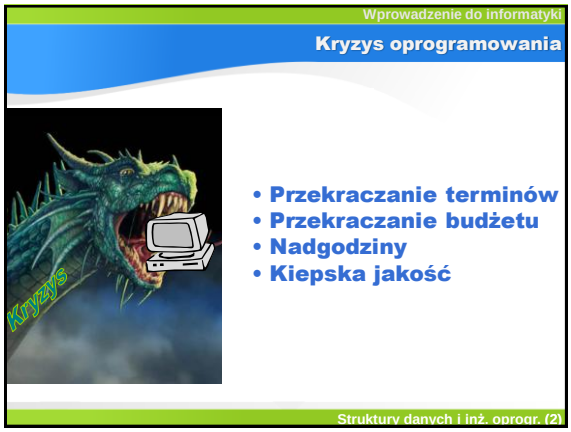
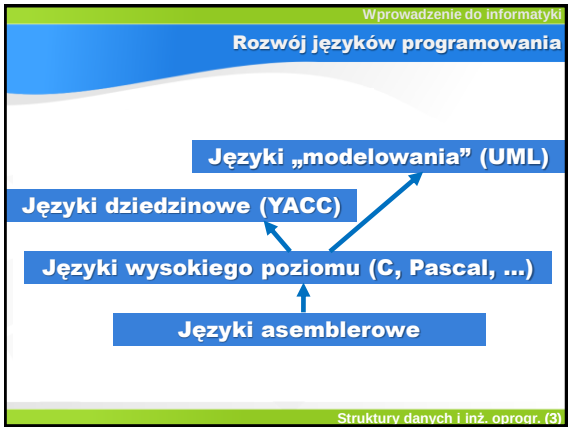
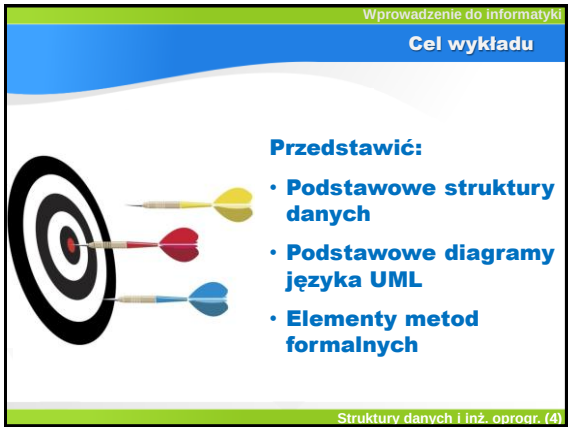
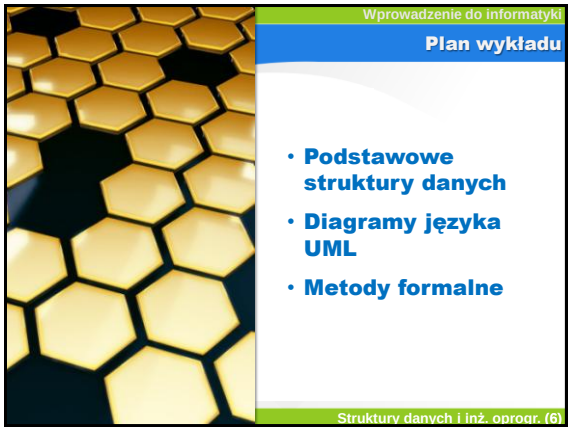

Wprowadzenie do informatyki
Jerzy Nawrocki
Wydział Informatyki
Politechnika Poznańska
jerzy.nawrocki@put.poznan.pl
Struktury danych i inżynieria oprogramowania


Wprowadzenie do informatyki
Kryzys oprogramowania
Kryzys
• Przekraczanie terminów
• Przekraczanie budżetu
• Nadgodziny
• Kiepska jakość
Struktury danych i inż. oprogr. (2)


Wprowadzenie do informatyki
Rozwój języków programowania
Języki „modelowania” (UML)
Języki dziedzinowe (YACC)
Języki wysokiego poziomu (C, Pascal, ...)
Języki asemblerowe
Struktury danych i inż. oprogr. (3)


Wprowadzenie do informatyki
Cel wykładu
Przedstawić:
• Podstawowe struktury danych
• Podstawowe diagramy języka UML
• Elementy metod formalnych
Struktury danych i inż. oprogr. (4)


Wprowadzenie do informatyki
Cykl życia
Wymagania
Projekt
Wykonanie
Struktury danych i inż. oprogr. (5)


Wprowadzenie do informatyki
Plan wykładu
• Podstawowe struktury danych
• Diagramy języka UML
• Metody formalne
Struktury danych i inż. oprogr. (6)

Tablice

Age

Wartość	5	10	15	30
Indeks	0	1	2	3

Age [j+1]

C

```
int Age [4];
...
Age[emp]= Age[emp] + 1;
```

Pascal

```
Age [0..3] of integer;
...
Age[emp]:=  Age[emp] + 1;
```


$$p(x) = \sum_{k=0} a_k x^k = a_0 x^0 + a_1 x^1 + a_2 x^2 + \ldots$$

```
float p(float x, int n, float a[ ]){
    float result, PowerX;
    int k = 0;
    PowerX = 1;
    result = a[0] * PowerX;
    while (k < n){
        k = k+1;
        PowerX = PowerX * x;
        result = result + a[k] * PowerX;
        }
    return result;
    }
```


Rekordy

Student

Born:	1990
Cash:	777,51

Student.Age

C

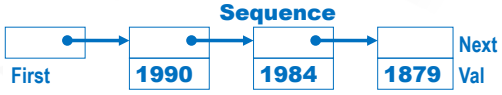
```
struct  Student {int Born
                 float Cash;};
struct Student Einstein;
Einstein.Born= 1879;
```

Pascal

```
Student = record Born: integer;
                 Cash: real;
          end;
var Einstein: Student;
Einstein.Born := 1879;
```


Listy i wskaźniki

Sequence



Pascal

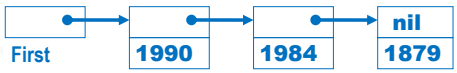
C

```
struct  Elem {int Val
              Elem *Next;};
struct Elem *First;
...
X = (*First).Val;
```

```
Elem = record Val: integer;
               Next: ^Elem;
       end;
var First: ^Elem;
X := First^.Val;
```

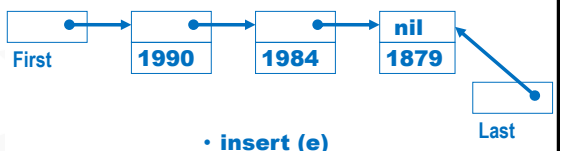

- **Stos: push(e), pop**
- **Kolejka: insert(e), take**
- **Kolejka priorytetowa: insert(e), takeMax**

Stos



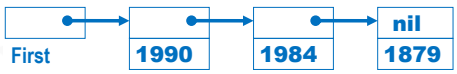
- **push (e)**
- **pop**

Kolejka



- insert (e)
- take

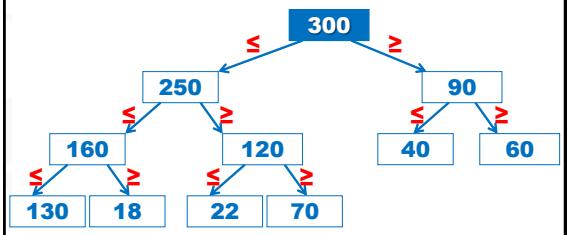
Kolejka priorytetowa



- insert (e)
- takeMax

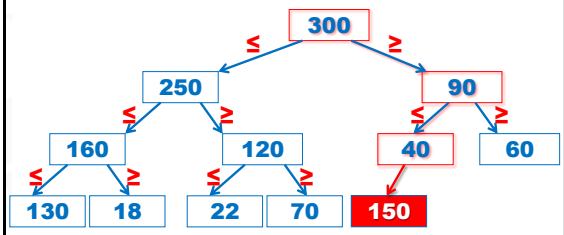
Kolejka priorytetowa jako drzewo binarne

Wartość(Ojciec) ≥ Wartość(Syn)



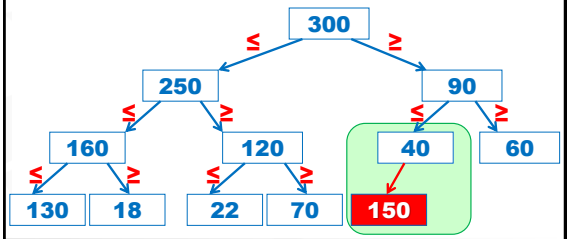
Kolejka priorytetowa jako drzewo binarne

Wartość(Ojciec) ≥ Wartość(Syn)



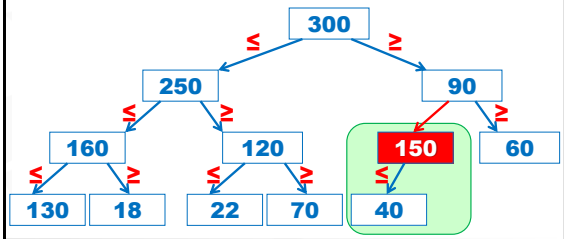
Kolejka priorytetowa jako drzewo binarne

Wartość(Ojciec) ≥ Wartość(Syn)



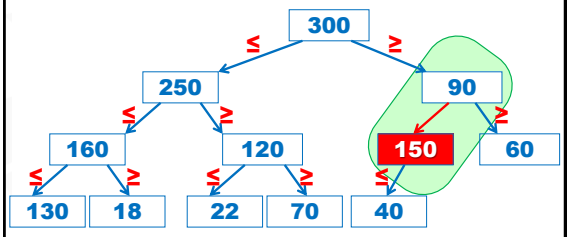
Kolejka priorytetowa jako drzewo binarne

Wartość(Ojciec) ≥ Wartość(Syn)



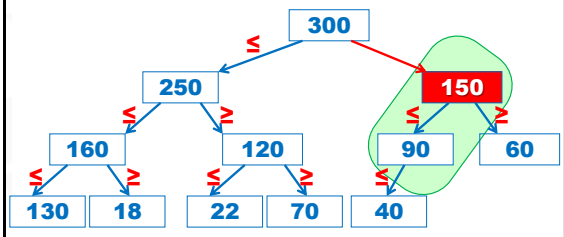
Kolejka priorytetowa jako drzewo binarne

Wartość(Ojciec) ≥ Wartość(Syn)



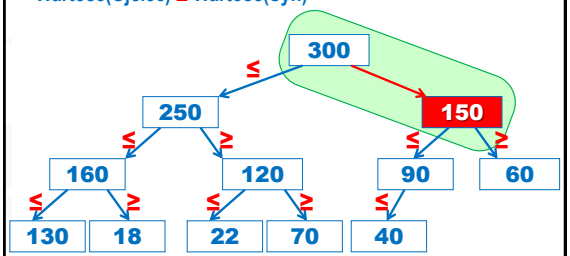
Kolejka priorytetowa jako drzewo binarne

Wartość(Ojciec) ≥ Wartość(Syn)



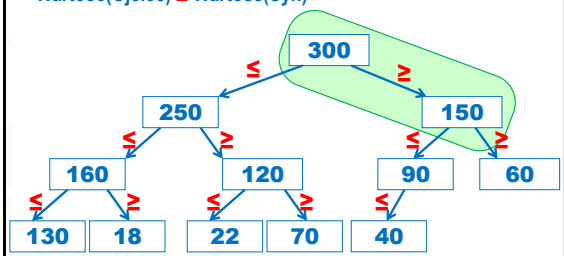
Kolejka priorytetowa jako drzewo binarne

Wartość(Ojciec) ≥ Wartość(Syn)



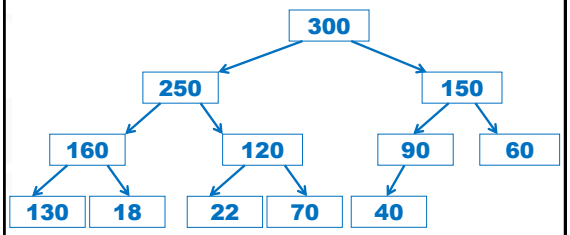
Kolejka priorytetowa jako drzewo binarne

Wartość(Ojciec) ≥ Wartość(Syn)



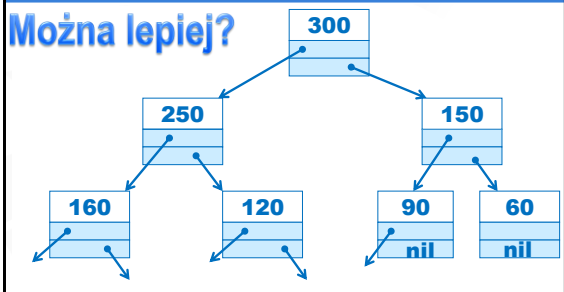
Kolejka priorytetowa jako drzewo binarne

Jak zaimplementować to drzewo?



Implementacja za pomocą wskaźników

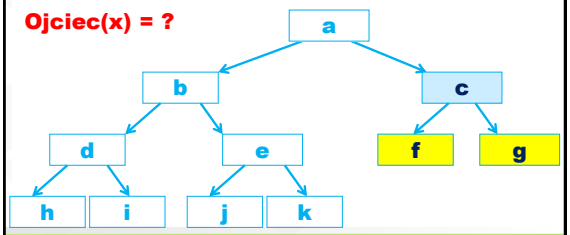
Można lepiej?



Drzewo binarne bez wskaźników

a	b	c	d	e	f	g	h	i	j	k
1	2	3	4	5	6	7	8	9	10	11

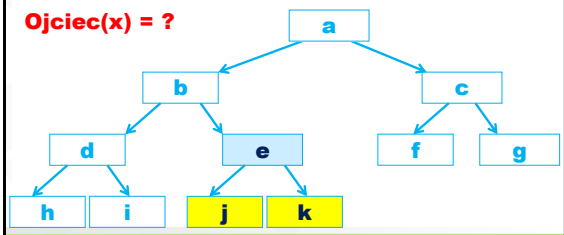
Ojciec(x) = ?



Drzewo binarne bez wskaźników

a	b	c	d	e	f	g	h	i	j	k
1	2	3	4	5	6	7	8	9	10	11

Ojciec(x) = ?



Drzewo binarne bez wskaźników

a	b	c	d	e	f	g	h	i	j	k
1	2	3	4	5	6	7	8	9	10	11

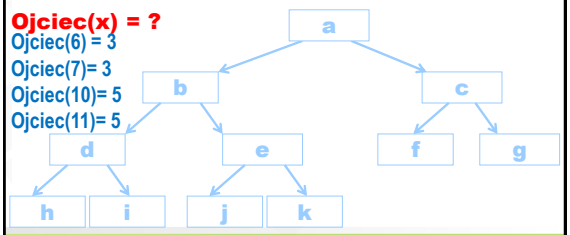
Ojciec(x) = ?

Ojciec(6) = 3

Ojciec(7)= 3

Ojciec(10)= 5

Ojciec(11)= 5



Drzewo binarne bez wskaźników

a	b	c	d	e	f	g	h	i	j	k
1	2	3	4	5	6	7	8	9	10	11

Ojciec(x)= entier(x/2)

Ojciec(6) = 3

Ojciec(7)= 3

Ojciec(10)= 5

Ojciec(11)= 5

C	Pascal
int(x/2)	x div 2

Plan wykładu

- Podstawowe struktury danych
- **Diagramy języka UML**
- Metody formalne

www.uml.org

Diagramy UML

- Diagramy stanów
- Diagramy przypadków użycia
- Diagramy sekwencji
- Diagramy czynności
- Diagramy klas
- . . .

Diagram stanów

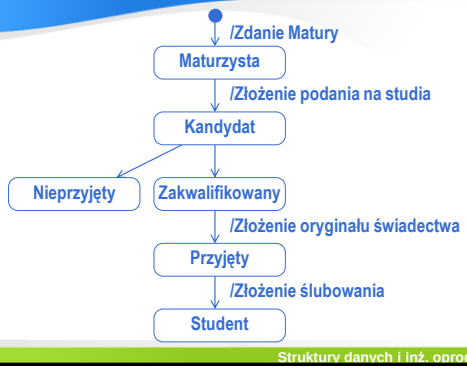


Diagram przypadków użycia

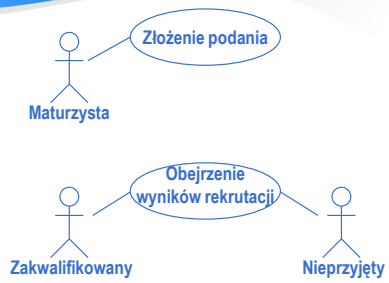


Diagram sekwencji

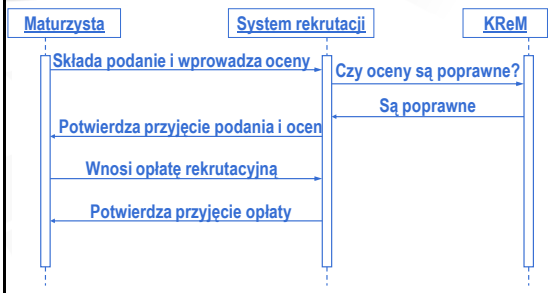
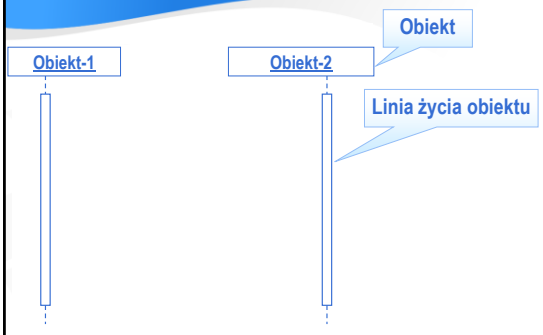
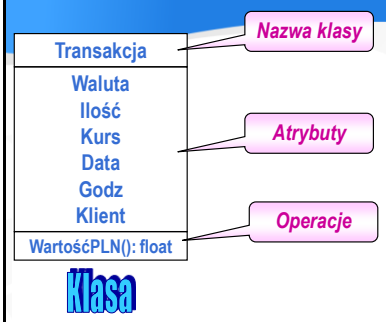


Diagram sekwencji



Klasy i obiekty



Klasy i obiekty

Transakcja
Waluta Ilość Kurs Data Godz Klient
WartośćPLN(): float

Klasa

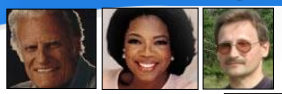
Transakcja
Waluta USD Ilość 1000 Kurs 3.41 Data 2005.10.13 Godz 13:15 Klient Amica
WartośćPLN(): float

Obiekt

Transakcja
Waluta EUR Ilość 1500 Kurs 4.22 Data 2005.10.14 Godz 11:14 Klient Tivoli
WartośćPLN(): float

Obiekt

Jakie atrybuty i operacje?



Obywatel
NIP Podatek Zaliczka . . .
DoZwrotu(): float . . .

Urząd Skarbowy

Obywatel
PESEL Punkty Karany . . .
DodajPkt(int): float WyzerujPkt()

Policja

Dziedziczenie

Pracownik
Konto: string
NoweKonto(string)

Etatowy
Konto: string Pensja: float
NoweKonto(string) NowaPensja(float) Wypłata()

Godzinowy
Konto: string Stawka: float Godz: float
NoweKonto(string) NowaStawka(float) NoweGodz(float) Wypłata()

Dziedziczenie

Pracownik
Konto: string
NoweKonto(string)

Etatowy
Pensja: float
NowaPensja(float) Wypłata()

Godzinowy
Stawka: float Godz: float
NowaStawka(float) NoweGodz(float) Wypłata()

Dziedziczenie

Pracownik
Konto: string
NoweKonto(string)

Etatowy
Pensja: float
NowaPensja(float) Wypłata()

Godzinowy
Stawka: float Godz: float
NowaStawka(float) NoweGodz(float) Wypłata()

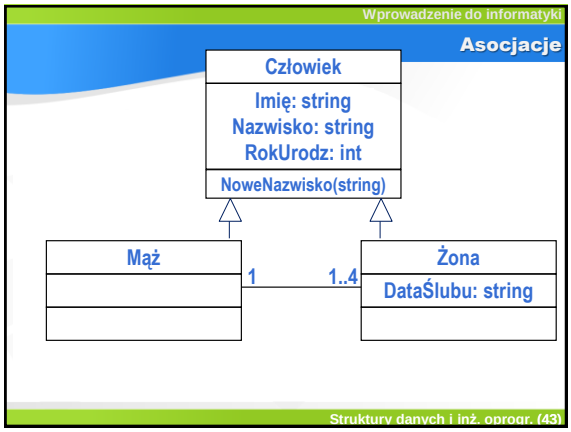
Asocjacje

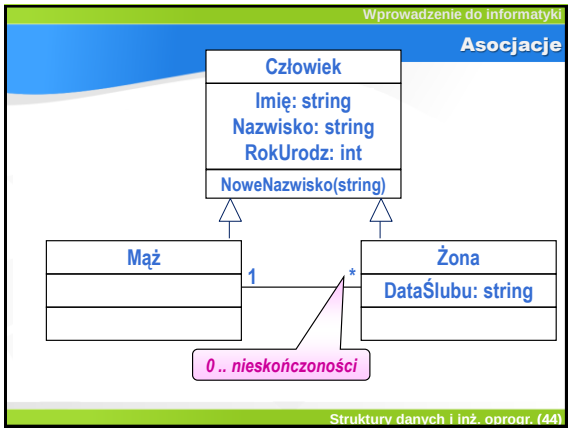
Człowiek
Imię: string Nazwisko: string RokUrodz: int
NoweNazwisko(string)

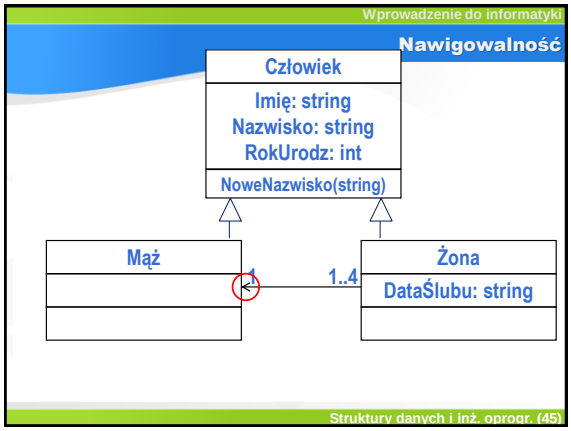
Mąż
DataŚlubu: string

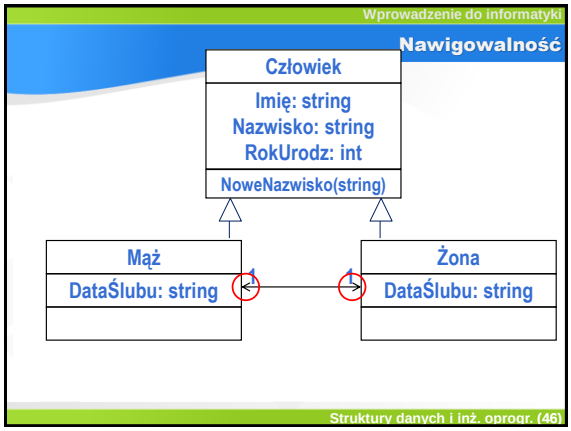
1 — 1

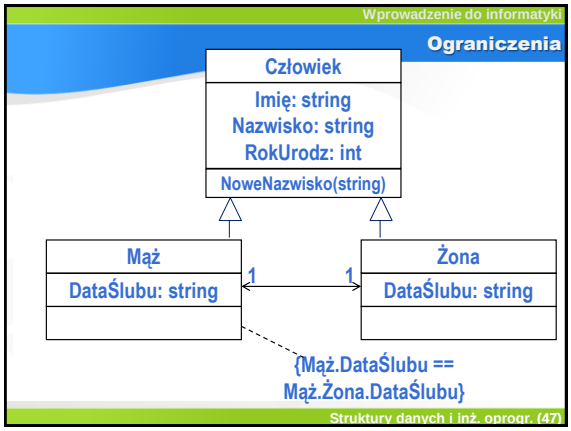
Żona
DataŚlubu: string

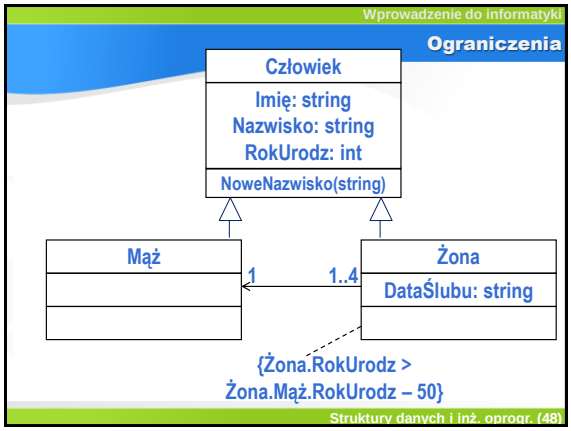

Wprowadzenie do informatyki
Asocjacje
Człowiek
Imię: string
Nazwisko: string
RokUrodz: int
NoweNazwisko(string)
Mąż
Żona
DataŚlubu: string
1
1..4
Struktury danych i inż. oprogr. (43)


Wprowadzenie do informatyki
Asocjacje
Człowiek
Imię: string
Nazwisko: string
RokUrodz: int
NoweNazwisko(string)
Mąż
Żona
DataŚlubu: string
1
*
0 .. nieskończoności
Struktury danych i inż. oprogr. (44)


Wprowadzenie do informatyki
Nawigowalność
Człowiek
Imię: string
Nazwisko: string
RokUrodz: int
NoweNazwisko(string)
Mąż
Żona
DataŚlubu: string
1
1..4
Struktury danych i inż. oprogr. (45)


Wprowadzenie do informatyki
Nawigowalność
Człowiek
Imię: string
Nazwisko: string
RokUrodz: int
NoweNazwisko(string)
Mąż
DataŚlubu: string
Żona
DataŚlubu: string
1
1
Struktury danych i inż. oprogr. (46)


Wprowadzenie do informatyki
Ograniczenia
Człowiek
Imię: string
Nazwisko: string
RokUrodz: int
NoweNazwisko(string)
Mąż
DataŚlubu: string
Żona
DataŚlubu: string
1
1
{Mąż.DataŚlubu ==
Mąż.Żona.DataŚlubu}
Struktury danych i inż. oprogr. (47)


Wprowadzenie do informatyki
Ograniczenia
Człowiek
Imię: string
Nazwisko: string
RokUrodz: int
NoweNazwisko(string)
Mąż
Żona
DataŚlubu: string
1
1..4
{Żona.RokUrodz >
Żona.Mąż.RokUrodz – 50}
Struktury danych i inż. oprogr. (48)

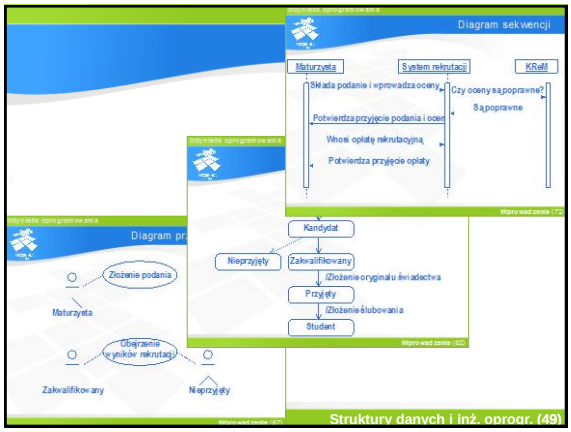
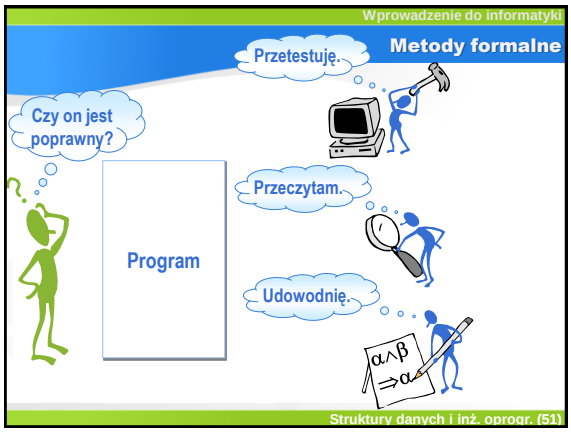
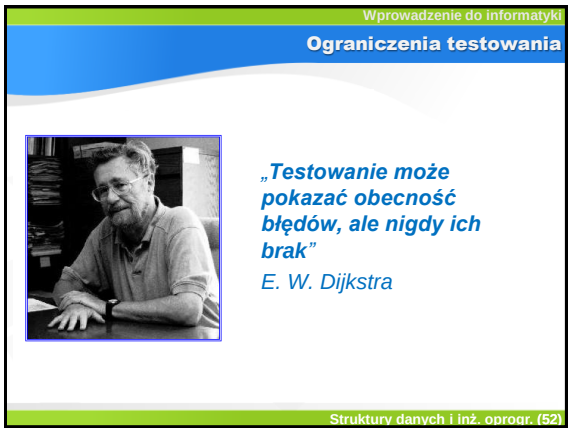
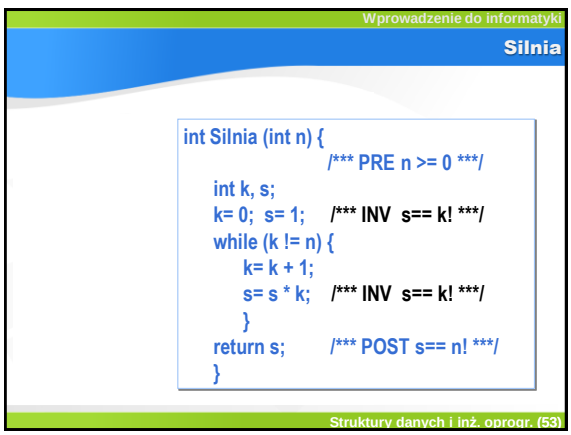


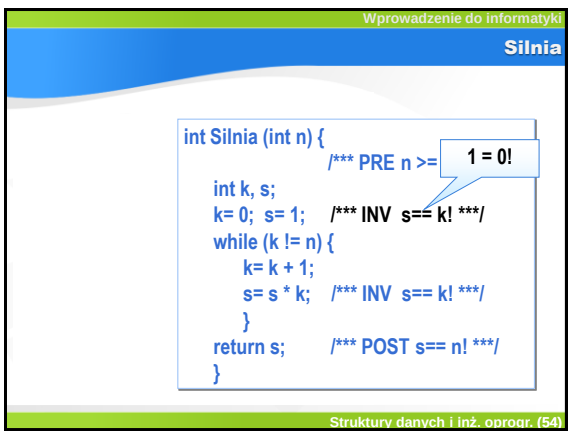
Diagram sekwencji
Maturzysta
System rekrutacji
KRaM
Składa podanie i wprowadza oceny
Czy oceny są poprawne?
Są poprawne
Potwierdza przyjęcie podania i ocen
Wnosi opłatę rekrutacyjną
Potwierdza przyjęcie opłaty
Kandydat
Nieprzyjęty
Zakwalifikowany
Złożenie oryginału świadectwa
Przyjęty
Złożenie ślubowania
Student
Złożenie podania
Maturzysta
Odebranie wyników rekrutacji
Zakwalifikowany
Nieprzyjęty
Struktury danych i inż. oprogr. (49)


Wprowadzenie do informatyki
Plan wykładu
• Podstawowe struktury danych
• Diagramy języka UML
• Metody formalne
Struktury danych i inż. oprogr. (50)


Wprowadzenie do informatyki
Metody formalne
Przetestuję.
Czy on jest poprawny?
Program
Przeczytam.
Udowodnię.
Struktury danych i inż. oprogr. (51)


Wprowadzenie do informatyki
Ograniczenia testowania
„Testowanie może pokazać obecność błędów, ale nigdy ich brak”
E. W. Dijkstra
Struktury danych i inż. oprogr. (52)


Wprowadzenie do informatyki
Silnia
int Silnia (int n) {
/*** PRE n >= 0 ***/
int k, s;
k= 0; s= 1; /*** INV s== k! ***/
while (k != n) {
k= k + 1;
s= s * k; /*** INV s== k! ***/
}
return s; /*** POST s== n! ***/
}
Struktury danych i inż. oprogr. (53)


Wprowadzenie do informatyki
Silnia
int Silnia (int n) {
/*** PRE n >= 0 ***/
1 = 0!
int k, s;
k= 0; s= 1; /*** INV s== k! ***/
while (k != n) {
k= k + 1;
s= s * k; /*** INV s== k! ***/
}
return s; /*** POST s== n! ***/
}
Struktury danych i inż. oprogr. (54)

Silnia

```
int Silnia (int n) {
                    /*** PRE n >= 0 ***/
    int k, s;
    k= 0;  s= 1;    /*** INV  s== k! ***/
    while (k != n) {
        k= k + 1;
        s= s * k;   /*** INV  s== k! ***/
        }
    return s;       /*** POST s== n! ***/
    }
```

s== k!

s== k! ∧ k== k + 1 ⇒ s == (k – 1)!

Silnia

```
int Silnia (int n) {
                    /*** PRE n >= 0 ***/
    int k, s;
    k= 0;  s= 1;    /*** INV  s== k! ***/
    while (k != n) {
        k= k + 1;
        s= s * k;   /*** INV  s== k! ***/
        }
    return s;       /*** POST s== n! ***/
    }
```

s== (k – 1)!

s == (k – 1)! ∧ s == s * k ⇒ s == (k – 1)! * k == k!

Silnia

```
int Silnia (int n) {
                    /*** PRE n >= 0 ***/
    int k, s;
    k= 0;  s= 1;    /*** INV  s== k! ***/
    while (k != n) {
        k= k + 1;
        s= s * k;   /*** INV  s== k! ***/
        }
    return s;       /*** POST s== n! ***/
    }
```

k == n

Dowodzenie poprawności programów

Wolfgang Reif

5 000 LOC

Specyfikacja

7 000 LOC

Program

Specyfikacja aksjomatyczna

LOTOS

```
type ext_nat_numbers is
    sorts nat
    opns 0 → nat
         suc: nat → nat
         _ + _ : nat, nat → nat
eqns forall x,y ofsort nat
    x + 0 = x;
    x + succ(y) = succ(x+y);
```

int zero ()
int succ (int x)
int plus (int x, int y)

$\forall_x$ plus(x, zero())= x
$\forall_{x,y}$ plus(x,succ(y))= succ(plus(x,y))

Nasza intuicja:
plus(2, 3)= 5

Niestandardowa implementacja

```
int zero ()
    { return 1; }
int succ (int x)
    { return 2*x; }
int plus (int x, int y)
    { return x * y; }
```

Implementacja spełnia te warunki

$\forall_x$ plus(x, zero())= x
$\forall_{x,y}$ plus(x,succ(y))= succ(plus(x,y))

... ale plus(2,3)= 6

Nasza intuicja:
plus(2, 3)= 5