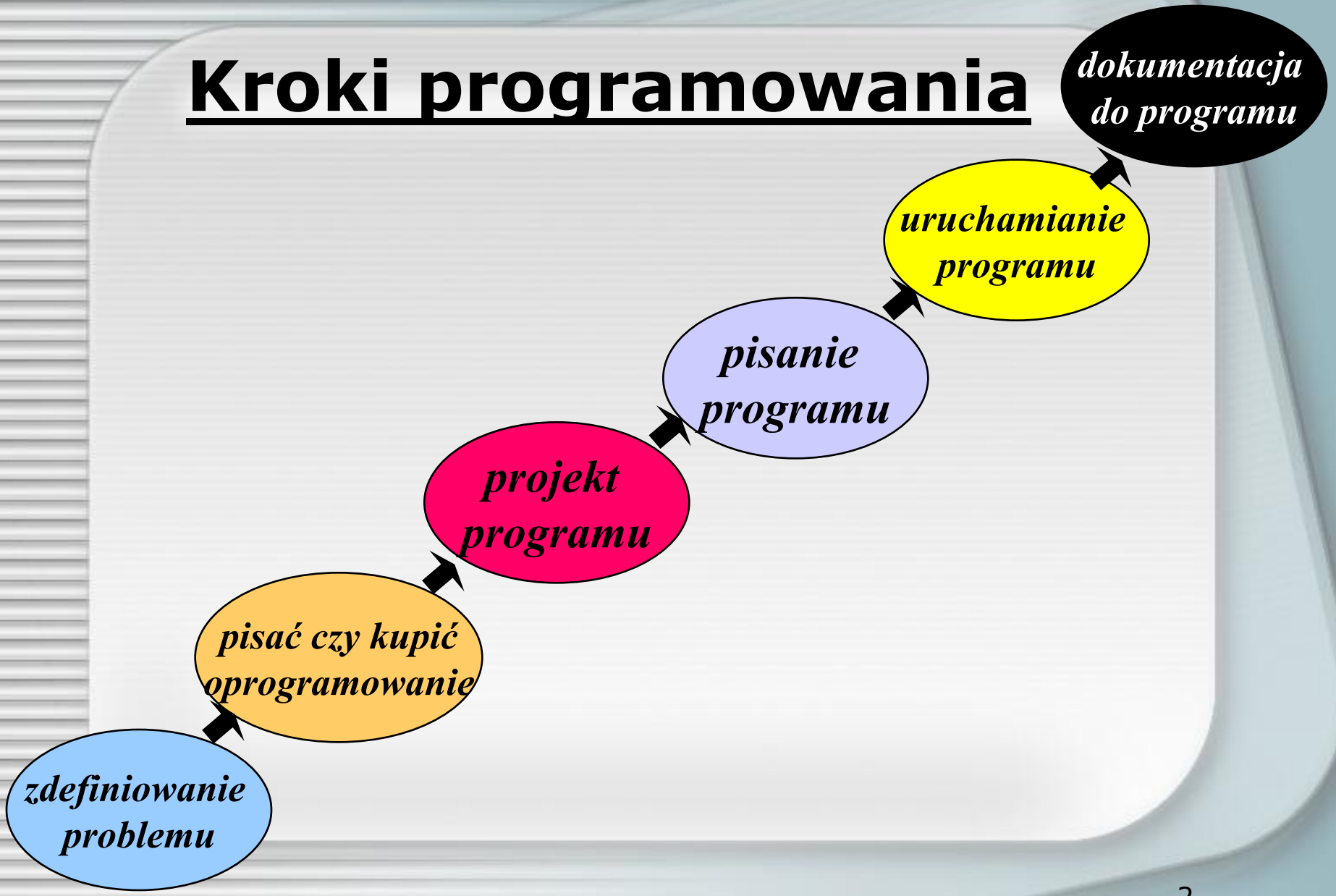
ALGORYTMY

© Instytut Informatyki Stosowanej
Politechnika Łódzka

Kroki programowania



Etapy budowy systemu informatycznego dla przedsiębiorstwa

1 *To, co klient zamówił*

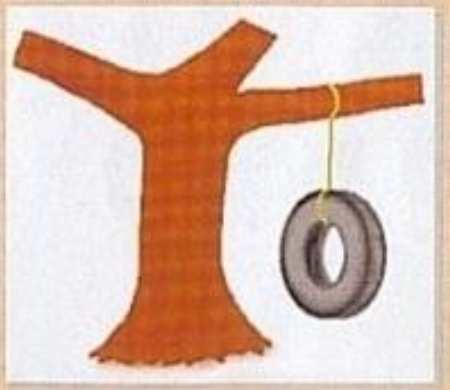
2 *To, co analityk zrozumiał.*

3 *To, co opisywał projekt.*

4 *To, co wykonali programiści.*

5 *Projekt po uruchomieniu i wdrożeniu.*

6 *To, za co klient zapłacił.*



7 *A to, czego klient potrzebował*

8 *Praktyczne zastosowanie projektu.*

Algorytm

Algorytm jest to sposób rozwiązania zagadnienia, podany w formie przepisu określającego skończoną liczbę operacji oraz kolejność w jakiej operacje te powinny być wykonywane.

Algorytm jest podstawowym pojęciem informatyki.

Cechy algorytmu

- **Uniwersalność** – zapewnienie rozwiązania każdego zadania należącego do określonego typu zadań.
- **Jednoznaczność –** prezentacja metody postępowania w postaci skończonej listy prostych i jednoznacznych rozkazów.
- **Zbieżność** – dla każdego dopuszczalnego zbioru danych początkowych liczba operacji prowadzących do poszukiwanego wyniku jest skończona.
- **Powtarzalność** – dla analogicznych danych algorytm, uzyska analogiczne wyniki.

Metody zapisu algorytmów

- **Zapis słowny** w postaci ciągu kroków (języka potocznego)
- **Zapis w notacji matematycznej**
- **Zapis w postaci graficznej** (schemat blokowy)
- **Zapis w języku symbolicznym** (pseudokod)
- **Zapis w języku programowania** (program)
- **Zapis w języku formalnym** (np. UML)
- **Strukturogramy**

Zapis słowny

- Najbardziej rozpowszechniony w życiu codziennym.
- **Polega na logicznym i zrozumiałym przedstawieniu listy kolejnych kroków, które należy wykonać, aby osiągnąć efekt**.
- Zalety: prostota, bez konstrukcji formalnych, szeroki sposób możliwego do użycia słownictwa.
- Wady: mała precyzja opisu, możliwość błędnej interpretacji, mała zwięzłość.

Przykład opisu słownego

Oblicz czas pracy dla pracownika A

1. Wyzeruj godziny pracy i godziny nadliczbowe.
2. Pobierz czas rozpoczęcia pracy i jej zakończenia.
3. Jeśli pracował po godzinie 17:00, oblicz godziny nadliczbowe.
4. Oblicz godziny pracy.
5. Dodaj godziny pracy do całkowitej liczby godzin pracy.
6. Dodaj godziny pracy nadliczbowej do całkowitej liczby godzin pracy nadliczbowej.
7. Jeśli są jeszcze kolejne godziny pracy tego pracownika, powróć do punktu 2 i powtórz kroki (2-7).

Pseudokod

Pseudokod to sposób zapisu algorytmu, który zachowując strukturę charakterystyczną dla kodu zapisanego w języku programowania, rezygnuje ze ścisłych reguł składniowych na rzecz prostoty i czytelności.

Pseudokod nie zawiera szczegółów implementacyjnych. Często też pomija opis działania podprocedur, zaś nietrywialne kroki algorytmu opisywane są z pomocą formuł matematycznych lub zdań w języku naturalnym.

Pseudokod

Zadanie proste:

przypisz średniej **wartość** zero
czytaj x
pisz wynik

Zadanie decyzyjne

jeżeli warunek **to** zadanie
jeżeli warunek1 **to** zadanie1
w przeciwnym przypadku zadanie2

zadanie iteracyjne podczas gdy

podczas gdy warunek **wykonuj** zadanie

Notacja matematyczna

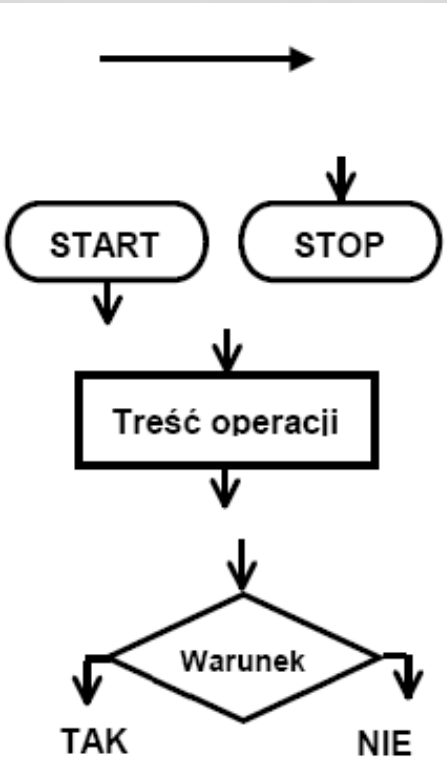
- Wykorzystuje symbole matematyczne
- Oszczędna i precyzyjna
- Użyteczna dla algorytmów przeznaczonych do obliczania wartości wyrażenia

$$\sum_{i=1}^{100} i^2$$

Zapis graficzny

- **Schemat blokowy** (ang. *flowchart, block diagram, block scheme, flow diagram*).
- **Przedstawia algorytm w postaci graficznej z wykorzystaniem bloków operacyjnych oraz wskazaniem przepływu sterowania.**
- Zalety: formalizuje zapis algorytmów.
- Wady: problem z przedstawieniem dużych algorytmów.

Symbole graficznej prezentacji algorytmów



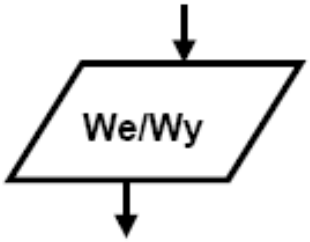
Poszczególne elementy schematu łączy się za pomocą strzałek

Skrzynki START i STOP wskazują początek i koniec każdego algorytmu.

W skrzynce instrukcyjnej umieszcza się polecenia do wykonania (instrukcje) - podstawienie, obliczenie, wprowadzenie wartości.

W skrzynce warunkowej umieszcza się warunek, który decyduje o wyborze dalszej drogi postępowania. Ze skrzynki wychodzą dwa połączenia: TAK (wybierane, gdy warunek jest spełniony), NIE (gdy warunek nie jest spełniony).

Symbole graficznej prezentacji algorytmów

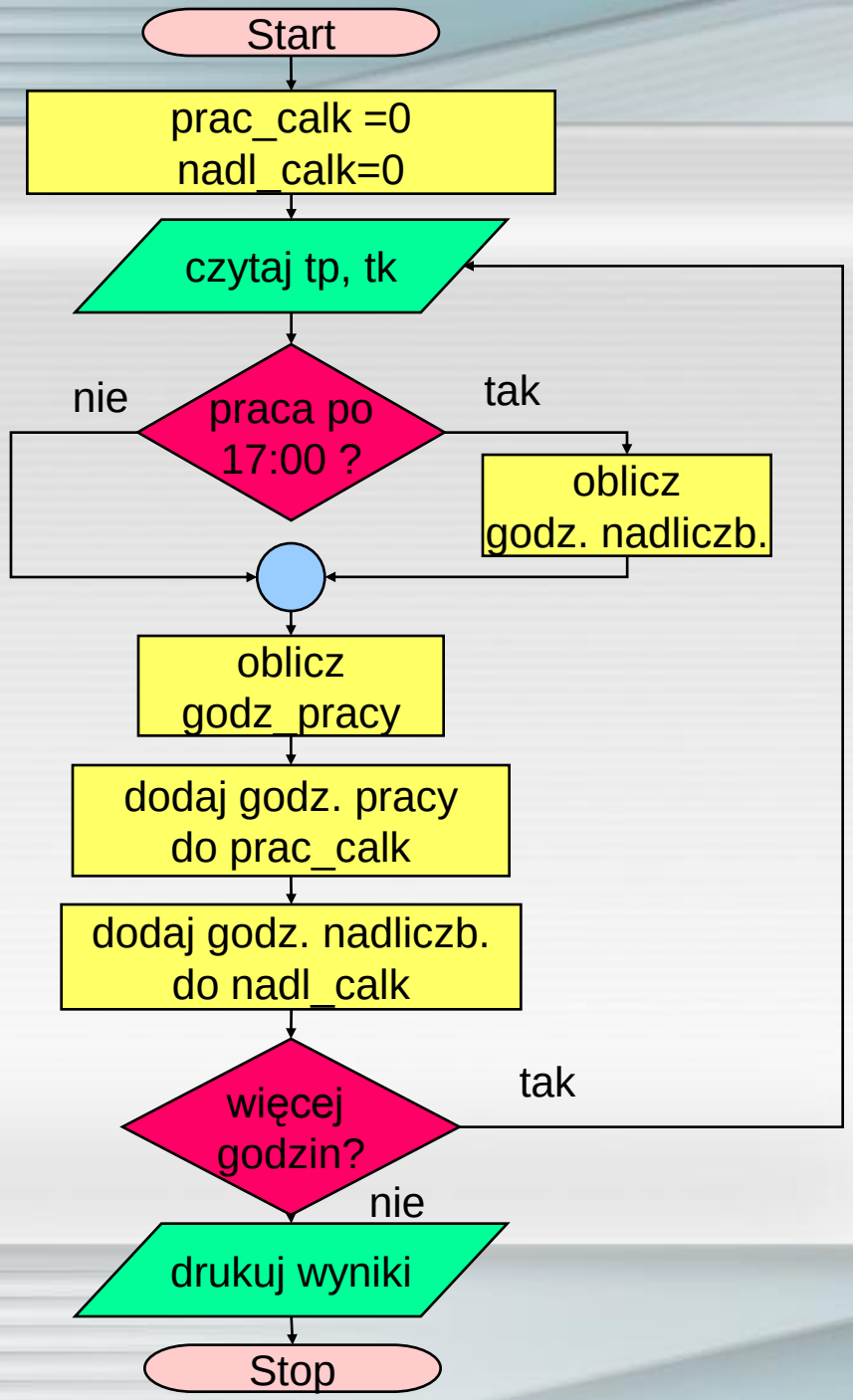


W skrzynce wejścia/wyjścia umieszcza się wprowadzane dane lub wyprowadzane wyniki.

Figura symbolizuje proces, który został już kiedyś zdefiniowany.

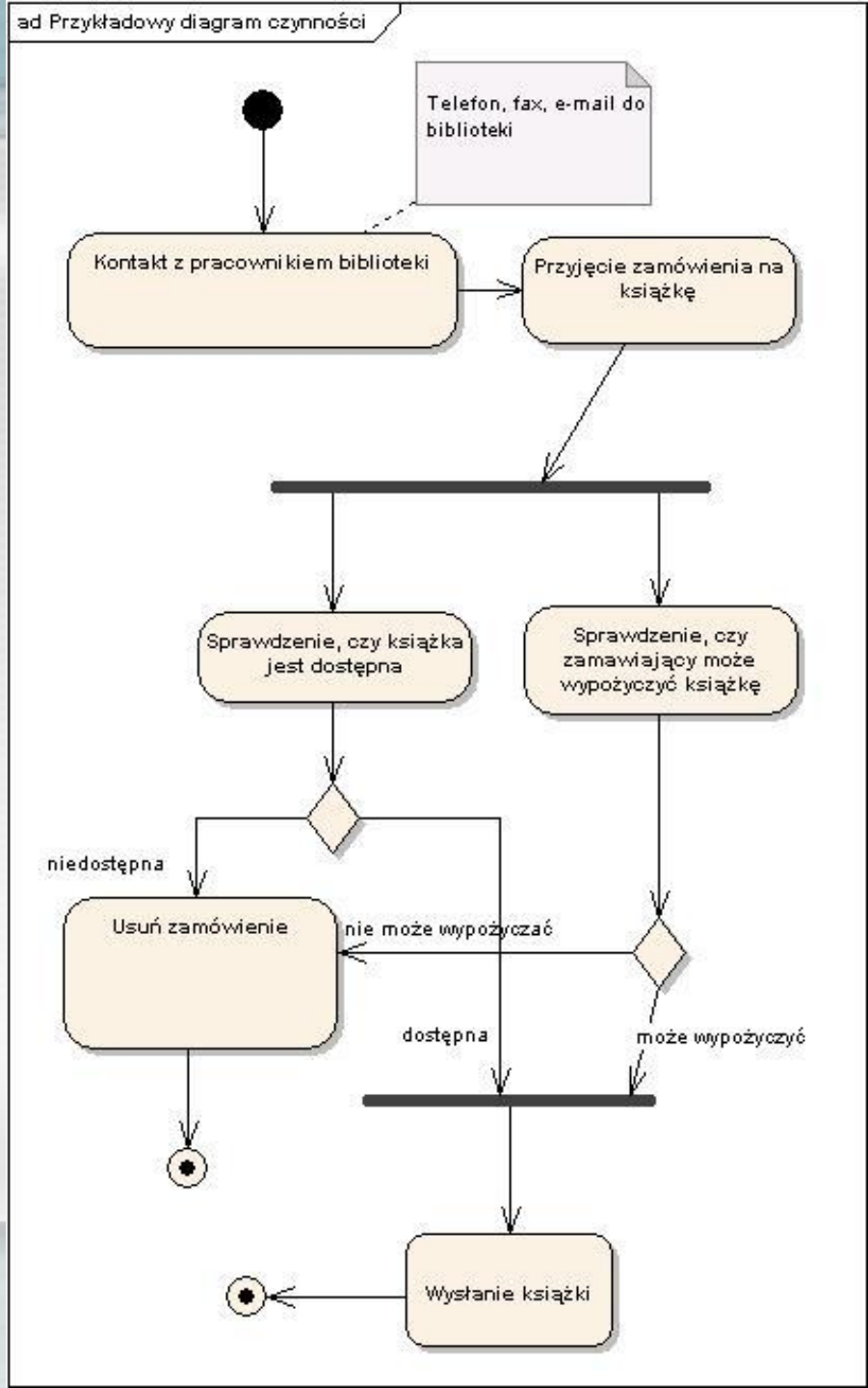
Koło symbolizuje tzw. łącznik stronicowy.

Symbol łącznika między stronicowego.

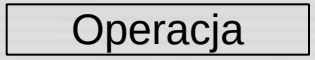

Start
prac_calk =0
nadl_calk=0
czytaj tp, tk
praca po 17:00 ?
nie
tak
oblicz godz. nadliczb.
oblicz godz_pracy
dodaj godz. pracy do prac_calk
dodaj godz. nadliczb. do nadl_calk
więcej godzin?
tak
nie
drukuj wyniki
Stop

Język UML

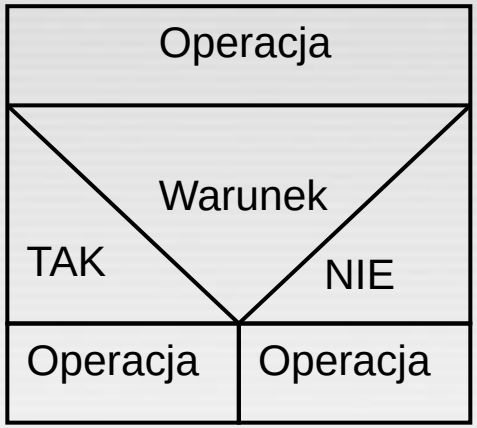
- Diagramy stanów



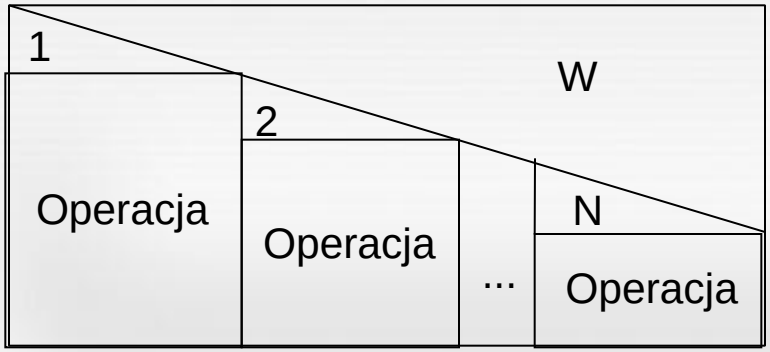
Strukturogramy



Blok operacyjny

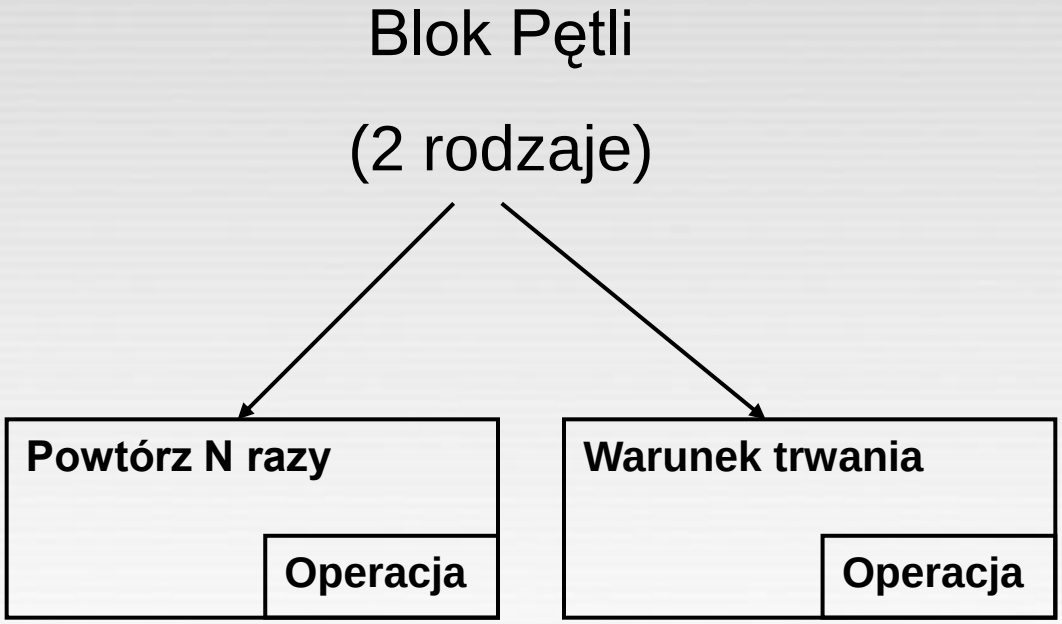


Blok warunkowy



Blok wielokrotnego wyboru

Strukturogramy



Strukturogramy

Implementacja

Zapis (sposób zapisu) algorytmu w konkretnym języku programowania.

Zapis w języku programowania

- Zapis formalny
- Bardzo precyzyjny
- Mało przejrzysty
- Zrozumiały wyłącznie dla programisty

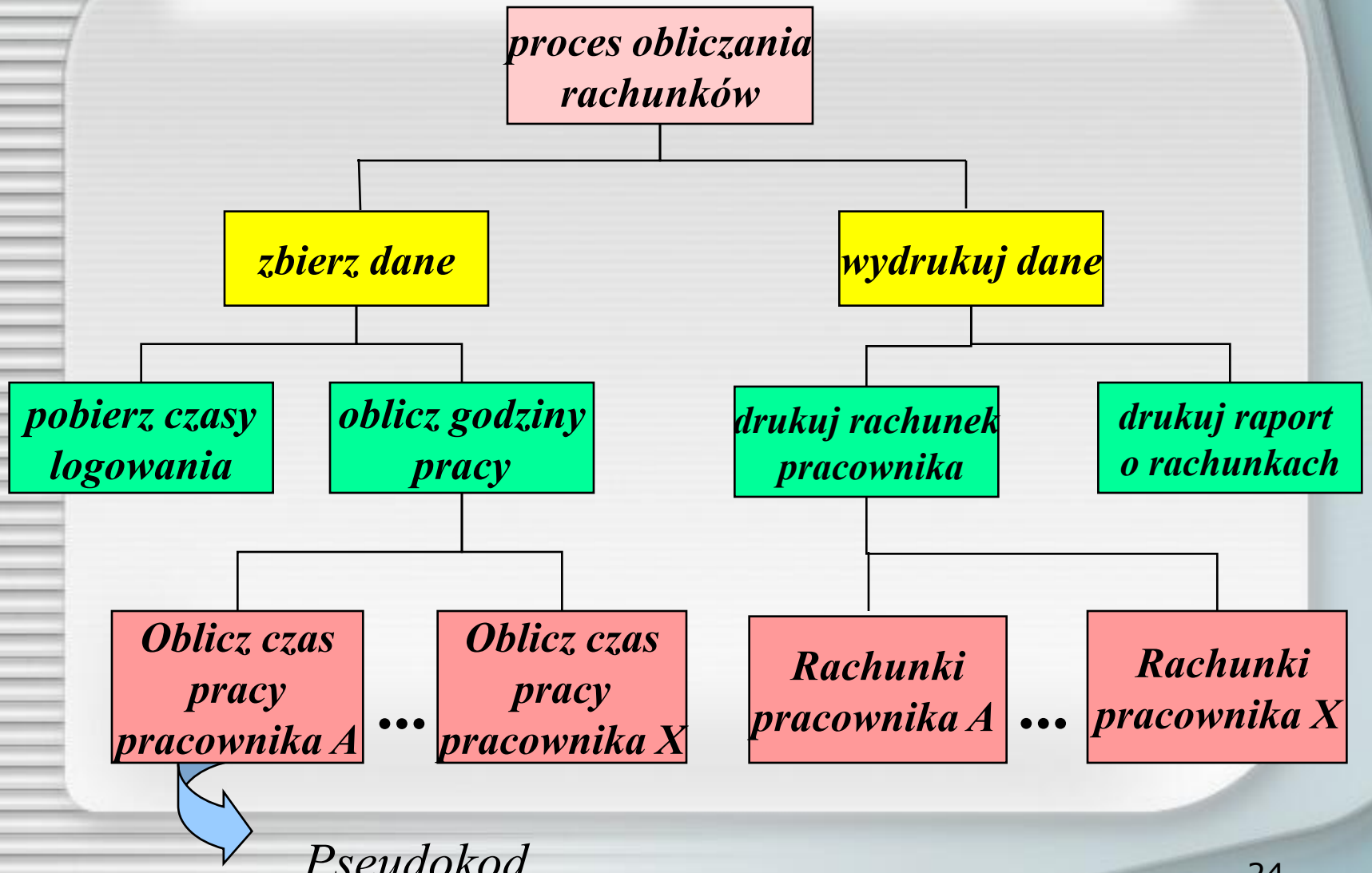
Pascal

```
Program Czas_pracy;
var     plik : text; {zmienna plikowa typu tekst}
        prac_calk, nadl_calk, g_pracy, nadliczbowe : real;
        g_pocz, min_pocz, g_kon, min_kon : integer;
Begin
assign (plik,'time.txt'); {skojarzenie zmiennej plikowej z plikiem na dysku}
reset (plik); {otwarcie pliku do czytania}
prac_calk := 0;      nadl_calk := 0;
while not eof(plik) do
    begin
    readln(plik, g_pocz, min_pocz, g_kon, min_kon); {odczyt z pliku}
    if(g_kon>=17) then nadliczbowe:=(g_kon - 17)+(min_kon/60)
                          else nadliczbowe:= 0;
    g_pracy:=(g_kon-g_pocz)+(min_kon-min_pocz)/60-nadliczbowe;
    prac_calk:=prac_calk+g_pracy;
    nadl_calk:= nadl_calk+nadliczbowe;
    end;
close(plik); {zamknięcie pliku}
writeln('Godziny pracy = ', prac_calk); {wydruk wyników na ekranie}
writeln('Godziny nadliczbowe = ', nadl_calk);
End.
```

C

```c
#include <stdio.h>                    //plik czas_pracy.c
int main(){
 FILE *f;
 char znak;
 int g_pocz, min_pocz, g_kon, min_kon;
 float prac_calk, nadl_calk, g_pracy, nadliczbowe;
 f=fopen("time.txt","r");  //otwarcie pliku do czytania
 if(f!=NULL) {  //jesli plik istnieje
  prac_calk = 0; nadl_calk = 0;
  while(znak!=EOF) {
   fscanf(f,"%d %d %d %d", &g_pocz, &min_pocz, &g_kon, &min_kon); //odczyt
   if(g_kon>=17) nadliczbowe = (g_kon - 17) + (min_kon/60.0);
   else nadliczbowe = 0;
   g_pracy = (g_kon - g_pocz) + (min_kon - min_pocz)/60.0 - nadliczbowe;
   prac_calk = prac_calk + g_pracy;
   nadl_calk = nadl_calk + nadliczbowe;
   znak=fgetc(f);
  }
  fclose(f); //zamknięcie pliku
  printf("Godziny pracy = %f, nadliczbowe = %f\n", prac_calk, nadl_calk);
 }
 else printf("Blad odczytu  z pliku\n");
 return 0;
}
```

Projektowanie programu metodą Top-down



Uruchamianie programu

- Sprawdzanie poprawności działania programu
- Poprawianie błędów
- Testowanie programu metodą wstępującą (ang. *bottom-up*) dla reprezentatywnych danych testowych

Poprawność algorytmu

Dowód poprawności algorytmu jest rozumowaniem matematycznym, prowadzącym do formalnego wykazania, że dany algorytm przy poprawnych danych wejściowych da nam wynik spełniający wymagania.

Złożoność algorytmu

Złożoność obliczeniowa algorytmu jest to miara służąca do porównywania efektywności algorytmów.

Określa ona ilość zasobów komputerowych potrzebnych do wykonania algorytmu.

Podstawowe rozważane zasoby to:

- czas działania (złożoność czasowa);
- ilość zajmowanej pamięci (złożoność pamięciowa).

- Złożoność czasowa - określa jak zwiększa się czas wykonania algorytmu przy zwiększaniu rozmiaru danych dla tego algorytmu
- Złożoność pamięciowa - określa jak zmienia się zajętość pamięci komputera przy wzroście rozmiaru danych algorytmu.

Notacja „wielkie O”

Przykład:

$f(n) = n^2 + 100*n + \log_{10} n + 1000$

można przybliżyć jako:

$f(n) \sim n^2 + 100*n + O(\log_{10} n)$

albo jako:

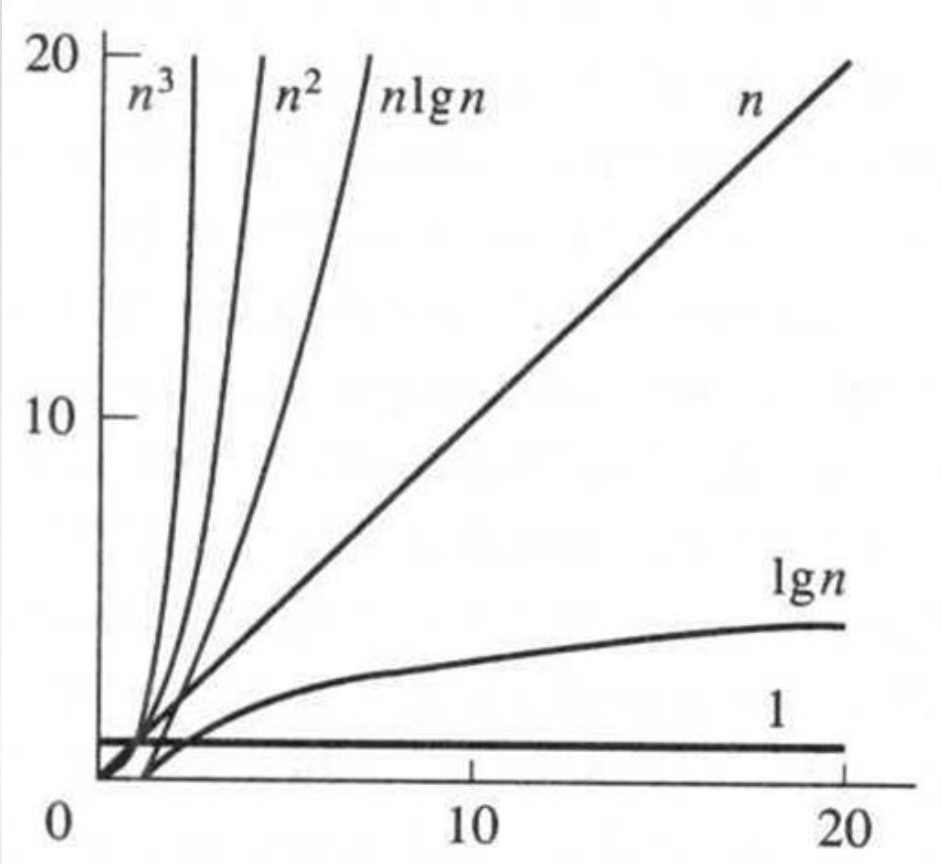
$f(n) \sim O(n^2)$

Klasy złożoności

Klasy złożoności obliczeniowej:

- **1** - stała
- **$\log_2 n$** - logarytmiczna
- **n** - liniowa
- **$n \log_2 n$** liniowo-logarytmiczna (lub quasi-liniowa)
- **n^2** - kwadratowa
- **n^c** - wielomianowa
- **c^n** - wykładnicza

Klasy złożoności



Klasy złożoności

Klasy algorytmów i ich czasy wykonania na komputerze działającym z szybkością 1 instrukcja na μs

klasa	złożoność	liczba operacji i czas wykonania			
	n	**10**		**10^3**	
stały	$O(1)$	1	1ms	1	1 ms
logarytm	$O(\log n)$	3.32	3ms	9.97	10 ms
liniowy	$O(n)$	10	10ms	10^3	1ms
kwadratowy	$O(n^2)$	10^2	100ms	10^6	1s
wykładniczy	$O(2^n)$	1024	1000ms	10^{301}	>> 10^{16} lat

Klasy złożoności

Jedną z najważniejszych funkcji przy ocenianiu efektywności algorytmów jest **funkcja logarytmiczna**. Jeżeli można wykazać że złożoność algorytmu jest rzędu logarytmicznego, algorytm można traktować jako bardzo dobry.