INFORMATYKA

C

WYDZIAŁ ELEKTROTECHNIKI I INFORMATYKI
ELEKTROTECHNIKA
DR INŻ. MICHAŁ ŁANCZONT

IX

Zakres wykładu

1. *Struktury*
2. *Unie*
3. *Wyliczenia*
4. *Zapis i odczyt danych do i z pliku*

Złożone typy danych

- W języku C można zdefiniować własny typ danych
- Tworzony typ opiera się na już dostępnych (`int`, `float`, `char`, `bool`) jak i innych typów deklaratywnych
- Dostępne są trzy typy zmiennych deklaratywnych
  - Struktura
  - Unia
  - Typ wyliczeniowy

Struktura

- Typ strukturalny umożliwia sprzężenie grupy zmiennych pod jednym zadeklarowanym typem
- Strukturę budują **pola** typów podstawowych, tablicowych, typów deklaratywnych (struktura, unia, wyliczeniowy)
- W kodzie programu deklaruje się konstrukcję struktury
- Na podstawie zdefiniowanej struktury można utworzyć zmienne strukturalne
- Dobrym obrazem struktury jest tabela dedykowana do przechowywania różnych typów danych

Struktura

- Deklaracja struktury

```
struct nazwa{
  int zmienna1;
  float zmienna2;
  char zmienna3[30];
};      lub    } zmiennaS0;
```

- Deklaracja zmiennej strukturalnej

```
struct nazwa zmiennaS1;
struct nazwa zmiennaS2[20];
struct nazwa *zmiennaS3;
struct nazwa zmiennaS1, zmiennaS3;
```

Struktura

- Deklaracja zmiennej strukturalnej

```
struct nazwa{
    int zmienna1;
  float zmienna2;
  char zmienna3[30];} zmiennaS1;
```

- Inicjacja zmienne strukturalnej

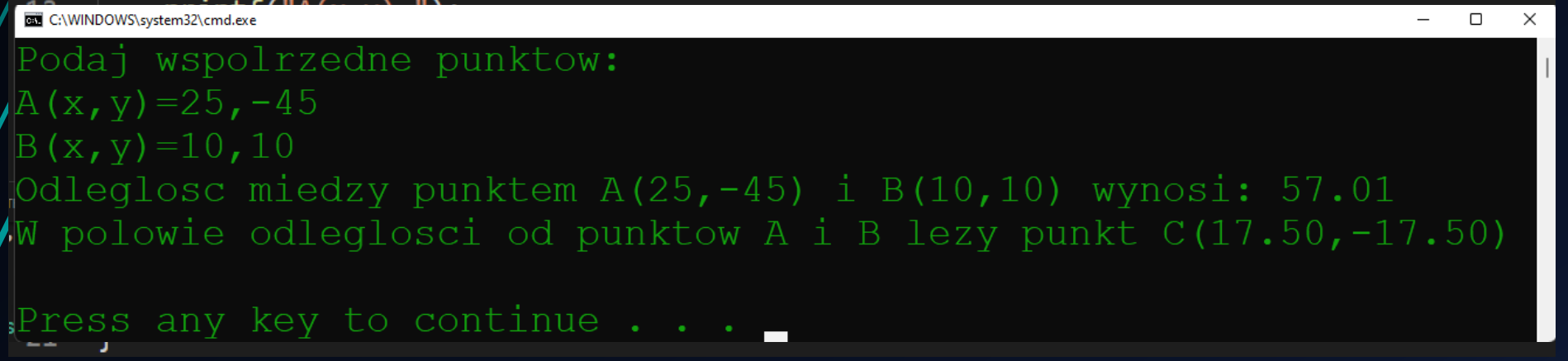
```
struct nazwa zmiennaS1 = {5,3.14,"ALF"};
struct nazwa zmiennaS2 = {10};
```

Struktura

- Zmienna strukturalna może być parametrem wejściowym funkcji
- Funkcja może zwracać wartość typu strukturalnego
- Dla potrzeby funkcji tworzona jest kopia zmiennej strukturalnej
- Przekazując tablicę strukturalną do funkcji przekazywany jest wskaźnik na tą strukturę

Struktura

```
22  float dist(struct punkt A,struct punkt B)
23  {
24      return sqrt(pow(A.y-B.y,2)+pow(A.x-B.x,2));
25  }
26  struct punkt srodek(struct punkt A,struct punkt B)
27  {
28      struct punkt C;
29      C.x=A.x+(B.x-A.x)/2;
30      C.y=A.y+(B.y-A.y)/2;
31      return C;
32  }
```



Struktura

- Stosując dynamiczną alokację pamięci można tworzyć dynamiczną tablicę strukturalną
- Procedury alokacji są analogiczne jak dla tablic klasycznych
- Alokować dynamicznie można także pola w strukturze
- Trzeba pamiętać o zwalnianiu rezerwacji pól przed zwolnieniem pamięci zarezerwowanej dla struktury

Struktura

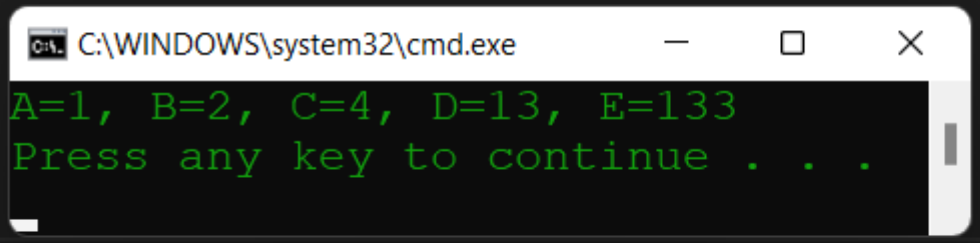
```
22  void druk(struct dane *A,int n)
23  {
24      system("cls");
25      printf("Wyswietlanie danych:\n");
26      for(int i=0; i<n; i++)
27      {
28          printf("Osoba[%i]: ",i+1);
29          printf("%s ",A[i].imie);
30          printf("%s, tel.: ",A[i].nazwisko);
31          printf("%s\n",A[i].telefon);
32      }
33  }
```

```
C:\WINDOWS\system32\cmd.exe
Osoba[1]: Adam Kot, tel.: +48123123123
Osoba[2]: Róża Wójcik, tel.: +48321321321
Osoba[3]: Jan Kowalski, tel.: +48333444555
Osoba[4]: Anna Maria Sykul-Mroczek, tel.: +48123456789

Press any key to continue . . .
```

Struktura

- Dla pól typu całkowitego w strukturze można ograniczyć rozmiar przydzielanej pamięci (w bitach) – nie większy niż wynikający z typu

```c
#include <stdio.h>

struct dane {
    unsigned int    A:1, //od 0 do 1
                    B:2, //od 0 do 3
                    C:3, //od 0 do 7
                    D:4, //od 0 do 15
                    E:8; //od 0 do 255
};
int main()
{
    struct dane liczba;
    liczba.A = 1;
    liczba.B = 2;
    liczba.C = 4;
    liczba.D = 13;
    liczba.E = 133;
    printf("A=%i, B=%i, C=%i, D=%i, E=%i",liczba.A,liczba.B,liczba.C,liczba.D, liczba.E);
}
```



Struktura

- Polami struktury mogą być inne struktury

```c
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
struct punkt {
        float x,y;};
struct prost{
    struct punkt A,B;};
float pole(prost);
struct prost podaj();
int main()
{
    printf("Program liczy pole prostokata okreslonego przez lewy-gorny i prawy-dolny punkt.\n");
    struct prost kw = podaj();
    printf("Pole prostokata wynosi: %.2f\n",pole(kw));
}
float pole(struct prost X){
    return abs(X.B.x-X.A.x)*abs(X.B.y-X.A.y);
}
```

```
C:\WINDOWS\system32\cmd.exe
Program liczy pole prostokata okreslonego przez lewy-gorny i prawy-dolny punkt.
Podaj wspolrzedne lewego-gornego rogu A(x,y): -20,30
Podaj wspolrzedne prawego-dolnego rogu B(x,y): 30,60
Pole prostokata wynosi: 1500.00

Press any key to continue . . .
```

Struktury

- Struktury są wygodnym narzędziem do przekazywania złożonych struktur danych do i z funkcji
- Dzięki zastosowaniu struktur w prosty sposób można uzyskać funkcję zwracającą wiele wartości zgrupowanych w jednej strukturze
- Niektóre funkcje dostępne w języku C korzystają z tego mechanizmu
- Informacje o aktualnym dacie i godzinie są zwracane przez funkcje biblioteki `time.h` w postaci struktury – `tm`

Przykład

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
struct dane{
    int x,y;
    int tab[10];
};
struct dane losuj(struct dane x)
{
    for(int i=0;i<10;i++)
        x.tab[i] = x.x+rand()%(x.y-x.x+1);
    return x;
}
void print(struct dane x)
{
    printf("Zakres losowania (%i,%i)\n",x.x,x.y);
    for(int i=0;i<10;i++)
        printf("%2i: %3i\n",i+1,x.tab[i]);
}
int main()
{
    srand(time(NULL));
    struct dane nowa;
    nowa.x = -100;
    nowa.y = 100;
    for(int i=0;i<10;i++)
        nowa.tab[i] = 1;
    nowa = losuj(nowa);
    print(nowa);
}
```

```
C:\Windows\system32\cmd.exe
Zakres losowania (-100,100)
 1:  30
 2: -59
 3: -68
 4: -86
 5: -54
 6: -100
 7:  88
 8: -13
 9: -39
10: -19

Press any key to continue . . .
```

Struktura tm

Pole	Znaczenie i zakres wartości
tm_sec	Sekundy domyślnie są wyrażone w zakresie [0..59].
tm_min	Minuty. Zakres [0..59]
tm_hour	Godziny. Zakres [0..23]
tm_mday	Dzień miesiąca. Zakres [1..31]
tm_mon	Miesiąc. Zakres [0..11]
tm_year	Obecny rok. Lata zaczynają się liczyć od roku 1900, czyli: wartość 0 --> 1900 rok.
tm_wday	Dzień tygodnia. Zakres [0..6]. Znaczenie poszczególnych wartości: 0 = Niedziela 1 = Poniedziałek 2 = Wtorek 3 = Środa 4 = Czwartek 5 = Piątek 6 = Sobota
tm_yday	Dzień roku. Zakres [0..365].
tm_isdst	Letnie/zimowe przesunięcie czasowe. Jeśli wartość jest większa od 0 to przesunięcie czasowe jest 'aktywne'. Jeśli wartość mniejsza od 0 to informacja jest

Czas

- Odczytanie daty i godziny za pomocą biblioteki `time.h` wymaga utworzenie dwóch zmiennych

```
time_t timeV;
struct tm *odczyt;
```

- Odczytanie aktualnego czasu systemowego realizuje funkcja `time()`

```
time(&timeV);
```

- Przypisanie do struktury zestawu informacji o dacie i czasie

```
data_czas=localtime(&timeV);// czas lokalny
data_czas=gmtime(&timeV);// czas  Greenwich
```

Czas

- Szczegółowe informacje o dacie i czasie można wyświetlić w sposób ustandaryzowany

```
puts(asctime(data_czas));
```

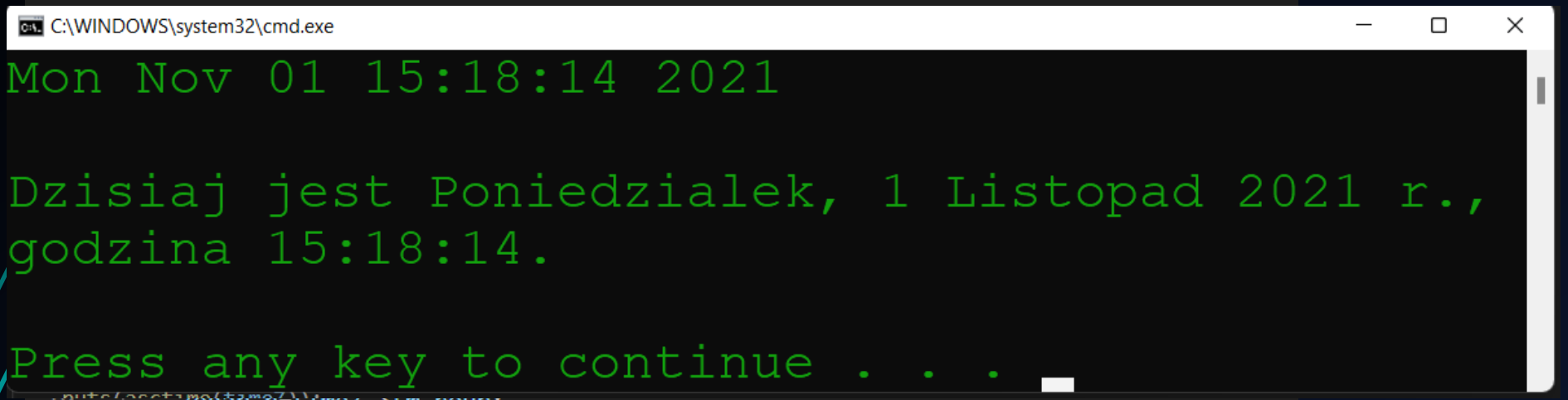
- lub poprzez odwoływanie się do poszczególnych elementów struktury

```
int dzienM = data_czas->tm_mday;
```

- Jeżeli odwołujemy się do struktury poprzez wskaźnik na nią, dostęp do poszczególnych pól realizowany jest przez symbol `->`

Przykład

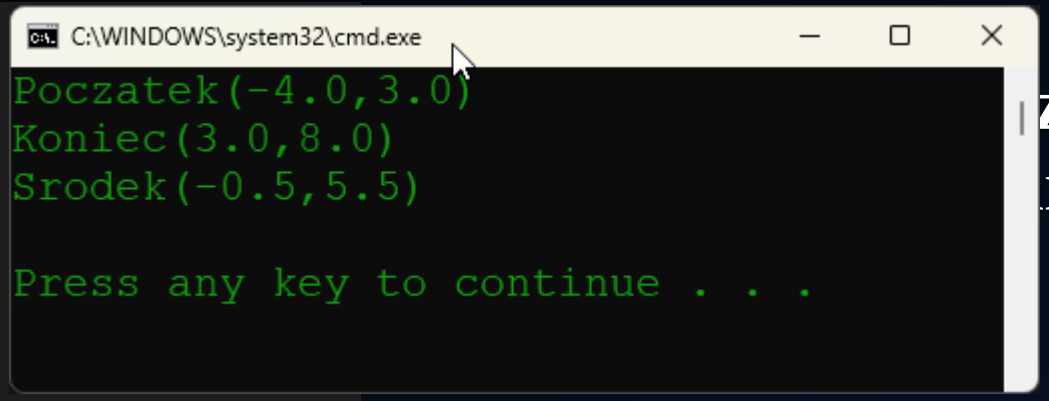
```c
#include <stdio.h>
#include <time.h>
const char dzien[7][13] = {"Niedziela","Poniedzialek","Wtorek","Sroda",
                           "Czwartek","Piatek","Sobota"};
const char mies[12][12] = {"Styczen","Luty","Marzec","Kwiecien","Maj",
                           "Czerwiec","Lipiec","Sierpien","Wrzesien",
                           "Pazdziernik","Listopad","Grudzien"};
```




```c
    zegar.h=timeZ->tm_hour;
    zegar.m=timeZ->tm_min;
    zegar.s=timeZ->tm_sec;
    zegar.dz=timeZ->tm_mday;
    zegar.ty=timeZ->tm_wday;
    zegar.mi=timeZ->tm_mon;
    zegar.rok=timeZ->tm_year+1900;
    puts(asctime(timeZ));
    printf("Dzisiaj jest %s, %d %s %d r., godzina %d:%d:%d.\n",
    dzien[zegar.ty],zegar.dz,mies[zegar.mi], zegar.rok,zegar.h, zegar.m,zegar.s);
}
```

Deklaratywna struktura

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
typedef struct point {
    float x,y;
    char nazwa[20];
} punkt;
punkt midle(punkt,punkt);
int main(){
    punkt A = {-4,3,"Poczatek"};
    punkt B = {3,8,"Koniec"};
    punkt C = midle(A,B);
    printf("%s(%.1f,%.1f)\n",A.nazwa,A.x,A.y);
    printf("%s(%.1f,%.1f)\n",B.nazwa,B.x,B.y);
    printf("%s(%.1f,%.1f)\n",C.nazwa,C.x,C.y);
}
punkt midle(punkt A, punkt B)
{
    punkt C;
    C.x = (A.x+B.x)/2;
    C.y = (A.y+B.y)/2;
    strcpy(C.nazwa,"Srodek");
    return C;
}
```



za
ie

1{

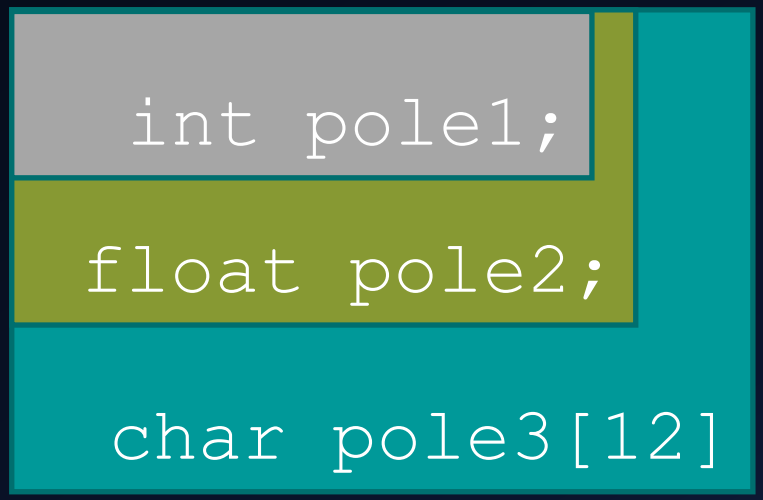
a_b;

_2, int,…);

Unia

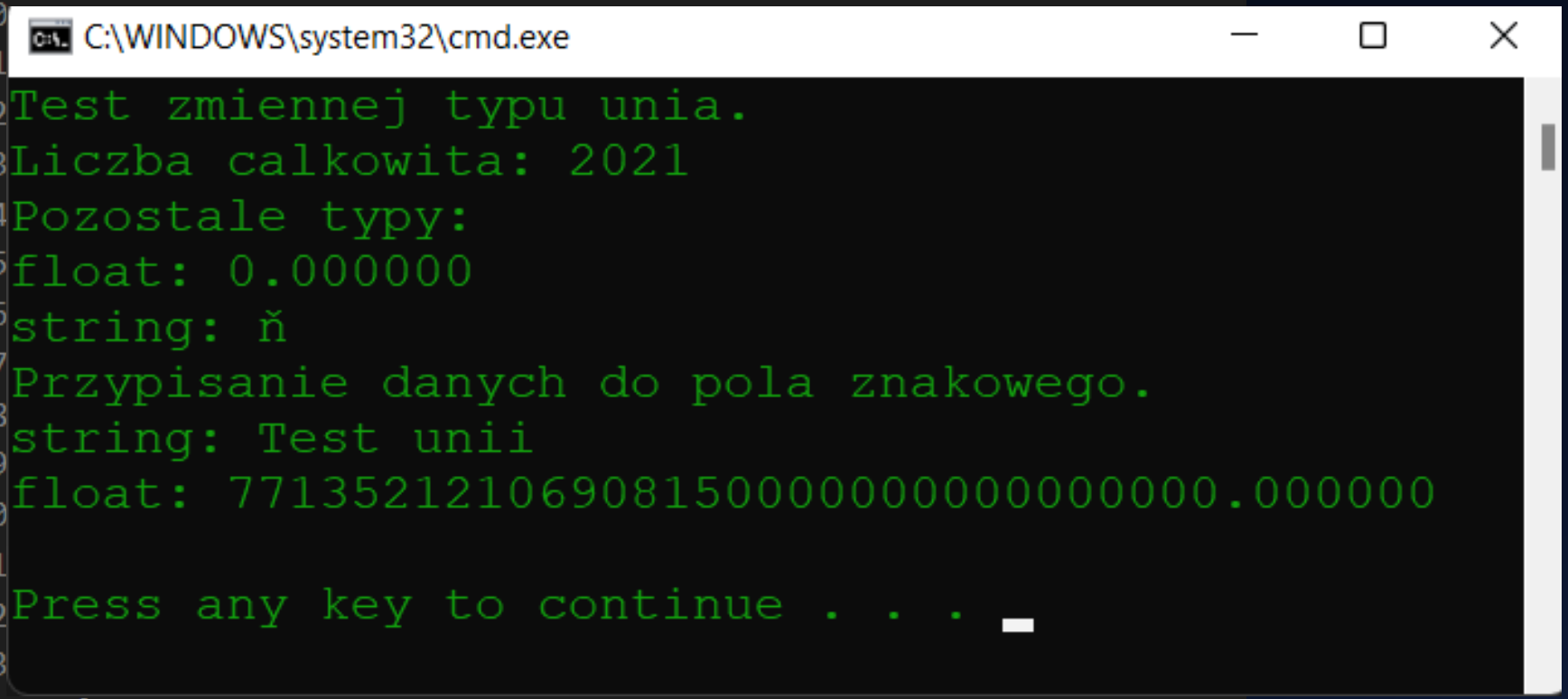
- Złożony typ danych definiowany przez użytkownika dający dostęp w danej chwili do jednego z zdefiniowanych pól
- Dla zadeklarowanej zmiennej typu unia przydzielany jest obszar pamięci konieczny do przechowywania pola o największym rozmiarze

```
union nazwa{
    int pole1;
    float pole2;
    char pole3[12];
};
```



Unia

```c
#include <stdio.h>
#include <string.h>

union dane{
    int i;
    float f;
    char c;
    char s[12];
};
```




```c
}
```

Typ wyliczeniowy `enum`

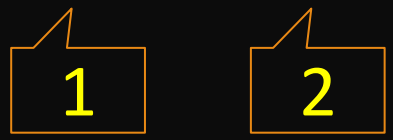
- Typ wyliczeniowy pozwala na utworzenie typu przyjmującego określone wartości
- Każdy elementów jest opisany numerem indeksowym (typ `int`)
- Domyślnie elementy są indeksowane od `0`
- Można do każdego elementy przypisać dowolny indeks (typu `int`)
- Zmienna utworzona na bazie typu `enum` może przyjmować wartości tylko z zadeklarowanej listy
- Definiowane elementy typu rozdzielane są przecinkiem ’`,`’

Typ wyliczeniowy

```c
#include <stdio.h>
#include <stdlib.h>
enum dane
{
    poniedzialek,wtorek,sroda,czwartek,piatek,sobota,niedziela
};
int main()
{
    da
    te
    ij
      
    ij
      
    ij
        printf("wtorek\n");
}
```

Typ wyliczeniowy

```c
#include <stdio.h>
#include <stdlib.h>
enum dane
{
    sobota=1,niedziela=0,poniedzialek,wtorek,sroda,czwartek,piatek
};
int main()
{
    dane test;
    test = sobota;
    if(test==sobota)
        printf("wtorek\n");
    if(test==wtorek)
        printf("wtorek\n");
    if(test==dane(wtorek))
        printf("wtorek\n");
    printf("%i",test);
}
```



Typ wyliczeniowy

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  enum dane
4  {
5      poniedzialek,wtorek,sroda=53,czwartek,piatek,sobota,niedziela,
6  };
7  int main()
8  {
9      dane test;
10     test = sobota;
11     if(test==sobota)
12         printf("wtorek\n");
13     if(test==wtorek)
14         printf("wtorek\n");
15     if(test==dane(wtorek))
16         printf("wtorek\n");
17     printf("%i",test);
18 }
```

0

1

54

Pliki

- Plik jest pewnym zbiorem danych, zapisanym w systemie plików na nośniku danych. Każdy plik ma skończoną długość, a informacja w nim zapisana jest ciągiem zer i jedynek.
- **pliki tekstowe** - poszczególne bajty w pliku można zinterpretować jako dane alfanumeryczne (znaki), zapisane przy pomocy określonego kodowania (np. ASCII).
- **pliki binarne** - poszczególne bajty w pliku mają dowolne wartości, niekoniecznie są interpretowalne jako znaki alfanumeryczne.

Pliki

- Struktura informacji w plikach binarnych jest ściśle określona przez oprogramowanie, które zapisuje tego typu pliki.
- Programista chcący odczytać i właściwie zinterpretować w swoim programie dane z pliku binarnego, powinien znać jego strukturę.

Zapis i odczyt z pliku

- Zapis i odczyt danych z pliku w języku C realizowany jest podobnie do obsługi standardowych portów wejścia i wyjścia (klawiatura, ekran)
- Dostęp do pliku realizowany jest w sposób sekwencyjny
  - Otwarcie dostępu do pliku w określonym trybie
  - Zapis-odczyt danych z pliku
  - Zamkniecie dostępu do pliku
- Powiązanie pomiędzy fizycznym plikiem na dysku a portem wejścia/wyjścia realizowane jest przez zmienną typu `FILE` zdefiniowaną w bibliotece `<stdio.h>`

Dostęp do pliku

- W kodzie programu należy zadeklarować wskaźnik na zmienną typu `FILE`

```
FILE *plik;
```

- Zadeklarowany wskaźnik może być w kodzie wielokrotnie wiązany i odpinany do różnych plików w wybranych trybach pracy
- Dostęp do pliku może być realizowany w jednym z trzech podstawowych trybów trybów:
  - Odczytu – `r`
  - Zapisu – `w`
  - Dopisywania - `a`

Dostęp do pliku

- Wybór typu pliku realizuje się poprzez wybranie odpowiedniego znacznika trybu
  - Tryb tekstowy (domyślny) - `t`
  - Tryb binarny – `b`
- Każdy z podstawowych trybów może być nadany w trybie rozszerzony ,'`+`'
  - Otwarcie pliku do czytania i nadpisywania (aktualizacja) – `r+`
  - Otwarcie pliku do nadpisywania i czytania – `w+`
  - Otwarcie pliku do dopisywania i odczytu (jeśli nie istnieje, to jest tworzony) – `a+`
- Można łączyć literały znaczników

Dostęp do pliku

- Podpięcie do zmiennej FILE fizycznego pliku wraz z określenie trybu dostępu realizuje funkcja fopen()

```
FILE *plik;
plik = fopen(”nazwa.txt”,”tryb”);
```

- Funkcja fopen() testuje w momencie podpięcia jego poprawność i w przypadku błędu zwraca wartość NULL

```
FILE *plik;
if(plik=fopen(”nazwa.txt”,”rt”))
    printf(”Plik do odczytu.\n”);
else
    printf(”Brak plik do odczytu.\n”);
```

Dostęp do pliku

dane.c > ...

```c
#include <stdio.h>
```

zapis.txt

```
liczba[0]=0
liczba[1]=6
liczba[2]=14
liczba[3]=24
liczba[4]=36
liczba[5]=50
liczba[6]=66
liczba[7]=84
liczba[8]=104
liczba[9]=126
```

```c
"wt"))
ba[%i]=%i\n",i,(i+5)*i);
rzyc pliku!\n");
mienna);
lik);
```

Dostęp do pliku

```
PS G:\Mój dysk\DYDAKTYKA\Informatyka I\Wykład\kody_wyk9> & .\"dane.exe"
liczba[0]=0
liczba[1]=6
liczba[2]=14
liczba[3]=24
liczba[4]=36
liczba[5]=50
liczba[6]=66
liczba[7]=84
liczba[8]=104
liczba[9]=126
PS G:\Mój dysk\DYDAKTYKA\Informatyka I\Wykład\kody_wyk9>
```

```
10          else
11              printf("Nie mozna utworzyc pliku!\n");
12          fclose(plik);
13      }
14      }
```

Określanie pozycji w pliku

```
C:\Windows\system32\cmd.exe
Kursor ustawiony na pozycji 8.
stowy testujacy funkcje ftell i fseek.
Jestem na pozycji: 48
Plik ma 48 Bajtow.
ftell i fseek.

Press any key to continue . . .
```

```
ujacy funkcje ftell i fseek.\n" );
```

```
 9
10
11
12
13
14
15
16
17
18          }
19
20          f
21      }
22      retur
23  }
```

kody_wyk9

Nowy | Sortuj | Wyświetl

Ten komputer › Google Drive (G:) › Mój dysk › DYDAKTYKA › Informatyka I › Wykład › kody_wyk9

Szybki dostęp
- Pulpit
- Pobrane
- Dokumenty
- Obrazy
- Google Drive (G
- 1_PROJEKT
- kody_wyk9
- pdf
- Wykład

Nazwa	Data modyfikacji	Typ	Rozmiar
dane	06.11.2021 14:07	C Source File	1 KB
dane	06.11.2021 14:07	Aplikacja	48 KB
dane.txt	04.11.2021 10:27	Dokument tekstowy	1 KB
enum	05.11.2021 19:49	C Source File	1 KB
enum	01.11.2021 16:41	Aplikacja	48 KB
intel	06.11.2021 14:07	Dokument tekstowy	1 KB
plik.out	06.11.2021 14:10	Plik OUT	1 KB
plik	06.11.2021 14:33	Dokument tekstowy	1 KB
see	06.11.2021 14:33	C Source File	1 KB
see	06.11.2021 14:33	Aplikacja	50 KB

Typ: Dokument tekstowy
Rozmiar: 48 B
Data modyfikacji: 06.11.2021 14:32

być
iku
się

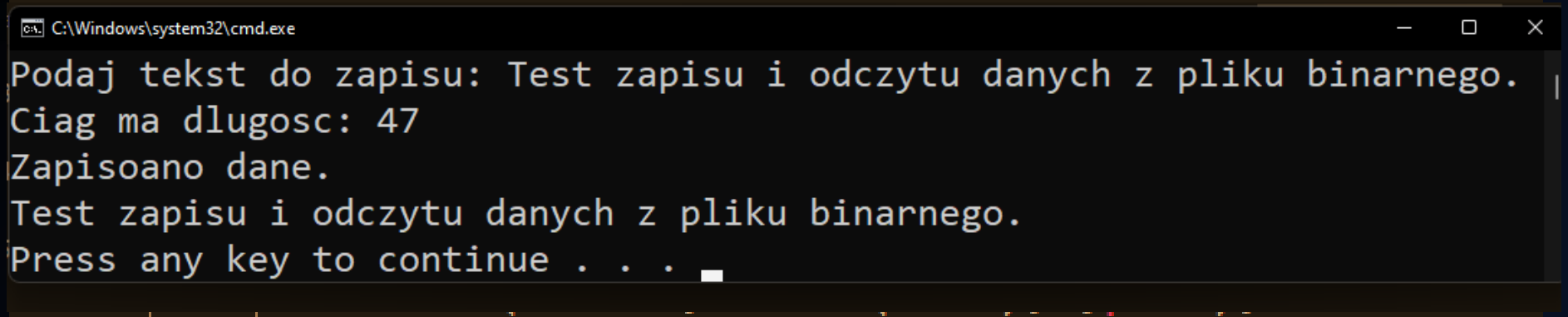
ane
ńca
ycji

) ;

nej

Plik binarny

```
16  void odczyt(char *nazwa)
```




```
22          do{
23              putchar(bufor);
24              fread(&bufor,sizeof(char),1,plik);
25          }while(!feof(plik));
26      }else
27          printf("Brak pliku!\n");
28      fclose(plik);
29  }
15  }
```

Plik binarny