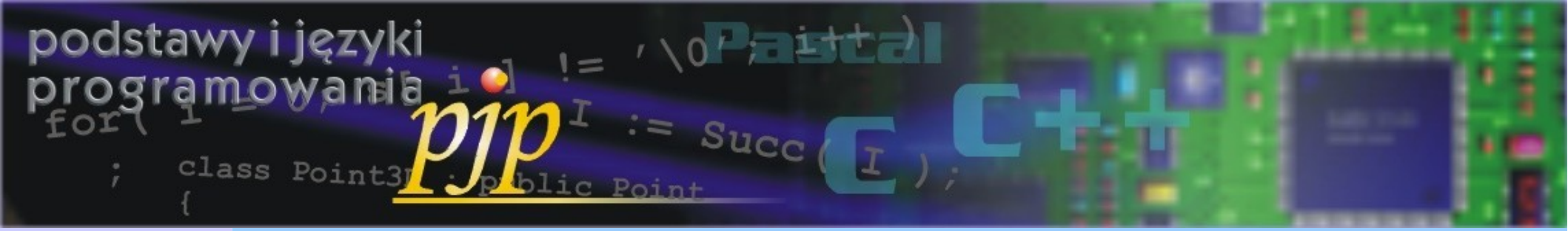


Podstawy programowania w języku C++

Część trzynasta

Tablice struktur, pliki struktur

Autor

Roman Simiński

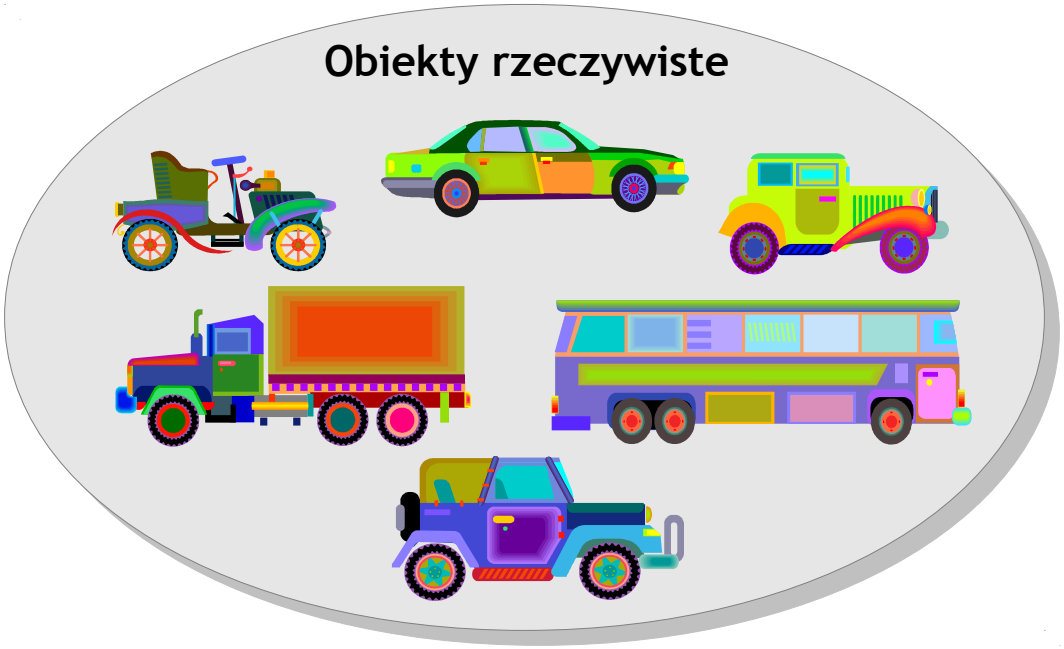
Kontakt

roman.siminski@us.edu.pl
www.programowanie.siminskionline.pl

System ewidencji pojazdów dla autokomisu

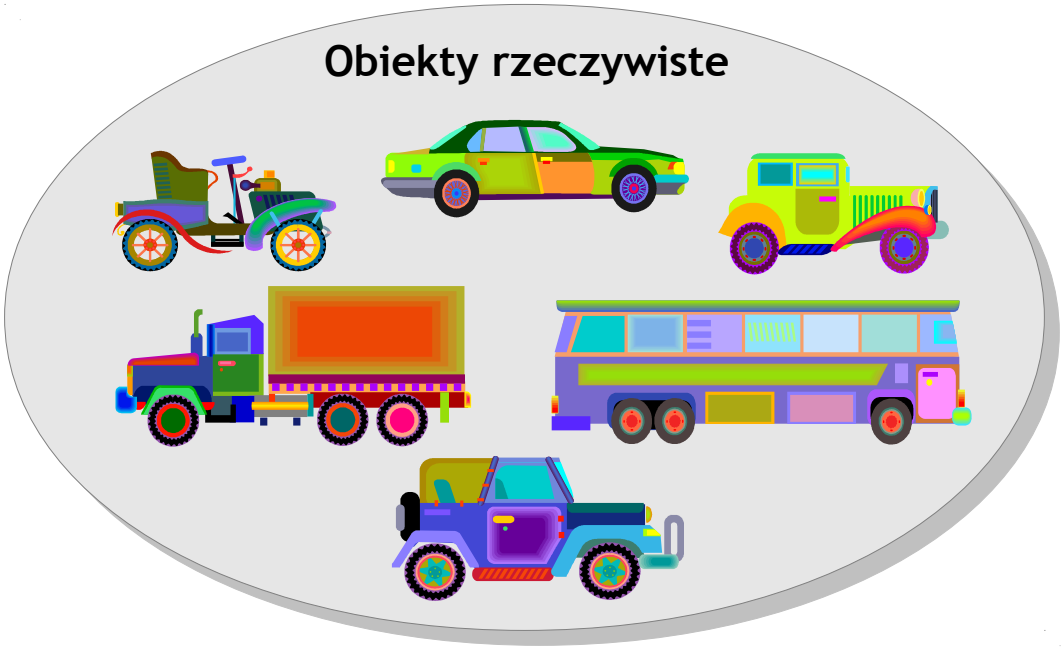
Jakich danych potrzebujemy?

Analityk i/lub projektant systemu informatycznego



Jakie informacje będziemy przetwarzać i przechowywać?

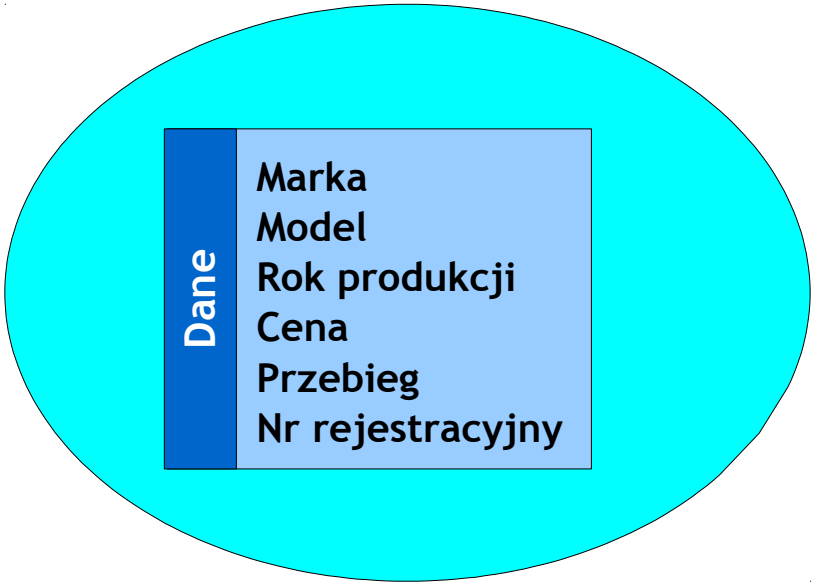
Analityk i/lub
projektant systemu
informatycznego



Dane

- Marka
- Model
- Rok produkcji
- Cena
- Przebieg
- Nr rejestracyjny

Dane opisują jeden pojazd



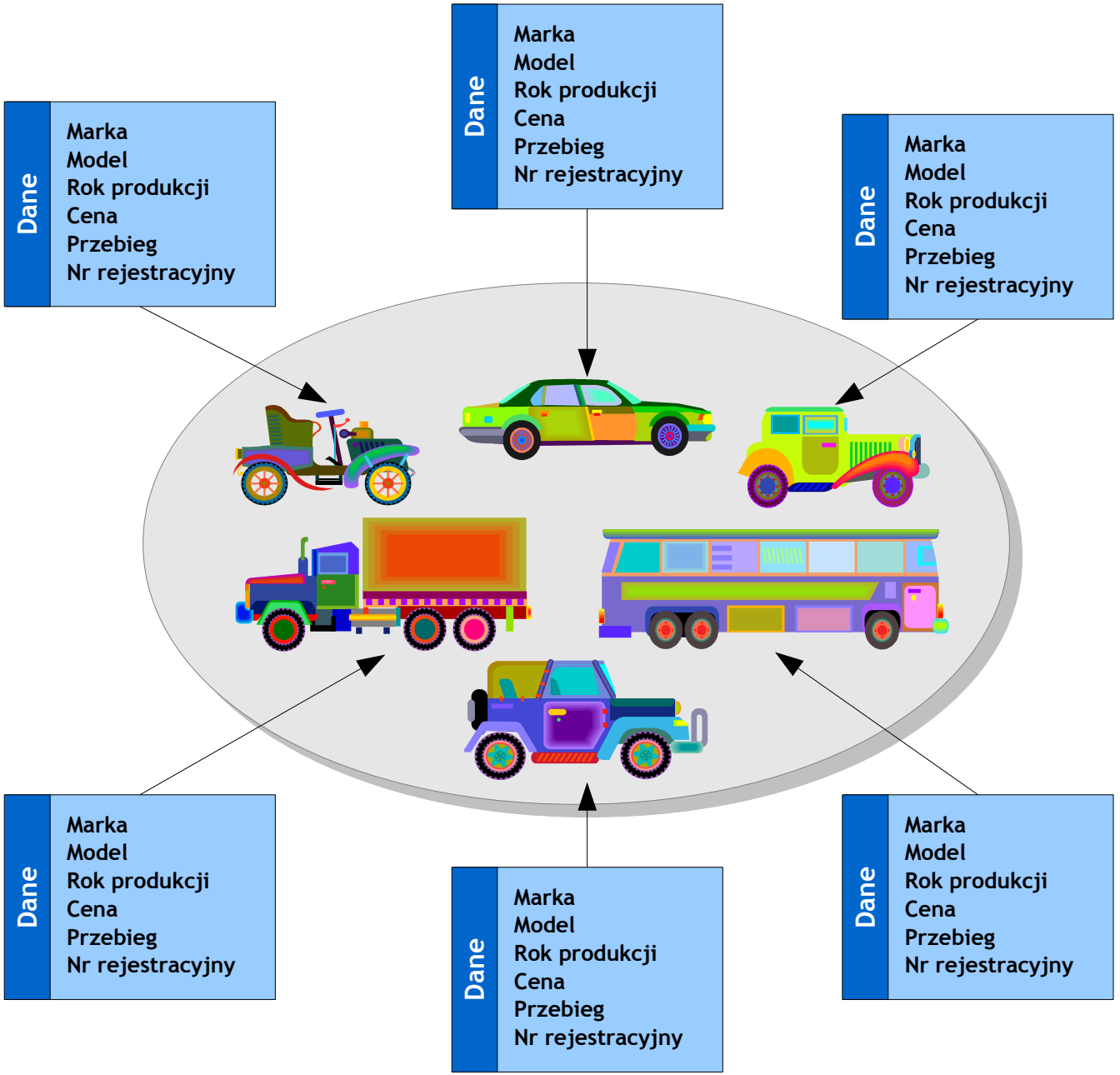
Dane opisujące jeden pojazd to porcja różnych informacji

Analityk i/lub projektant systemu informatycznego

Pojazdów jest wiele...

Potrzeba wiele porcji danych.

Każda z porcji jest złożona i zawiera różne dane opisujące pojazd.



Definicja typu strukturalnego raz jeszcze

```
/* Maksymalna dlugosc pol marka i model */
#define MAKS_M 20

/* Maksymalna dlugosc pola numeru rejestracyjnego */
#define MAKS_R 10

/* Deklaracja typu strukturalnego, opisu informacji o pojezdzie */
struct _pojazd
{
  char  marka[ MAKS_M ];
  char  model[ MAKS_M ];
  short int rok_prod;
  float cena;
  float przebieg;
  char  nr_rej[ MAKS_R ];
};

typedef struct _pojazd pojazd;
```

Typ *short int* jest reprezentowany w postaci 16-to bitowej *liczby ze znakiem* zarówno w kompilatorach 16-to jak i 32-u bitowych. Taka deklaracja poprawi *przenośność kodu* programu oraz pliku z danymi.

Potrzebujemy funkcji do odczytu/zapisu rekordu z stdio

```
pojazd a;

/* Przykladowe dane */
strcpy( a.marka, ”Honda” );
strcpy( a.model, ”Accord” );
a.rok_prod = 2006;
a.przebieg = 32850.5;
a.cena = 45000;
strcpy( a.nr_rej, ”S1 XXXX” );

pokaz_info( &a );
```

```
      Marka: Honda
      Model: Accord
  Rok prod.: 2006
       Cena: 32850.5
   Przebieg: 45000
Nr rejestr.: S1 XXXX
```

```
pojazd a;

czytaj_info( &a );
. . .

if( zmiana_rocznika ) /* Zmniejsz cene o 10% */
  a.cena -= a.cena * 0.9;

pokaz_info( &a );
```

```
Podaj dane pojazdu
Marka: Honda
Model: Accord
Rok produkcji:
```

Funkcja wczytująca zawartość struktury z stdin

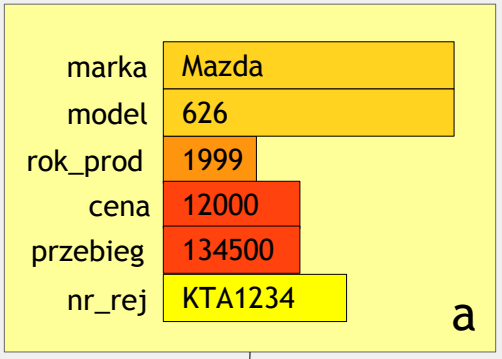
```
void czytaj_info( pojazd * info )
{
  char bufor[ 128 ];
  printf( "\nMarka: " ); gets( bufor );
  if( strlen( bufor ) >= MAKS_M ) bufor[ MAKS_M - 1 ] = '\0';
  strcpy( info->marka, bufor );

  printf( "Model: " );  gets( bufor );
  if( strlen( bufor ) >= MAKS_M ) bufor[ MAKS_M - 1 ] = '\0';
  strcpy( info->model, bufor );

  printf( "Rok produkcji: " ); gets( bufor );
  info->rok_prod = atoi( bufor );

  printf( "Cena: " ); gets( bufor );
  info->cena = atof( bufor );

  printf( "Przebieg: " ); gets( bufor );
  info->przebieg = atof( bufor );

  printf( "Numer rejestracyjny: " ); gets( bufor );
  if( strlen( bufor ) >= MAKS_R ) bufor[ MAKS_R - 1 ] = '\0';
  strcpy( info->nr_rej, bufor );
}
```

Funkcja wyprowadzająca zawartość struktury do stdout

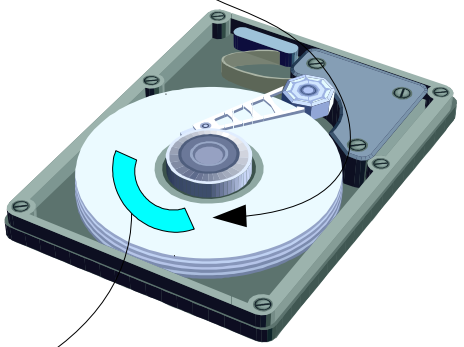
```
void pokaz_info( pojazd * info )
{
  printf( "\nMarka: %s", info->marka );
  printf( "\nModel: %s", info->model );
  printf( "\nRok produkcji: %d", info->rok_prod );
  printf( "\nCena: %g", info->cena );
  printf( "\nPrzebieg: %g", info->przebieg );
  printf( "\nNr rejestracyjny: %s", info->nr_rej );
}`
```

Zapis blokowy struktury do pliku

```
pojazd a;
FILE * f;

czytaj_info( &a );

if( ( f = fopen( "pojazdy.dat", "wb" ) ) != NULL )
{
  fwrite( &a, sizeof( pojazd ), 1, f );

  fclose( f );
}
```

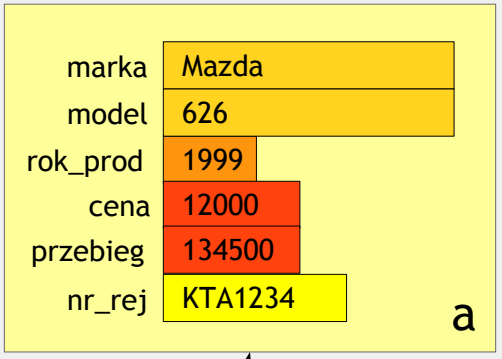



```
fwrite( &a, sizeof( pojazd ), 1, plik );
```



Odczyt blokowy struktury z pliku

```
pojazd a;
FILE * f;

if( ( f = fopen( ”pojazdy.dat”, ”rb” ) ) != NULL )
{
  fread( &a, sizeof( pojazd ), 1, f );

  pokaz_info( &a );

  fclose( f );
}
```




```
fread( &a, sizeof( pojazd ), 1, plik );
```

Odczyt struktury z pliku — wykorzystanie funkcji

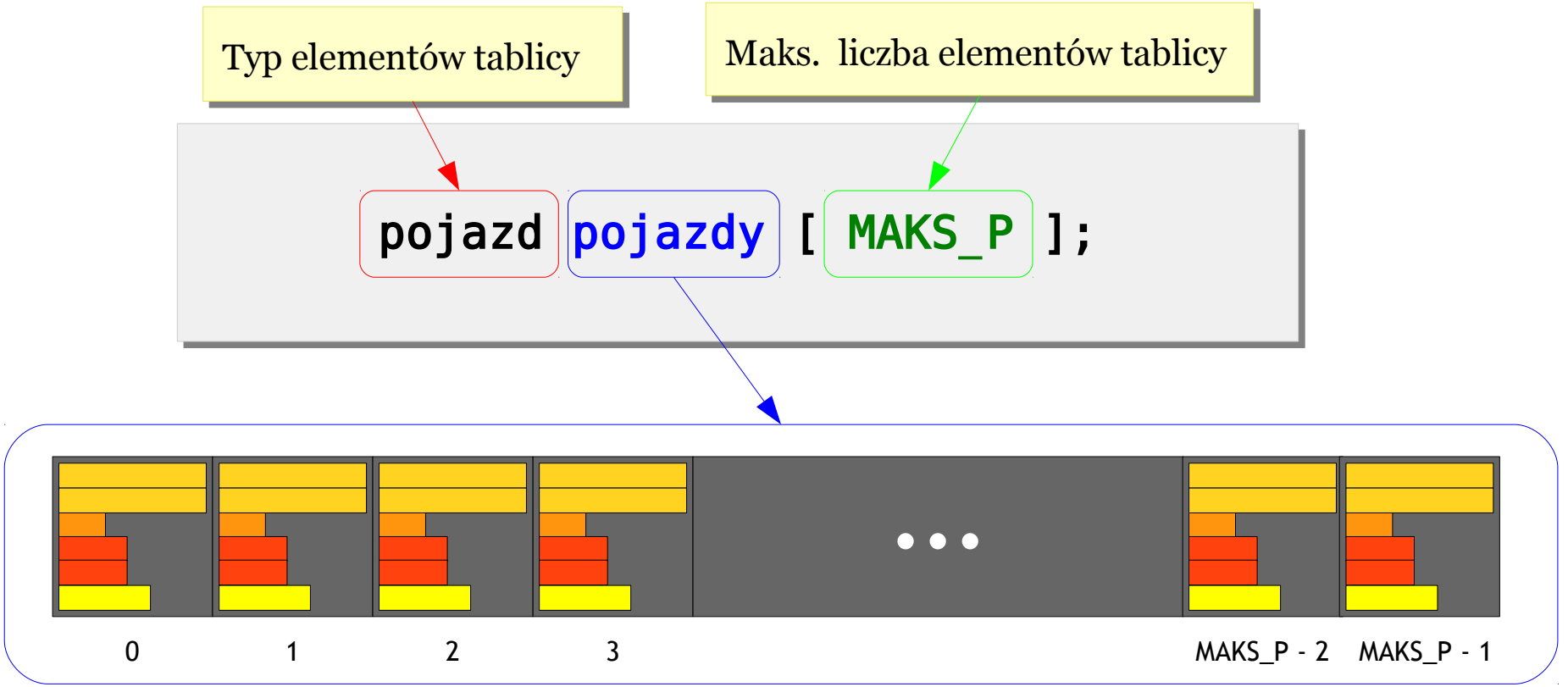
```
int info_z_pliku( pojazd * info, FILE * file )
{
  return ( fread( info, sizeof( pojazd ), 1, file ) == 1 );
}

. . .

if( ( f = fopen( ”auta.dat”, ”rb” ) ) != NULL )
{
  if( info_z_pliku( &a, f ) )
    pokaz_info( &a );
  else
    printf( ”Blad odczytu danych” );

  fclose( f );
}
```

Zapis struktury do pliku — wykorzystanie funkcji

```
int info_do_pliku( pojazd * info, FILE * file )
{
  return ( fwrite( info, sizeof( pojazd ), 1, file ) == 1 );
}

. . .

czytaj_info( &a );

if( ( f = fopen( ”auta.dat”, ”wb” ) ) != NULL )
{
  if( info_do_pliku( &a, f ) )
    printf( ”Dane zapisane poprawnie” );
  else
    printf( ”Blad zapisu danych” );

  fclose( f );
}
```

Tablica struktur — pseudotabela z danymi

```
/* Maksymalna liczba ewidencjonowanych pojazdow  */
#define MAKS_P 200

/* Tablica struktur opisujacych pojazdy */
pojazd pojazdy[ MAKS_P ];
```

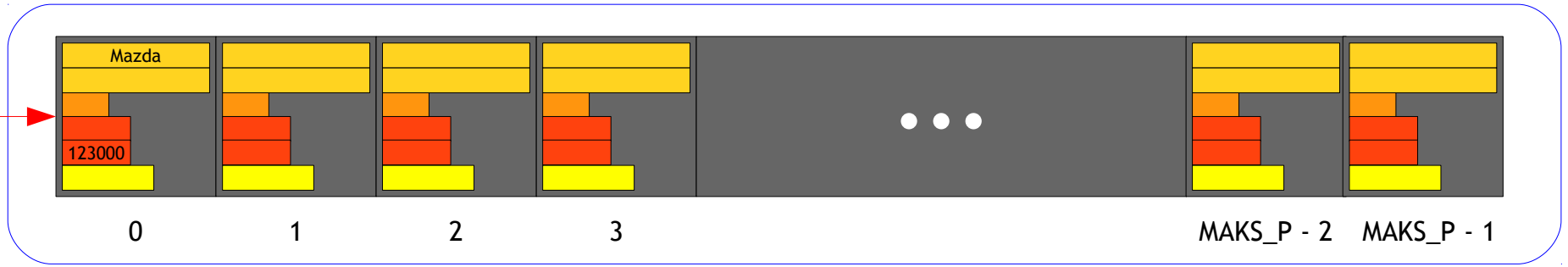


Tablica struktur

Tablica struktur — jak odwoływać się do pól struktur w tablicy?

```
/* Maksymalna liczba ewidencjonowanych pojazdow  */
#define MAKS_P 200

/* Tablica struktur opisujacych pojazdy */
pojazd pojazdy[ MAKS_P ];
```

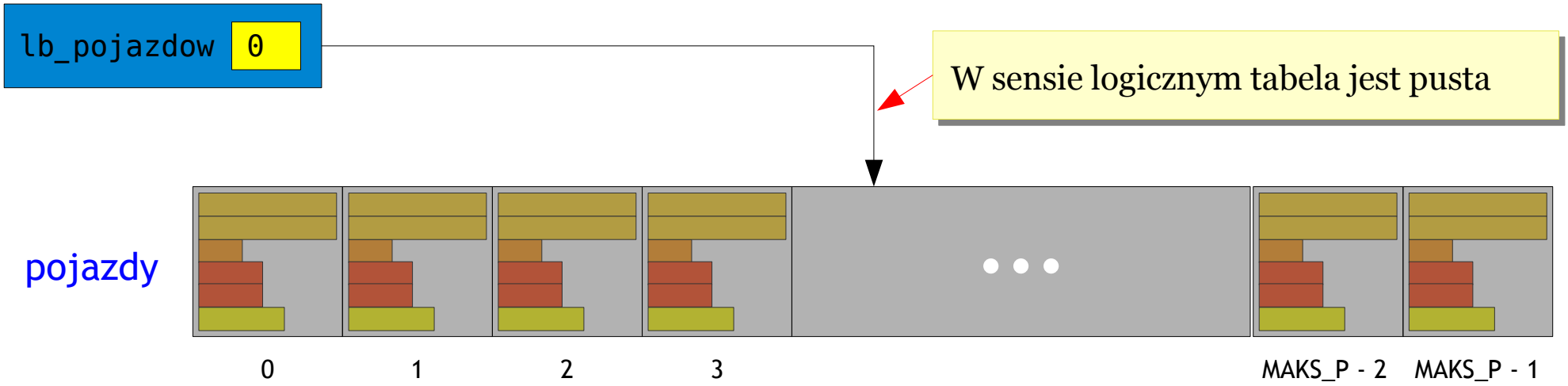
```
pojazdy[ 0 ].przebieg = 123000;
strcpy( pojazdy[ 0 ].marka, ”Mazda” );
```



Tablica struktur

Tablica struktur — pusty magazyn na dane o pojazdach

```
/* Maksymalna liczba ewidencjonowanych pojazdow  */
#define MAKS_P 200

/* Tablica struktur opisujacych pojazdy */
pojazd pojazdy[ MAKS_P ];

/* Aktualna liczba ewidencjonowanych pojazdow, domyślnie zerowana */
short int lb_pojazdow = 0;

/* Nazwa pliku danych ewidencji pojazdow */
const char nazwa_pliku[] = "pojazdy.dat";
```

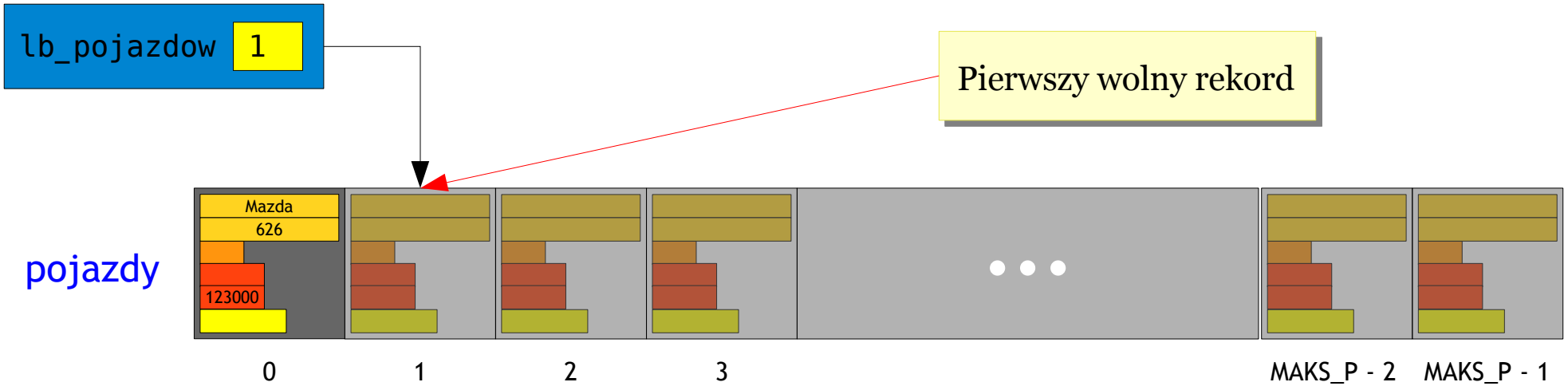


Tablica struktur — dopisanie rekordu do ewidencji

```
. . .
czytaj_info( &pojazdy[ lb_pojazdow ] );
lb_pojazdow++;
. . .
```

lub

```
. . .
czytaj_info( &pojazdy[ lb_pojazdow++ ] );
. . .
```



Tablica struktur — dopisywanie kolejnych rekordów do ewidencji

```
void dopisz_pojazd( void )
{
  int jeszcze_jeden; /* Czy wczytac dane nastepnego pojazdu? */

  if( lb_pojazdow < MAKS_P ) /* Czy jest miejsce w tablicy? */
    do
    {
      czytaj_info( &pojazdy[ lb_pojazdow ] );
      lb_pojazdow++;

      printf( "\nCzy wprowadzasz nastepny pojazd? (t/n): " );

      jeszcze_jeden = ( tolower( getchar() ) == 't' );
      fflush( stdin );

    }
    while( jeszcze_jeden && lb_pojazdow < MAKS_P );

  if( lb_pojazdow == MAKS_P ) /* Czy wyczerpano miejsce w tablicy? */
    printf( "\nEwidencja pelna!" );
}
```

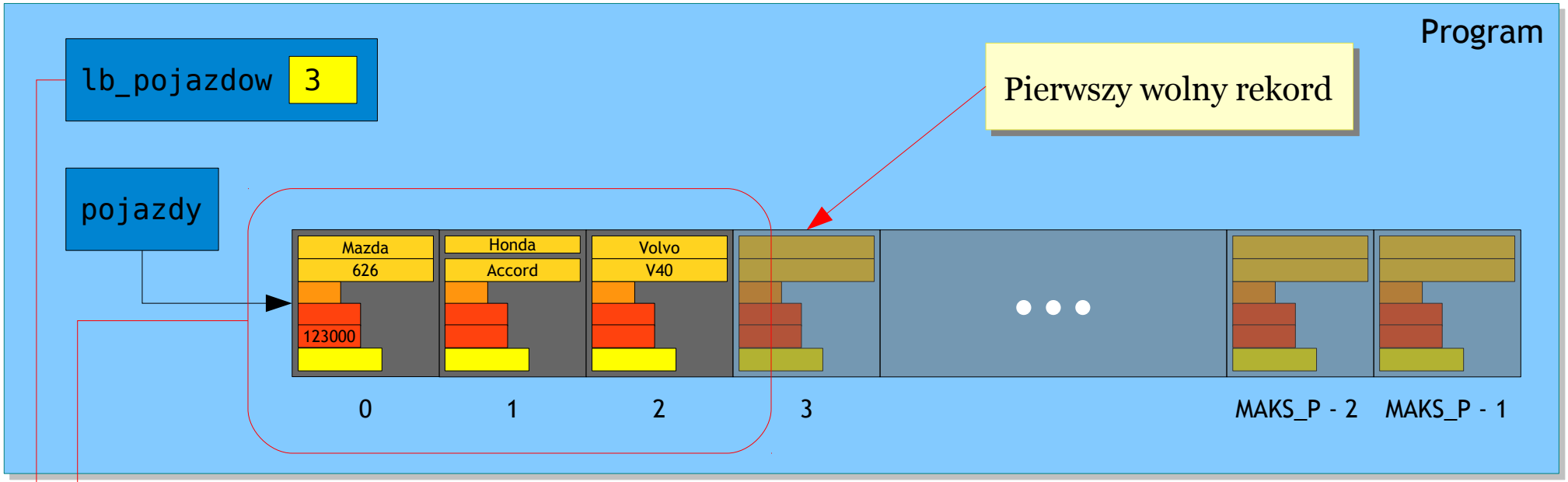
Tablica struktur — wypisywanie kolejnych rekordów z ewidencji

```
void pokaz_pojazdy( void )
{
  int nr;

  if( lb_pojazdow == 0 )
    printf( "\nEwidencja jest pusta." );

  for( nr = 0; nr < lb_pojazdow; nr++ )
  {

    printf( "\nDane pojazdu nr: %d\n", nr + 1 );
    pokaz_info( &pojazdy[ nr ] );

    if( nr < lb_pojazdow - 1 )
      printf( "\n\n[Enter] = Nastepny pojazd >>" );
    else
      printf( "\n\n[Enter] = Zakoncz przeglad" );

   ( void )getchar(); fflush( stdin );
  }
}
```

```
Dane pojazdu nr: 1

      Marka: Mazda
      Model: 626
  Rok prod.: 1999
       Cena: 12000
   Przebieg: 134500
Nr rejestr.: KTA1234

[Enter] = Nastepny pojazd >>_
```

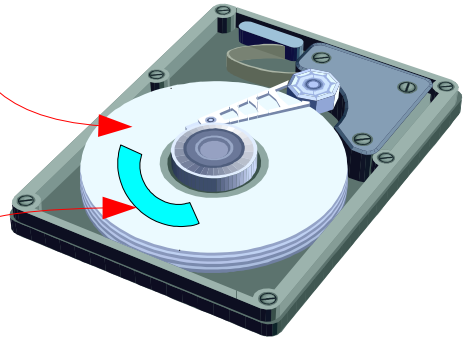
```
Dane pojazdu nr: 3

      Marka: Volvo
      Model: V40
  Rok prod.: 2001
       Cena: 26000
   Przebieg: 124500
Nr rejestr.: KAT1234

[Enter] = Zakoncz przeglad_
```

Tablica struktur — zapis z tablicy do pliku



Zapis liczby pojazdów:
1-en blok o rozmiarze **sizeof(lb_pojazdow) ,**
spod adresu **&lb_pojazdow**

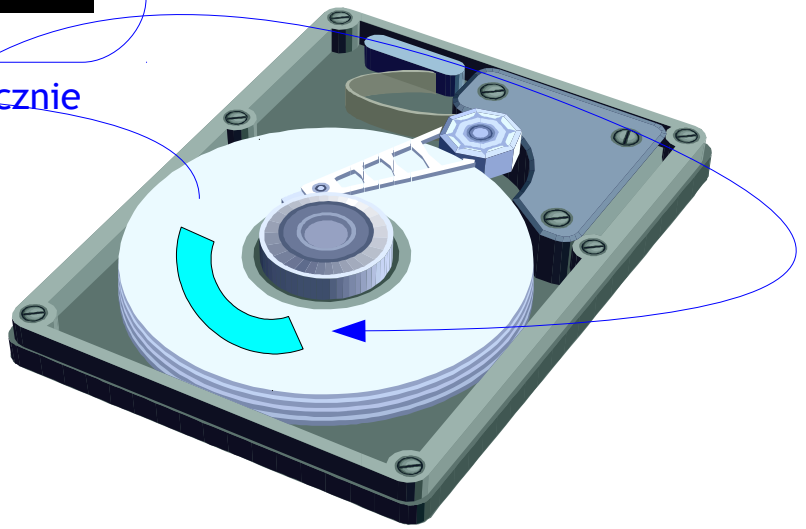
Zapis info o pojazdach:
tyle bloków o rozmiarze **sizeof(pojazd)** ile wynosi **lb_pojazdow,** spod adresu **pojazdy**

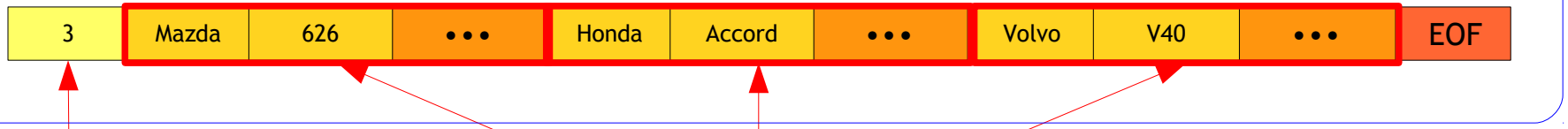


Struktura pliku ewidencji pojazdów

```
03 00 4D 61 7A 64 61 00 00 00 00 00 00 00 00 00 00 00 00 00 ¦ ..Mazda.............
00 00 00 36 32 36 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ¦ ...626..............
00 00 00 00 CF 07 00 00 00 80 3B 46 00 59 03 48 4B 54 41 ¦ ..........;F.Y.HKTA
31 32 33 34 00 00 00 00 00 48 6F 6E 64 61 00 00 00 00 00 ¦ 1234.....Honda.....
00 00 00 00 00 00 00 00 00 00 41 63 63 6F 72 64 00 00 00 ¦ ..........Accord...
00 00 00 00 00 00 00 00 00 00 00 D0 07 00 00 00 C0 DA 46 ¦ ...................F
00 60 EA 47 4B 44 4E 31 32 33 34 00 00 00 00 00 56 6F 6C ¦ .`.GKDN1234.....Vol
76 6F 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 56 34 ¦ vo.................V4
30 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 D1 ¦ 0...................
07 00 00 00 20 CB 46 00 2A F3 47 4B 41 54 31 32 33 34 00 ¦ .... .F.*.GKAT1234.
00 00 00 00                                             ¦ ....
```

Fizycznie



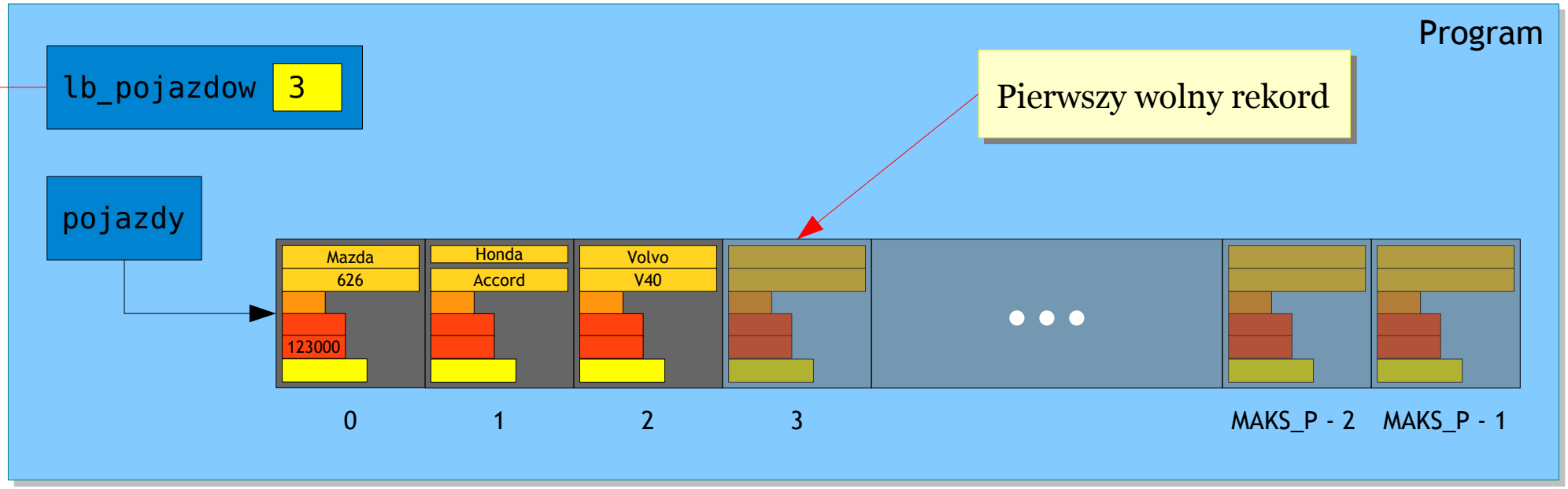


Logicznie

Liczba rekordów

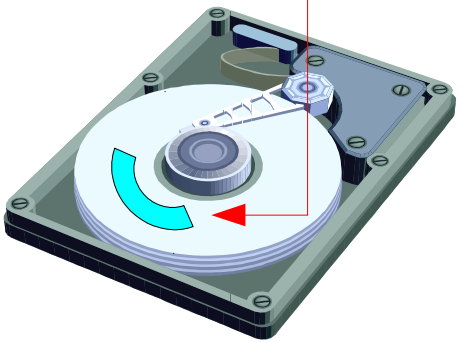
3 rekordy opisu pojazdów

Tablica struktur — zapis liczby pojazdów

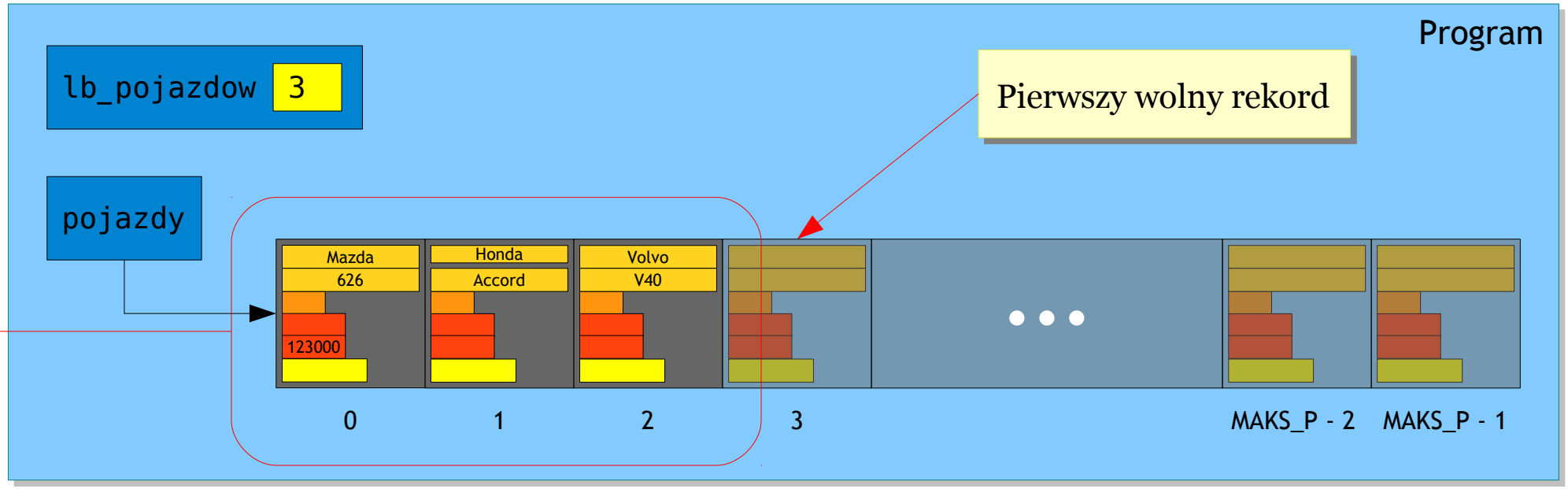



```
fwrite( &lb_pojazdow, sizeof( lb_pojazdow ), 1, plik );
```

```
03 00 4D 61 7A 64 61 00 00 00 00 00 00 00 00 00 00 00 00 00 ¦ ..Mazda.............
00 00 00 36 32 36 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ¦ ...626..............
00 00 00 00 CF 07 00 00 00 80 3B 46 00 59 03 48 4B 54 41 ¦ ..........;F.Y.HKTA
31 32 33 34 00 00 00 00 00 48 6F 6E 64 61 00 00 00 00 00 ¦ 1234.....Honda.....
00 00 00 00 00 00 00 00 00 00 41 63 63 6F 72 64 00 00 00 ¦ ..........Accord...
00 00 00 00 00 00 00 00 00 00 00 D0 07 00 00 00 C0 DA 46 ¦ ...................F
00 60 EA 47 4B 44 4E 31 32 33 34 00 00 00 00 00 56 6F 6C ¦ .`.GKDN1234.....Vol
76 6F 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 56 34 ¦ vo.................V4
30 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 D1 ¦ 0...................
07 00 00 00 20 CB 46 00 2A F3 47 4B 41 54 31 32 33 34 00 ¦ .... .F.*.GKAT1234.
00 00 00 00                                             ¦ ....
```

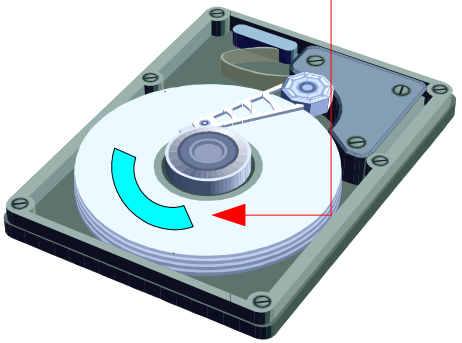


Tablica struktur — zapis rekordów z danymi pojazdów




```
fwrite( pojazdy, sizeof( pojazd ), lb_pojazdow, plik );
```

```
03 00 4D 61 7A 64 61 00 00 00 00 00 00 00 00 00 00 00 00 00 | . Mazda.....
00 00 00 36 32 36 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | ...626.....
00 00 00 00 CF 07 00 00 00 80 3B 46 00 59 03 48 4B 54 41 | ..........;F.Y.HKTA
31 32 33 34 00 00 00 00 00 48 6F 6E 64 61 00 00 00 00 00 | 1234.....Honda.....
00 00 00 00 00 00 00 00 00 00 41 63 63 6F 72 64 00 00 00 | ..........Accord...
00 00 00 00 00 00 00 00 00 00 00 D0 07 00 00 00 C0 DA 46 | ...................F
00 60 EA 47 4B 44 4E 31 32 33 34 00 00 00 00 00 56 6F 6C | .`.GKDN1234.....Vol
76 6F 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 56 34 | vo.................V4
30 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 D1 | 0...................
07 00 00 00 20 CB 46 00 2A F3 47 4B 41 54 31 32 33 34 00 | .... .F.*.GKAT1234.
00 00 00 00                                             | ....
```



Tablica struktur — zapis rekordów z kontrolą poprawności

```
int n;

n = fwrite( &lb_pojazdow, sizeof( lb_pojazdow ), 1, plik );

if( n != 1  )
  printf( "\nBlad zapisu pliku ewidencji pojazdow." );


if( lb_pojazdow > 0 )
{
  n = fwrite( pojazdy, sizeof( pojazd ), lb_pojazdow, plik );

  if( n != lb_pojazdow  )
    printf( "\nBlad w pliku ewidencji pojazdow." );
}
```

Liczba zapisanych bloków jest niezgodna

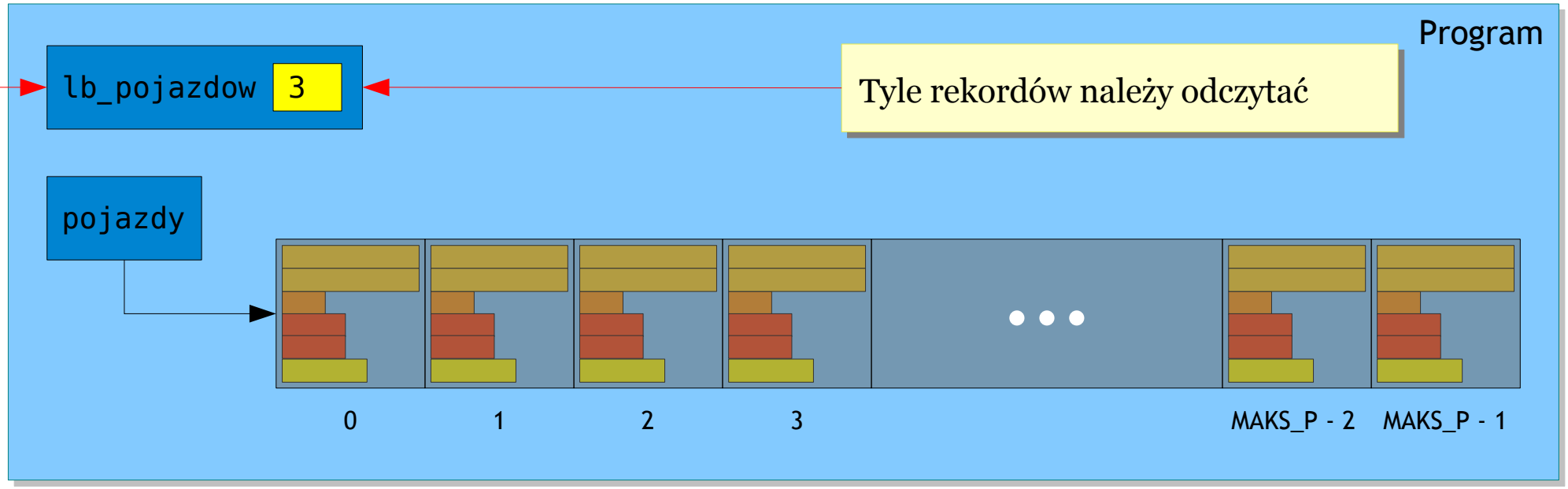
Tablica struktur — zapis z wykorzystaniem funkcji

```
void z_tablicy_do_pliku( void )
{
  FILE * plik;

  printf( "\nZapisywanie ewidencji..." );

  if( ( plik = fopen( nazwa_pliku, "wb" ) ) == NULL )
    printf( " blad aktualizacji pliku." );
  else
  {
    int n;

    n = fwrite( &lb_pojazdow, sizeof( lb_pojazdow ), 1, plik );
    if( n != 1  )
      printf( "\nBlad zapisu pliku ewidencji pojazdow." );

    if( lb_pojazdow > 0 )
    {
      n = fwrite( pojazdy, sizeof( pojazd ), lb_pojazdow, plik );
      if( n != lb_pojazdow  )
        printf( "\nBlad w pliku ewidencji pojazdow." );
    }
    fclose( plik );
  }
}
```

Zapis liczby pojazdów w ewidencji

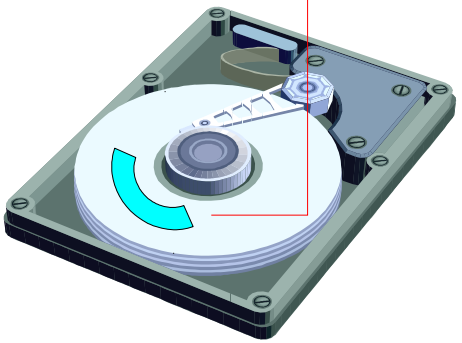
Zapis blokowy rekordów z tablicy

Tablica struktur — odczyt liczby pojazdów z pliku

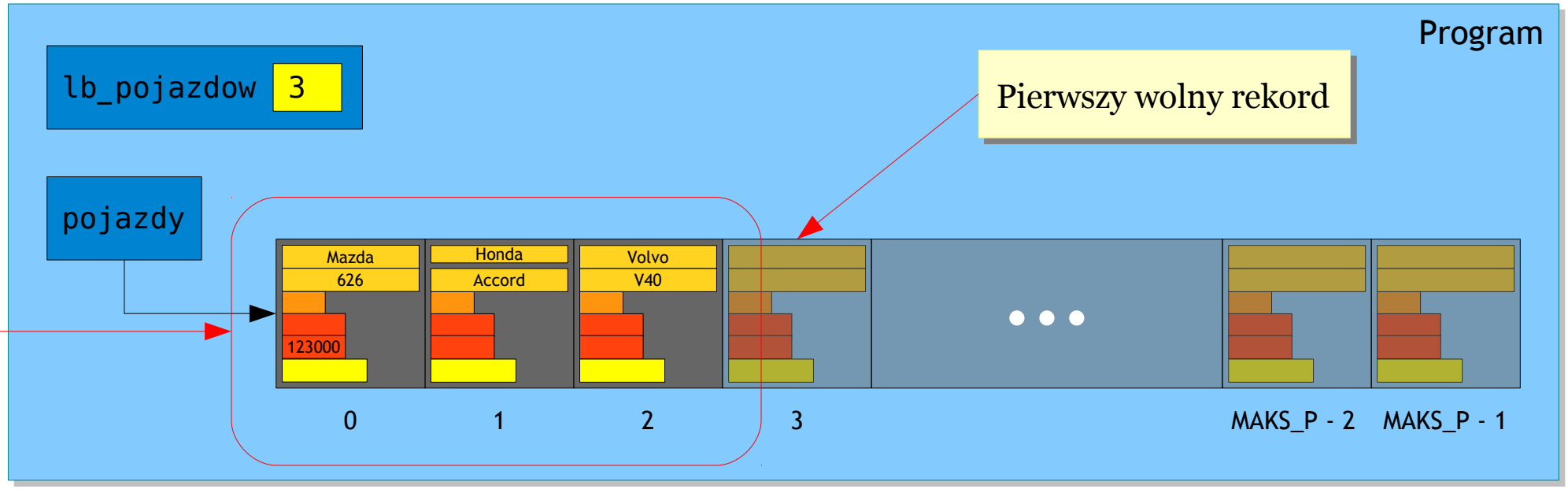



```
fread( &lb_pojazdow, sizeof( lb_pojazdow ), 1, plik );
```

```
03 00 4D 61 7A 64 61 00 00 00 00 00 00 00 00 00 00 00 00 00 ¦ ..Mazda..............
00 00 00 36 32 36 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ¦ ...626..............
00 00 00 00 CF 07 00 00 00 80 3B 46 00 59 03 48 4B 54 41 ¦ ..........;F.Y.HKTA
31 32 33 34 00 00 00 00 00 48 6F 6E 64 61 00 00 00 00 00 ¦ 1234.....Honda.....
00 00 00 00 00 00 00 00 00 00 41 63 63 6F 72 64 00 00 00 ¦ ..........Accord...
00 00 00 00 00 00 00 00 00 00 00 D0 07 00 00 00 C0 DA 46 ¦ ...................F
00 60 EA 47 4B 44 4E 31 32 33 34 00 00 00 00 00 56 6F 6C ¦ .`.GKDN1234.....Vol
76 6F 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 56 34 ¦ vo.................V4
30 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 D1 ¦ 0...................
07 00 00 00 20 CB 46 00 2A F3 47 4B 41 54 31 32 33 34 00 ¦ .... .F.*.GKAT1234.
00 00 00 00                                             ¦ ....
```

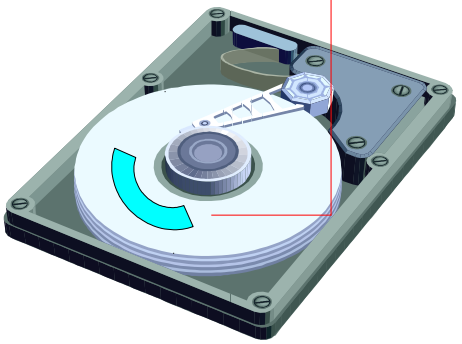


Tablica struktur — odczyt danych z pliku, do rekordów w tablicy




```
fread( pojazdy, sizeof( pojazd ), lb_pojazdow, plik );
```

```
03 00 4D 61 7A 64 61 00 00 00 00 00 00 00 00 00 00 00 00 00 | . Mazda.....
00 00 00 36 32 36 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | ...626.....
00 00 00 00 CF 07 00 00 00 80 3B 46 00 59 03 48 4B 54 41 | ..........;F.Y.HKTA
31 32 33 34 00 00 00 00 00 48 6F 6E 64 61 00 00 00 00 00 | 1234.....Honda.....
00 00 00 00 00 00 00 00 00 00 41 63 63 6F 72 64 00 00 00 | ..........Accord...
00 00 00 00 00 00 00 00 00 00 00 D0 07 00 00 00 C0 DA 46 | ...................F
00 60 EA 47 4B 44 4E 31 32 33 34 00 00 00 00 00 56 6F 6C | .`.GKDN1234.....Vol
76 6F 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 56 34 | vo.................V4
30 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 D1 | 0...................
07 00 00 00 20 CB 46 00 2A F3 47 4B 41 54 31 32 33 34 00 | .... .F.*.GKAT1234.
00 00 00 00                                             | ....
```



Tablica struktur — odczyt z wykorzystaniem funkcji

```
void z_pliku_do_tablicy( void )
{
  FILE * plik;

  printf( "\nLadowanie ewidencji..." );

  if( ( plik = fopen( nazwa_pliku, "rb" ) ) == NULL )
    printf( "plik ewidencji pojazdow nie istnieje." );
  else
  {
    int n;

    n = fread( &lb_pojazdow, sizeof( lb_pojazdow ), 1, plik );
    if( n != 1  )
      printf( "\nBlad w pliku ewidencji pojazdow." );

    n = fread( pojazdy, sizeof( pojazd ), lb_pojazdow, plik );
    if( n != lb_pojazdow  )
      printf( "\nBlad w pliku ewidencji pojazdow." );

    fclose( plik );
  }
}
```

Odczyt liczby pojazdów w ewidencji

Odczyt blokowy rekordów z pliku