypadkową liczbę w przedziale [0,1]
- wygodniej będzie użyć tylko ćwiartki koła wpisanego w kwadrat (o boku 1)



wyznaczenie π za pomocą generatora liczb pseudolosowych

- $\pi/4$ to (m.in.) stosunek pola koła do pola opisanego na nim kwadratu
- generator liczb pseudolosowych może generować przypadkową liczbę w przedziale [0,1]
- wygodniej będzie użyć tylko ćwiartki koła wpisanego w kwadrat (o boku 1)
- spróbujmy wylosować punkt (parę współrzędnych) – wiele razy
- sprawdźmy, czy ten punkt leży w ćwiartce koła
- ilość punktów „trafionych” do wszystkich da nam stosunek pola powierzchni ćwiartki koła do pola kwadratu



wyznaczenie π za pomocą generatora liczb pseudolosowych

- funkcja generująca liczbę losową to `rand()`
- zwraca liczbę całkowitą w zakresie [0,`RAND_MAX`]
- stała `RAND_MAX` jest zdefiniowana w `stdlib.h`
- jak otrzymać liczbę w zakresie [0,1]?

wyznaczenie π za pomocą generatora liczb pseudolosowych

- funkcja generująca liczbę losową to `rand()`
- zwraca liczbę całkowitą w zakresie [0,`RAND_MAX`]
- stała `RAND_MAX` jest zdefiniowana w `stdlib.h`
- jak otrzymać liczbę w zakresie [0,1]?
- wystarczy podzielić `rand()/RAND_MAX`
- ale `rand()` to `int` i `RAND_MAX` też – dzielenie będzie całkowite (i zawsze zero!)
- `http://www.oa.uj.edu.pl/~soida/wyklady/C/progs/pi.c`

czytanie liczb, wyznaczenie średniej

- czytamy liczby – póki są do przeczytania
- funkcja `scanf()` zwraca ilość poprawnie przeczytanych wartości
- sumujemy wszystkie kolejne czytane liczby
- jednocześnie zliczamy ile przeczytaliśmy liczb
- średnia to iloraz sumy liczb przez ich ilość

wyznaczenie średniej i odchylenia standardowego

- czytamy liczby – póki są do przeczytania
- do odchylenia standardowego potrzebujemy średnią
  $\sigma^2 = \frac{1}{n} \sum_{i=1}^{n} (x_i - \bar{x})^2$
- sumujemy wszystkie kolejne czytane liczby i ich ilość
- liczymy średnią
- . . . i nie mamy już informacji o poszczgólnych liczbach(!)

wyznaczenie średniej i odchylenia standardowego

- czytamy liczby – póki są do przeczytania
- do odchylenia standardowego potrzebujemy średnią
  $\sigma^2 = \frac{1}{n} \sum_{i=1}^{n} (x_i - \bar{x})^2$
- sumujemy wszystkie kolejne czytane liczby i ich ilość
- liczymy średnią
- ... i nie mamy już informacji o poszczgólnych liczbach(!)
- przechować lczby w tablicy $\rightarrow$ jaki rozmiar?
- jeśli czytamy z pliku można go przeczytać raz jeszcze

wyznaczenie średniej i odchylenia standardowego

- czytamy liczby – póki są do przeczytania
- do odchylenia standardowego potrzebujemy średnią
  $\sigma^2 = \frac{1}{n} \sum_{i=1}^{n} (x_i - \bar{x})^2$
- sumujemy wszystkie kolejne czytane liczby i ich ilość
- liczymy średnią
- ... i nie mamy już informacji o poszczgólnych liczbach(!)
- przechować lczby w tablicy $\rightarrow$ jaki rozmiar?
- jeśli czytamy z pliku można go przeczytać raz jeszcze
- można pomyśleć i policzyć ...

wyznaczenie średniej i odchylenia standardowego

$$\sigma^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i^2 - 2\bar{x}x_i + \bar{x}^2) =$$

$$= \frac{1}{n}\left(\sum_{i=1}^{n} x_i^2 - 2\bar{x}\sum_{i=1}^{n} x_i + \sum_{i=1}^{n}\bar{x}^2\right) =$$

$$= \frac{1}{n}\left(\sum_{i=1}^{n} x_i^2 - 2\bar{x}n\bar{x} + n\bar{x}^2\right) = \frac{1}{n}\sum_{i=1}^{n} x_i^2 - \bar{x}^2$$

$$\sigma^2 = \frac{1}{n}\sum_{i=1}^{n} x_i^2 - \left(\frac{1}{n}\sum_{i=1}^{n} x_i\right)^2$$

- potrzebujemy sumę liczb i sumę ich kwadratów - to się da wyliczyć w jednej pętli cztając dane
- `http://www.oa.uj.edu.pl/~soida/wyklady/C/progs/sr_od1.c`

regresja liniowa

- mamy ciągi $\{x_i\}$ i $\{y_i\}$
- chcemy dopasować prostą $y = ax + b$
- np. metodą najmniejszych kwadratów – minimalizujemy

$$\chi^2(a, b) = \sum_{i=1}^{n} (y_i - ax_i - b)^2$$

- dwa równania na dwie niewiadome:

$$\frac{\partial \chi^2}{\partial a} = -2 \sum_{i=1}^{n} x_i(y_i - ax_i - b) = 0$$

$$\frac{\partial \chi^2}{\partial b} = -2 \sum_{i=1}^{n} (y_i - ax_i - b) = 0$$

regresja liniowa

- rozwiązanie:

$$a = \frac{n\sum_{i=1}^{n}(x_i y_i) - \sum_{i=1}^{n} x_i \sum_{i=1}^{n} y_i}{n\sum_{i=1}^{n} x_i^2 - (\sum_{i=1}^{n} x_i)^2}$$

$$b = \frac{\sum_{i=1}^{n} x_i^2 \sum_{i=1}^{n} y_i - \sum_{i=1}^{n} x_i \sum_{i=1}^{n}(x_i y_i)}{n\sum_{i=1}^{n} x_i^2 - (\sum_{i=1}^{n} x_i)^2}$$

- przykładowe dane:
  `http://www.oa.uj.edu.pl/~soida/wyklady/C/progs/linreg.dat`

funkcje, rekurencja – silnia

- napiszmy program liczący silnię dla podanej liczby:

$$n! = \prod_{i=1}^{n} i$$

funkcje, rekurencja – silnia

- napiszmy program liczący silnię dla podanej liczby:

$$n! = \prod_{i=1}^{n} i$$

- a może skorzystać z definicji silni:

$$n! = \begin{cases} 1 & \text{dla} \quad n = 0 \\ n \cdot (n-1)! & \text{dla} \quad n > 0 \end{cases}$$

funkcje, rekurencja – ciąg Fibonacciego

- definicja ciągu Fibonacciego:

$$F_n = \begin{cases} 0 & \text{dla} \quad n = 0 \\ 1 & \text{dla} \quad n = 1 \\ F_{n-1} + F_{n-2} & \text{dla} \quad n > 1 \end{cases}$$

funkcje, rekurencja – dwumian Newtona

- definicja $\binom{n}{k}$:

$$(x+y)^n = \sum_{k=0}^{n} \binom{n}{k} x^k y^{n-k}$$

- wzór rekurencyjny:

$$\binom{n}{k} = \begin{cases} 1 & \text{dla} \quad n = 0,\ k = 0,\ k = n \\ \binom{n-1}{k-1} + \binom{n-1}{k} & \text{dla} \quad n > 0,\ 0 < k < n \end{cases}$$

- wzór iteracyjny:

$$\binom{n}{k} = \frac{n!}{k! \cdot (n-k)!}$$

funkcje, rekurencja – dwumian Newtona

- lepiej liczyć tak:

$$\binom{n}{k} = \frac{\prod_{i=m}^{n} i}{\prod_{i=1}^{l} i}, \; m = \max(k, n-k) + 1, \; l = \min(k, n-k)$$

- i zliczać sukcesywnie iloczyn w liczniku i od razu dzielić go przez kolejny czynnik w mianowniku
- a co z ewentualnym dzieleniem stałopozycyjnym?!

funkcje, rekurencja – dwumian Newtona

- lepiej liczyć tak:

$$\binom{n}{k} = \frac{\prod_{i=m}^{n} i}{\prod_{i=1}^{l} i}, \; m = \max(k, n-k) + 1, \; l = \min(k, n-k)$$

- i zliczać sukcesywnie iloczyn w liczniku i od razu dzielić go przez kolejny czynnik w mianowniku
- a co z ewentualnym dzieleniem stałopozycyjnym?!
- nic – jest OK.

problemy z kalendarzem

- wyznacz dzień tygodnia dla podanej daty (dzień, miesiąc, rok)
- na początek tylko dla XXI wieku
- wiemy, że 1 stycznia 2001 to był poniedziałek
- kalendarz gregoriański:
  - co czwarty rok jest przestępny, ale
  - lata „pełne setki" nie są przestępne, ale
  - co czwarta „pełna setka" jest przestępna
  - przykładowe rozwiązanie:
    `http://www.oa.uj.edu.pl/~soida/wyklady/C/progs/dtyg.c`

idą święta...

- narysuj choinkę na terminalu
- choinka ma dowolną ilość „pięter” i podstawkę
  - piętro składa się z n rzędów gwiazdek tworzących trapez o górnej podstawie $2\lfloor n/2 \rfloor - 1$, w każdym niższym rzędzie o 2 gwiazdki więcej
  - podstawka to 3 gwiazdki
- wygodnie jest napisać funkcję rysującą jedno piętro o zadanej wysokości i górnej podstawie
- przykładowa choinka z pięcioma piętrami $\longrightarrow$

```
        *
       ***
        *
       ***
      *****
       ***
      *****
     *******
    *********
       ***
      *****
     *******
    *********
   ***********
      *****
     *******
    *********
   ***********
  *************
 ***************
       ***
```

trochę zabawy... – gra „życie”

- mamy dwuwymiarowy świat komórek z „żyjątkami”
- w komórce może być tylko jedno żyjątko (lub może go nie być)
- swiat z żyjątkami ewoluuje z pokolenia na pokolenie (skokowo)
- reguły ewolucji:
  - jeśli żyjątko ma miej niż dwóch sąsiadów – ginie (z samotności)
  - jeśli żyjątko ma więcej niż trzech sąsiadów – ginie (z przeludnienia)
  - jeśli komórka ma dokładnie trzech sąsiadów – rodzi się w niej żyjątko

trochę zabawy... – gra „życie”

- mamy dwuwymiarowy świat komórek z „żyjątkami”
- w komórce może być tylko jedno żyjątko (lub może go nie być)
- swiat z żyjątkami ewoluuje z pokolenia na pokolenie (skokowo)
- reguły ewolucji:
  - jeśli żyjątko ma miej niż dwóch sąsiadów – ginie (z samotności)
  - jeśli żyjątko ma więcej niż trzech sąsiadów – ginie (z przeludnienia)
  - jeśli komórka ma dokładnie trzech sąsiadów – rodzi się w niej żyjątko
- przydatne informacje:
  - stan populacji przechowujemy w tablicy (2D)
  - stan populacji możemy wypisać w terminalu
  - napis `"\033[H\033[J"` czyści terminal i ustawia kursor w lewym górnym rogu (dla większości terminali)
  - funkcja `usleep(usec)` zatrzymuje program na `usec` μs

trochę zabawy... – gra „życie”

- przydatne informacje:
  - stan populacji przechowujemy w tablicy (2D)
  - stan populacji możemy wypisać w terminalu
  - napis `"\033[H\033[J"` czyści terminal i ustawia kursor w lewym górnym rogu (dla większości terminali)
  - funkcja `usleep(usec)` zatrzymuje program na `usec` μs
- do zaprogramowania mamy trzy części:
  1. wygenerować początkową populację
  2. ze „starej” populacji wyliczyć „nową”
  3. pokazać stan populacji
  - ostatnie dwa punkty wykonujemy w nieskończonej pętli

kilka operacji na tablicach (wektorach)

iloczyn skalarny wektorów

$$\vec{a} \cdot \vec{b} = \sum_{i=1}^{n} a_i b_i$$

iloczyn wektorowy wektorów (3D)

$$(\vec{a} \times \vec{b})_i = \sum_{j=1}^{3} \sum_{k=1}^{3} \epsilon_{ijk} a_j b_k$$

$$\begin{aligned} (\vec{a} \times \vec{b})_1 &= a_2 b_3 - a_3 b_2 \\ (\vec{a} \times \vec{b})_2 &= a_3 b_1 - a_1 b_3 \\ (\vec{a} \times \vec{b})_3 &= a_1 b_2 - a_2 b_1 \end{aligned}$$

kilka operacji na tablicach (wektorach)

- a teraz zróbmy z tych operacji funkcje:
  - iloczyn skalarny: `dot(a,b)` zwraca wartość iloczynu
  - iloczyn wektorowy: `cross(a,b)` – czy tak?
- jak zwrócić „wartość” wektora?

kilka operacji na tablicach (wektorach)

- a teraz zróbmy z tych operacji funkcje:
  - iloczyn skalarny: `dot(a,b)` zwraca wartość iloczynu
  - iloczyn wektorowy: `cross(a,b)` – czy tak?
- jak zwrócić „wartość" wektora?
- można tak: jeśli $c = a \times b$ to wywołujemy: `cross(c,a,b)`
- ale chciałoby się: `c=cross(a,b)`

kilka operacji na tablicach (macierzach)

mnożenie macierzy

$$(AB)_{i,j} = \sum_{k=1}^{n} a_{i,k} b_{k,j}$$

wyznacznik macierzy

$$|A| = \sum_{i=1}^{n} (-1)^i a_{i,1} |A_{i,1}|$$

kilka operacji na tablicach (macierzach)

mnożenie macierzy

$$(AB)_{i,j} = \sum_{k=1}^{n} a_{i,k} b_{k,j}$$

wyznacznik macierzy

$$|A| = \sum_{i=1}^{n} (-1)^i a_{i,1} |A_{i,1}|$$

- przekazanie tablicy więcej niż 1-wymiarowej do funkcji nie jest proste
- zamiast np. dwuwymiarowej `a[i][j]` lepiej użyć jednowymiarowej `a[i*N+j]` (`N` – długość wiersza)

trochę numeryki – wyznaczanie miejsca zerowego funkcji

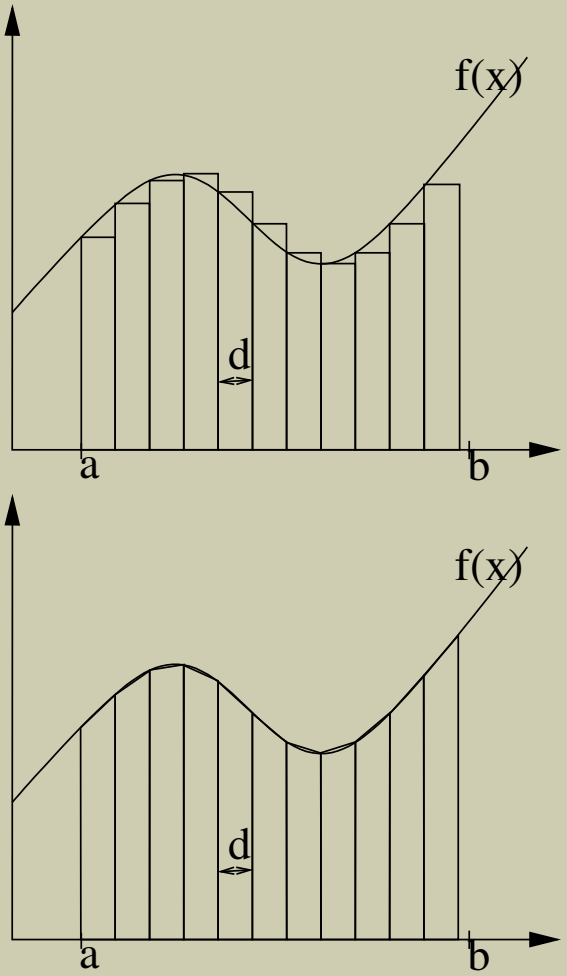
- należy wyznaczyć miejsce zerowe zadanej funkcji w zadanym przedziale
- jak to w numeryce bywa – potrzeba też zadać wymaganą dokładność wyniku
- wykonamy to zadanie metodą bisekcji
  - bisekcja – czyli podział na dwie części
  - w kolejnych krokach pętli poszukujemy miejsca zerowego w jednej połówce przedziału
- problemy do rozwiązania:
  1. jak sprawdzić czy w przedziale jest miejsce zerowe
  2. po podziale przedziału – w której połowce się znajduje
  3. kiedy osiągnęliśmy wymaganą dokładność (kiedy skończyć program)
  4. co jest wynikiem (która wartość w przedziale)

trochę numeryki – wyznaczanie miejsca zerowego funkcji

- odpowiedzi na pytania:
  1. jeśli wartości funkcji na końcach przedziału mają różne znaki
  2. w jednym z dwóch – tym na którego końcach funkcja ma różne znaki
  3. gdy długość przedziału w którym jest miejce zerowe nie jest większa od zadanej dokłądności
  4. dowolna wartość z przedziału – np. jego środek lub dowolny koniec
- dodatkowe podpowiedzi i uwagi:
  - dwie wartości mają różne znaki jeśli ich iloczyn jest ujemny
  - jeśli w zadanym przedziale, wartości funkcji na jego końcach są tego samego znaku to nie potrafimy tą metodą wyznaczyć miejsca zerowego (mimo, że może tam być) – informujemy użytkownika i kończymy program
  - wyznaczymy tylko jedno miejce zerowe w danym przedziale mimo, że może ich być więcej
  - trochę zaoszczędzimy czasu obliczeń jeśli wyliczamy wartość funkcji tylko raz dla tego samego argumentu (tak jak w moim rozwiązaniu)
  - `http://www.oa.uj.edu.pl/~soida/wyklady/C/progs/zero.c`

trochę numeryki – obliczenie całki zadanej funkcji

- oblicz całkę zadanej funkcji w wyznaczonym przedziale
- możemy przybliżyć wartość całki (pole pod wykresem funkcji) sumując pola prostokątów o stałej podstawie (tzw. krok całkowania) i wysokości równej wartości funkcji
- lepszym przybliżeniem będzie suma trapezów o stałej wysokości i podstawach równych wartościom funkcji na końcach przedziałów
- długość przedziału całkowania `[a,b]` nie musi być całkowitą wielokrotnością kroku całkowania



- `http://www.oa.uj.edu.pl/~soida/wyklady/C/progs/calka.c`

trochę numeryki – całkowanie równania różniczkowego

- jedno z najprostszych równań – równanie rozpadu $\frac{dm}{dt} = -\frac{m}{\tau}$
- wiemy, że rozwiązaniem jest $m(t) = m_0 e^{-t/\tau}$
- możemy obliczyć rozwiązanie (masę jako funkcję od czasu) w poszczególnych krokach czasowych (co Δt) „startując" od momentu $t = 0$ stosując przybliżenie $\Delta m = -\frac{m}{\tau}\Delta t$
- $m(t + \Delta t) = m(t) - \frac{m(t)}{\tau}\Delta t$
- z warunkiem początkowym (dla $t = 0$) $m(0) = m_0$
- to co powyżej to tzw. schemat Eulera
- intersujące jest porównanie naszego rozwiązania z dokładnym

problem „z innej beczki”

- znając położenie obiektu na niebie (rektascensję i deklinację) oblicz przedział czasu gwiazdowego, kiedy obiekt jest powyżej pewnej wysokości nad horyzontem (na danej szerokości geogr.)
- potrzebne wzory:

$$\cos t = \frac{\cos z - \sin \phi \sin \delta}{\cos \phi \cos \delta}$$

$$\theta = \alpha - t$$

podobny problem (δ,t)$\rightarrow$(A,z)

- znamy kąt godzinny i deklinację oblicz azymut i wysokość (odl. zenitalną), dla danej szerokości geograficznej
- potrzebne wzory:

$$\cos z = \sin\delta \sin\phi + \cos\delta \cos\phi \cos t$$

$$\sin z = \sqrt{1 - \cos^2 z}$$

$$\sin A = \cos\delta \sin t / \sin z$$

$$\cos A = \frac{\sin\phi \cos z - \sin\delta}{\cos\phi \sin z}$$

operacje na plikach

- jeśli trzeba tylko coś przeczytać z jednego pliku lub zapisać do jednego pliku – wystarczy standardowe wejście i wyjście i przekierowanie
- jeśli korzystamy z większej ilości plików trzeba skorzystać z funkcji działających na plikach
- program (w języku C) operuje na pliku za pomocą *deskryptora pliku* – wskaźnika do struktury nazwanej `FILE` (zdefiniowanej w pliku nagłówkowym `stdio.h`)
- do formatowanego czytania i pisania należy używać funkcji `fscanf()` i `fprintf()`, mają one jeden argument więcej (pierwszy) – deskryptor pliku
- poza tymi (i innymi) funkcjami konieczne jest użycie funkcji `fopen()` do inicjalizacji deskryptora (*otwarcia* pliku) i `fclose()` do zakończenia działania na pliku (*zamknięcia*)

operacje na plikach – przykłady

czytanie z pliku

```
#include <stdio.h>
int main(){
FILE *fd;
double x,y;
/* ...*/
fd=fopen("plik.dat","r");
fscanf(fd,"%lf%lf",&x,&y);
/* ...*/
fclose(fd);
}
```

pisanie do pliku

```
#include <stdio.h>
int main(){
FILE *fd;
double x=2.,y=3.;
/* ...*/
fd=fopen("plik.dat","w");
/*              lub  "a"    */
fprintf(fd,"%lf %lf\n",x,y);
/* ...*/
fclose(fd);
}
```

uwagi n/t funkcji `fopen()`

tryb otwarcia	plik istnieje	plik nie istnieje
`"r"`	czytanie pliku od początku	błąd
`"w"`	zapisywanie pliku „od początku” oryginalna zawartość jest tracona	zapisywanie do pliku (tworzony nowy plik)
`"a"`	dopisywanie do pliku po oryginalnej zawartości	zapisywanie do pliku (tworzony nowy plik)

- jeśli z jakiegoś powodu nie można otworzyć pliku `fopen()` zwraca pusty wskaźnik (`NULL`)
- pliku nie można otworzyć np. gdy:
  - nie mamy praw do czytania/pisania z/do pliku
  - nie mamy prawa zapisu w katalogu w którym chcemy otworzyć plik do zapisu
  - nie istnieje wskazany katalog (w nazwie), który otwieramy do czytania/zapisu
- operacje na deskryptorze plików po niepowodzeniu otwarcia kończy się błędem naruszenia ochrony pamięci (warto sprawdzić poprawność otwarcia – użyteczna fukncja `perror()`)

uwagi n/t funkcji `fclose()`

- system operacyjny zamyka otwarte pliki przy poprawnie kończącym się programie
- pliku „do czytania” w zasadzie nie ma potrzeby zamykać
- pliku „do zapisu” właściwie też nie, ale
  - tylko jeśli program kończy się poprawnie
  - jeśli wystąpił błąd wykonania zapisywany plik zwykle jest „urwany” (lub nawet pusty)
- zamykanie już zamkniętego pliku jest błędem
- można ponownie otworzyć plik, ale najpierw trzeba go zamknąć
- istnieją trzy deskryptory plików zawsze „otwarte” i powiązane z programem: `stdin`, `stdout` i stderr – standardowe wejście, wyjście i „wyjście błędów” - nie należy ich zamykać

komunikacja z systemem operacyjnym

- argumenty wywołania programu
  - `int main(int arc, char *argv[])`
  - `argv[]` jest tablicą napisów – argumentów wywołania
- funkcja: `int system(char *)`
  - zadeklarowana w pliku nagłówkowym `stdlib.h`
  - argument (tekst) jest wykonywany jako polecenie systemu operacyjnego
  - wartością funkcji jest wartość zwracana przez to polecenie
- `main()` zwraca wartość `int`, więc można użyć `return <wartość>`
  - wartość argumentu jest przekazywana procesowi, który wywoływał nasz program
  - zwykło się traktować zwracaną wartość „0” jako poprawne zakończenie programu
  - jeśli program wywoływaliśmy z powłoki (lini poleceń terminala) to tą wartość można wykorzystać w tej powłoce – tworzenie tzw. „skryptów” (zestawów poleceń powiązanych różnymi zależnościami)