Wstęp do programowania

Pliki

Pamięci zewnętrzne

- Pamięć wewnętrzna (RAM) jest jednorodna. Dostęp do każdej komórki jest szybki i kosztuje tyle samo
- W przypadku pamięci zewnętrznych tak nie jest.
- Szybkość dostępu zależy od
  - rodzaju nośnika
  - sposobu wypełnienia nośnika
  - tego czy zapisujemy, czy odczytujemy informację
- Szybkośc dostępu do pamięci zewnętrznej jest o rzędy wielkości mniejsza niż dla RAM.

Bufory

- Bez wyjątku transmisja danych do RAM i z RAM odbywa się za pomocą buforów.
- Buforowanie danych powoduje m.in. to, że przesłanie pojedynczego nawet bitu powoduje zazwyczaj transmisję calego segmentu danych – system operacyjny zaklada, że zapewne będziemy zaraz potrzebowali kolejnej danej i na wszelki wypadek ściąga tyle, ile mieści się do bufora.

HDD

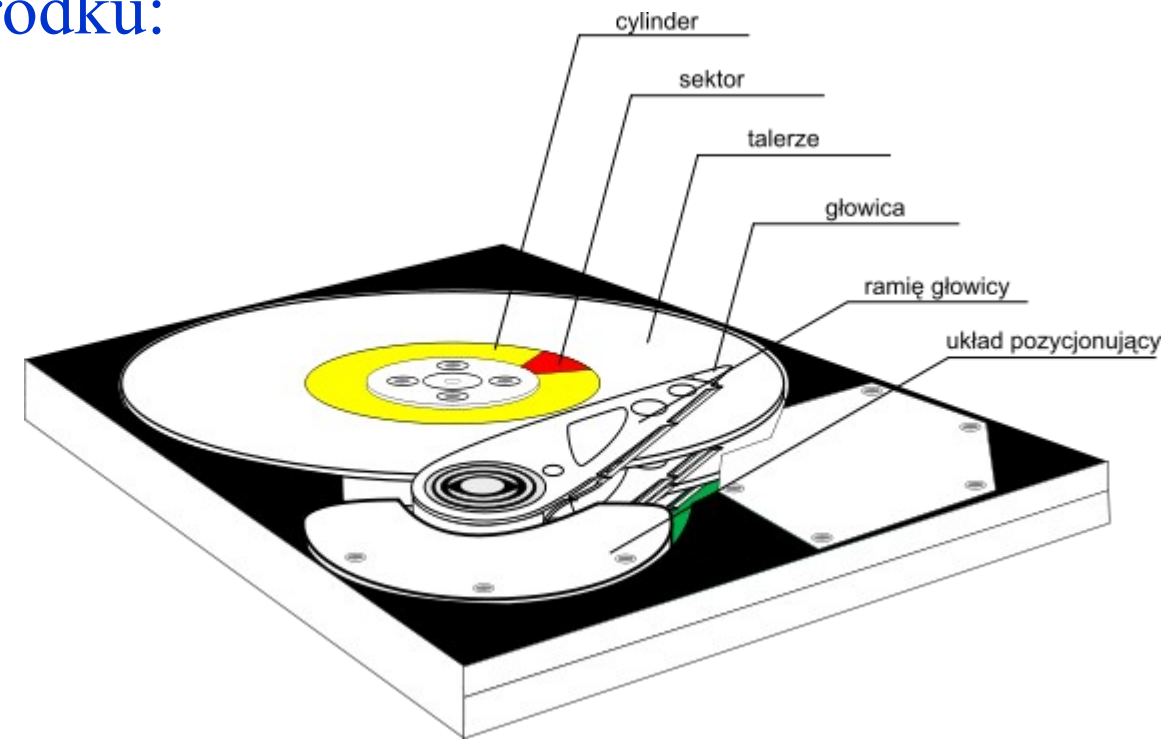
Twarde dyski są najpopularniejszym rodzajem pamięci zewnętrznej, choć pamięć fleszowa jest poważną konkurencją

Składają się z:

- talerzy
- cylindrów
- ścieżek
- sektorów

HDD - wygląd

- W środku:



Autor rysunku:Artur Kruszewski

Typ plikowy

- `type plik=file of typ_plikowy;`
- ... gdzie typ plikowy jest dowolnym typem poza
  - innym typem plikowym
  - wskaźnikiem

Praca z plikami

- `type plik=file of typ_plikowy;`
- Aby zacząć pracę z plikiem trzeba
  - zadeklarować zmienną typu plikowego
  - związać tę zmienną z konkretnym plikiem fizycznym
  - otworzyć plik wskazując tryb pracy (do zapisu lub odczytu)
  - ... a po zakończeniu pracy zamknąć plik
- Te wszystkie rzeczy robuimy z poziomu języka programowania, ale zazwyczaj zrealizowane są za pomocą specyficznych procedur związanych z daną realizacją języka

Praca z plikami - przykład

```
var f:file of Integer; n:Integer;
    A:array[1..maxn]:Integer;
begin
 Assign(f,'c:\dane.dat');
 Reset(f); n:=0;
 while not eof(f) do
  begin
   n:=n+1;
   Read(f,A[n]);
  end;
 Close(f);
 ...
end;
```

Dostępne procedury

- Dostępne są następujące procedury:
  - Assign(f,s)
  - Reset(f) przygotowanie pliku do odczytu
  - Rewrite(f) przygotowanie pliku do zapisu
  - eof(f) – funkcja mówiąca, czy jesteśmy na końcu pliku
  - Close(f) – procedura zwalniająca bufor
  - Read(f,dana)
  - Write(f,dana)

Pliki tekstowe

- Szczególnym przykładem plików są pliki tekstowe (dawniej `text = file of char`, dziś po prostu `text`)
- Pliki te mają strukturę wierszową. Klawisz <enter> powoduje zapisanie końca wiersza. Zazwyczaj jest to para znaków #13 #10.
- W plikach tekstowych dostępna jest funkcja Eoln(f), której wartością jest true, jeśli stoimy na końcu wiersza oraz procedury `Readln` i `Writeln`.
- Eof(f) → Eoln(f)

Pliki tekstowe

- Szczególnym przykładem plików są pliki tekstowe (dawniej `text = file of char`, dziś po prostu `text`)
- Pliki te mają strukturę wierszową. Klawisz <enter> powoduje zapisanie końca wiersza. Zazwyczaj jest to para znaków #13 #10.
- W plikach tekstowych dostępna jest funkcja Eoln(f), której wartością jest true, jeśli stoimy na końcu wiersza oraz procedury `Readln` i `Writeln`.
- Eof(f) → Eoln(f)

Schemat przetwarzania pliku ogólnego

```
var
   f:file of integer;
   zmienna:integer;
begin
   Assign(f, nazwa_pliku);
   Reset(f);
   while not Eof(f) do
     begin
       Read(f, zmienna);
       Przetwórz(zmienna)
     end;
   Close(f);
end.
```

Schemat przetwarzania pliku tekstowego

```
var f:text; ch:char;
begin
  Assign(f, nazwa_pliku); Reset(f);
  while not Eof(f) do  begin
      while not Eoln(f) do
        begin
          Read(f, ch)
          Przetwórz(ch)
        end;
      if not Eof(f) then Readln(f);
    end;
  Close(f);
end.
```

Przykład kłopotliwego zadania

- Usuń z posortowanego niemalejąco pliku f wszystkie wartości występujące w posortowanym niemalejąco pliku g i wynik zapisz w pliku h.
- Problem polega na tym, że aby przetworzyć dańć musimy ją przeczytać wpierw upewniwszy się, że plik się jeszcze nie skończył

Przykład kłopotliwego zadania

```
var  f,g,h:file of Integer; fx,gx:Integer;
begin
  Assign(f, nazwa_pliku1); Assign(g, nazwa_pliku2);
  Reset(f); Reset(g); Rewrite(h);
  if eof(g) then {Niczego nie wyrzucamy z f; kopiujemy wszystko
     do h}
       while not(eof(f) do
         begin
          Read(f,fx);
          Write(h,fx)
         end
  else if not eof(f) then
```

Przykład kłopotliwego zadania - cd.

```
begin Read(f,fx);  Read(g,gx);
  {teraz jesteśmy gotowi wystartować z
  porównywaniem decydującym o tym, który
  plik awansujemy}
  while (not Eof(f)) and (not eof(g)) do
     if fx < gx then begin
                          Write(h,fx); {nie ma
  w g elementu fx}
                          Read(f,fx)
                        end
     else if fx > gx then Read(g,gx) {stary
  element fg już był bezużyteczny}
          else {fx=gx}    Read(f,fx) {stary
  fx nie powinien być przepisany do h} {...}
```

Przykład kłopotliwego zadania - cd.

```
if eof(f) then {trzeba sprawdzić, czy w
  pliku g nie ma wartości fx} begin
  while (gx < fx) and (not eof(g) do
  Read(g,gx);
  if gx<>fx then Write(h,fx)
 end
else {w pliku g już nie ma niczego, więc
  tylko wartość gx trzeba wykluczyć z f}
 begin
  while (fx <> gx) and (not eof(f)) do
    begin
      Write(h,fx);
      Read(f,fx)
    end;
```

Przykład kłopotliwego zadania - zakończenie.

```
while (fx = gx) and (not eof(f)) do
Read(f,fx); {elementów równych gx nie wypisujemy}
  while not eof(f) do {kopiujemy resztę pliku f -
  elementy większe od gx}
    begin
      Write(h,fx);
      Read(f,fx)
    end;
   if fx<>gx then Write(h,fx)  {i nie zapominamy o
  ostatnim elemencie}
 end
Close(f);Close(g); Close(h);
end; {Uff!}
```

Wirtualne czytanie

- W naszym przyładzie przyjmijmy, że
  - Istnieje MAXINT
  - W plikach f,g nie ma MAXINT
- Pokażemy, jak można uprościć nasze zadanie za pomocą wirtualnej procedury czytającej

Wirtualne czytanie

```
procedure Czytaj(var p:plik; var
  x:Integer);
{procedura przypisuje zmiennej x
  wartość MAXINT, jeśli jesteśmy na
  końcu pliku lub czyta z pliku p
  kolejną wartość i podaje ją jako
  wartość zmiennej x}
begin
 if eof(p) then x:=MAXINT
 else Read(p,x)
end;
```

... i teraz

```
begin
Czytaj(f,fx); Czytaj(g,gx);
while (fx < MAXINT) >= (gx < MAXINT) do
  if fx < gx then begin
   Write(h,fx); {nie ma w g elementu fx}
   Czytaj(f,fx)
  end
  else if fx > gx then Czytaj(g,gx) {stary
  element fg już był bezużyteczny}
          else {fx=gx}    Czytaj(f,fx} {stary
  fx nie powinien być przepisany do h}
Close(f);Close(g); Close(h);
end;
```

i w końcu...

```
Close(Wykład)
```