Dr Katarzyna Grzesiak-Kopeć

Inżynieria oprogramowania

10 Programowanie strukturalne

Plan wykładu

- Tworzenie oprogramowania
- Najlepsze praktyki IO
- Inżynieria wymagań
- Technologia obiektowa i język UML
- Techniki IO
- Metodyki zwinne
- Refaktoryzacja
- Mierzenie oprogramowania
- Jakość oprogramowania
- Programowanie strukturalne
- Modelowanie analityczne
- Wprowadzenie do testowania

Paradygmat w nauce

- Wzorzec lub najogólniejszy model, wzorcowy przykład
  - Termin używany w wielu naukach, dotyczy tylko ich podstawowych założeń [Oxford English Dictionary]
- Zbiór pojęć i teorii tworzących podstawy danej nauki [Thomasa Kuhn, Struktura rewolucji naukowych]

Programowanie Proceduralne

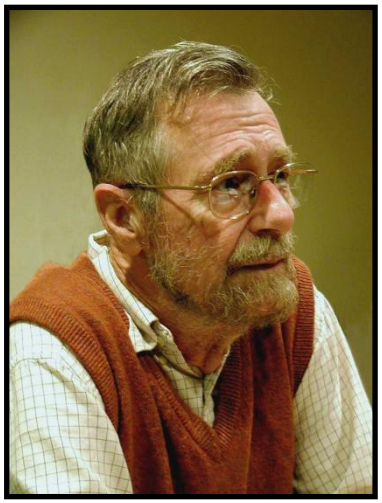
- Paradygmat programowania zalecający dzielenie kodu na procedury, czyli fragmenty wykonujące ściśle określone operacje
- Procedury nie powinny korzystać ze zmiennych globalnych (w miarę możliwości), lecz pobierać i przekazywać wszystkie dane (czy też wskaźniki do nich) jako parametry wywołania

Programowanie Strukturalne

- Paradygmat programowania zalecający hierarchiczne dzielenie kodu na moduły, które komunikują się jedynie poprzez dobrze określone interfejsy
- Rozszerzenie koncepcji programowania proceduralnego
- Pewna poddyscyplina programowania proceduralnego
  - Zaleca stosowanie: pętli i instrukcji warunkowych, oraz
  - Zaleca unikanie: goto, wielokrotnych punktów wejścia i wyjścia z kodu danego podbloku programu

Programowanie Strukturalne

- Edsger Wybe Dijkstra (1930-2002)
  - Holenderski naukowiec, pionier informatyki
  - informatyką zajmował się głównie od strony teoretycznej twierdząc, że tak samo dotyczy ona komputerów, jak astronomia - teleskopów



(photo ©2002 Hamilton Richards)

```
Dijkstra(G,w,s):

dla każdego wierzchołka v w V[G] wykonaj
  d[v] = nieskończone
  poprzednik[v] = niezdefiniowane
d[s] = 0
S = zbiór pusty
Q = V[G]
dopóki Q nie jest zbiorem pustym wykonaj
  u = Wyjmij_Min(Q)
  S = suma zbiorów S i {u}
  dla każdego wierzchołka v, który jest sąsiedni do u wykonaj
    jeżeli d[v] > d[u] + w(u,v) to
      d[v] = d[u] + w(u,v)
      poprzednik[v] = u
```

Pisz tak! (60te XXw.)

PROGRAMOWANIE STRUKTURALNE

- SEKWENCJA
- WARUNEK
- POWTÓRZENIE

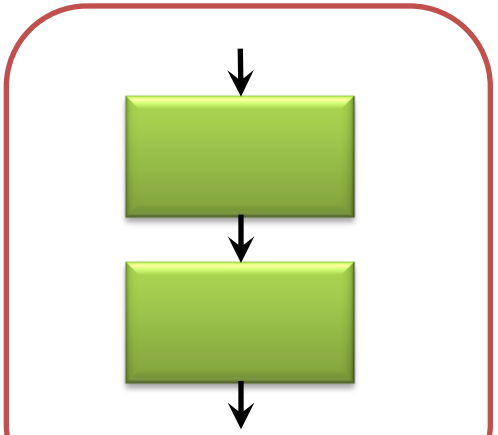
Dlaczego ograniczenia?

- By ograniczyć swobodę projektowania procedur do niewielkiej liczby przewidywalnych operacji
- Programy tak napisane są mniej złożone
  - Łatwiej czytać, testować, pielęgnować
- Każdy program można zapisać przy pomocy wyłącznie tych instrukcji!

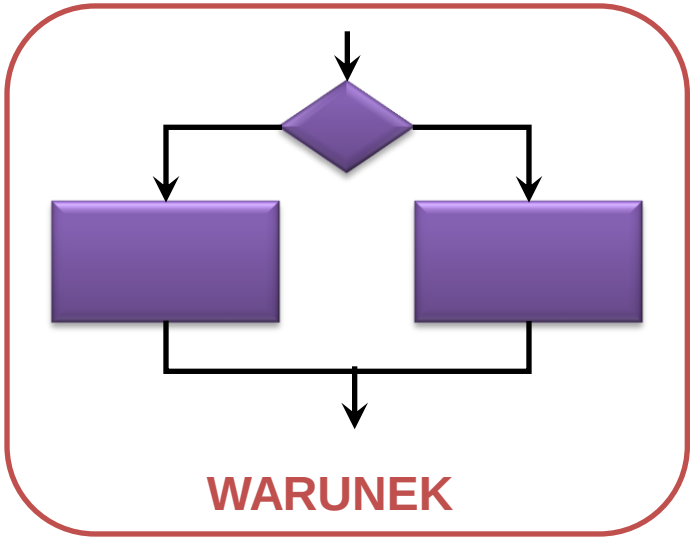
Czasami…

- Ścisłe przestrzeganie reguł prowadzi do niedoskonałych rozwiązań
  - Np. konieczność jednoczesnego opuszczenia kilku zagnieżdżonych pętli
- Co robić?
  - Zmienić algorytm
  - W kontrolowany sposób złamać zasadę konstrukcji strukturalnych i zaprojektować przepływ sterowania z wnętrza pętli na zewnątrz

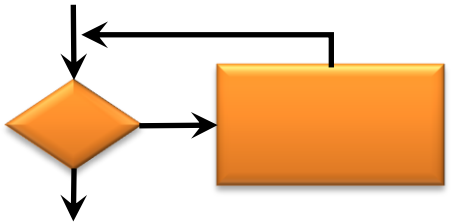
Schemat blokowy



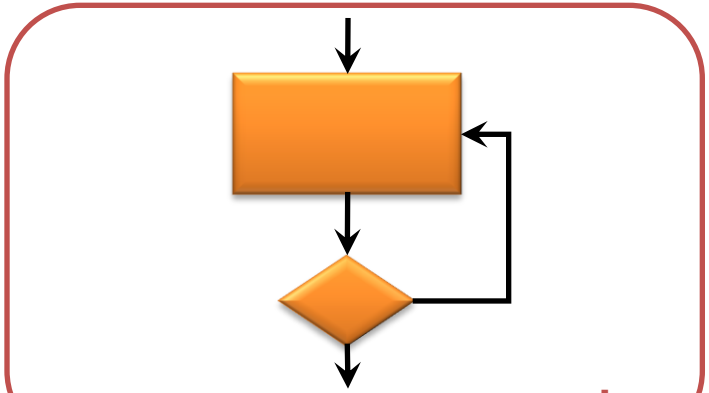
SEKWENCJA



WARUNEK



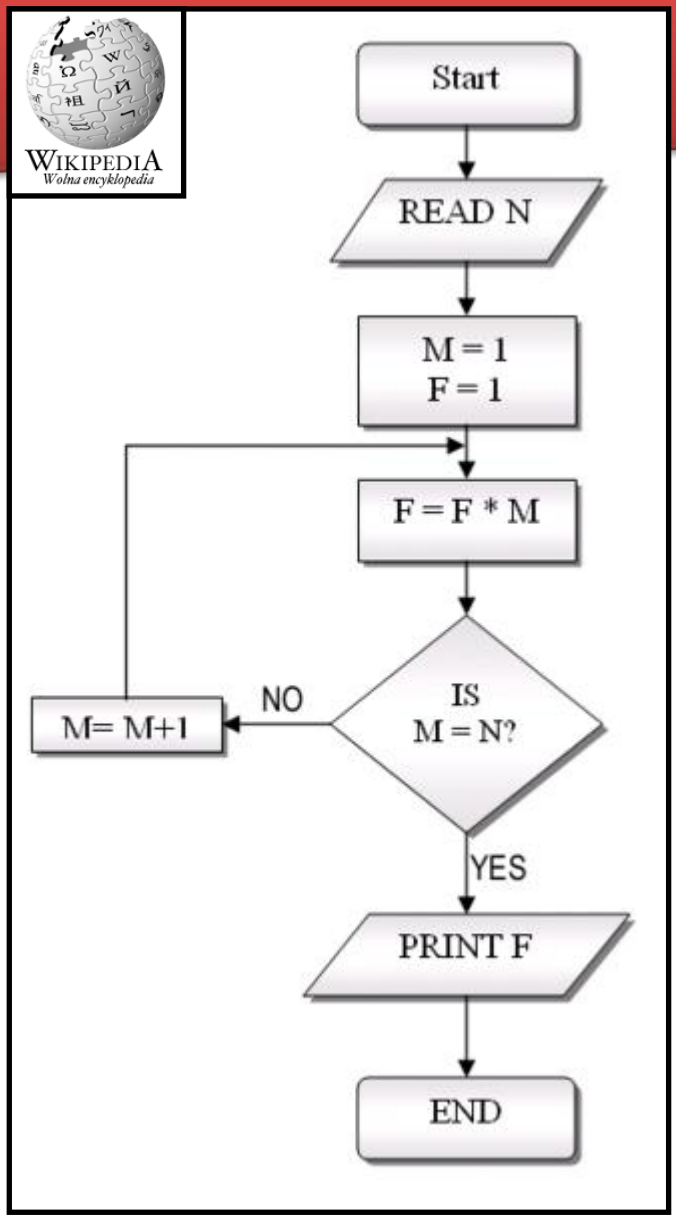
PĘTLA „DOPÓKI”



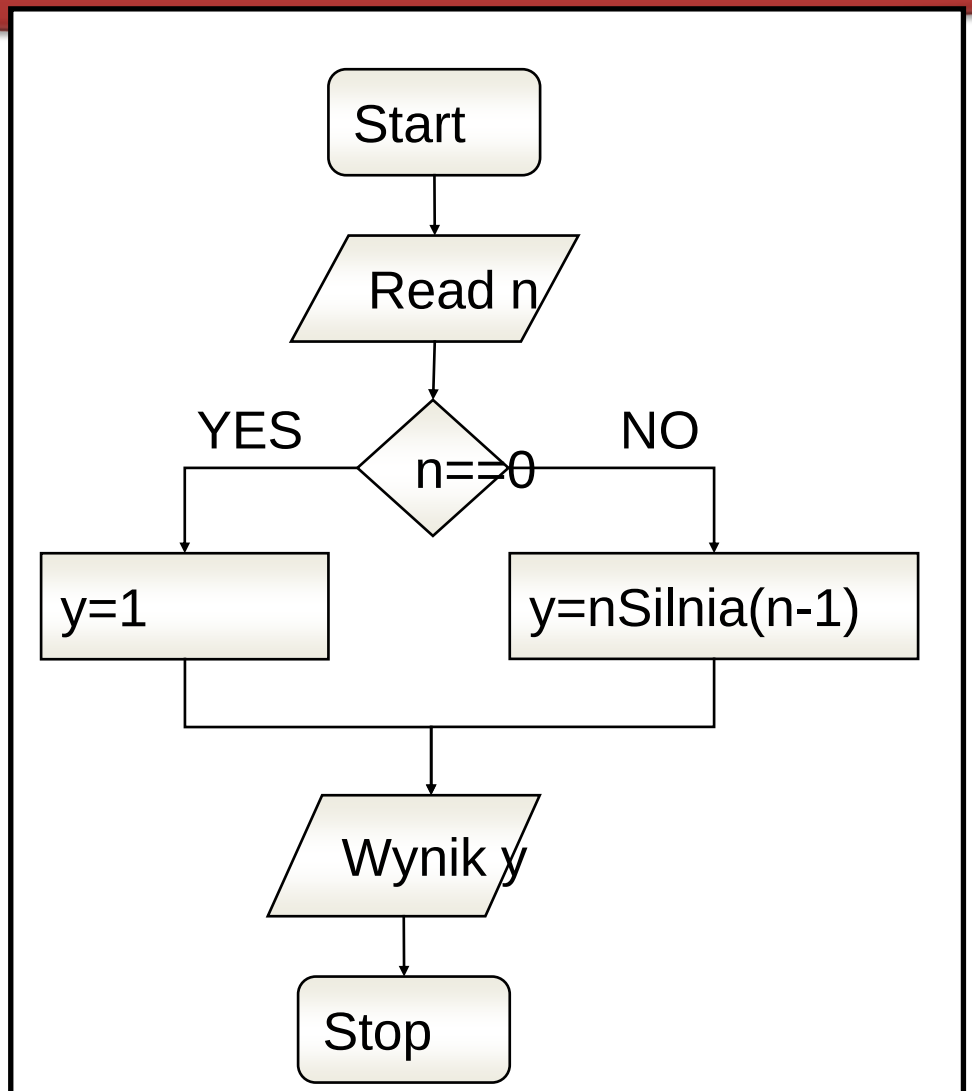
PĘTLA „POWTARZAJ AŻ”

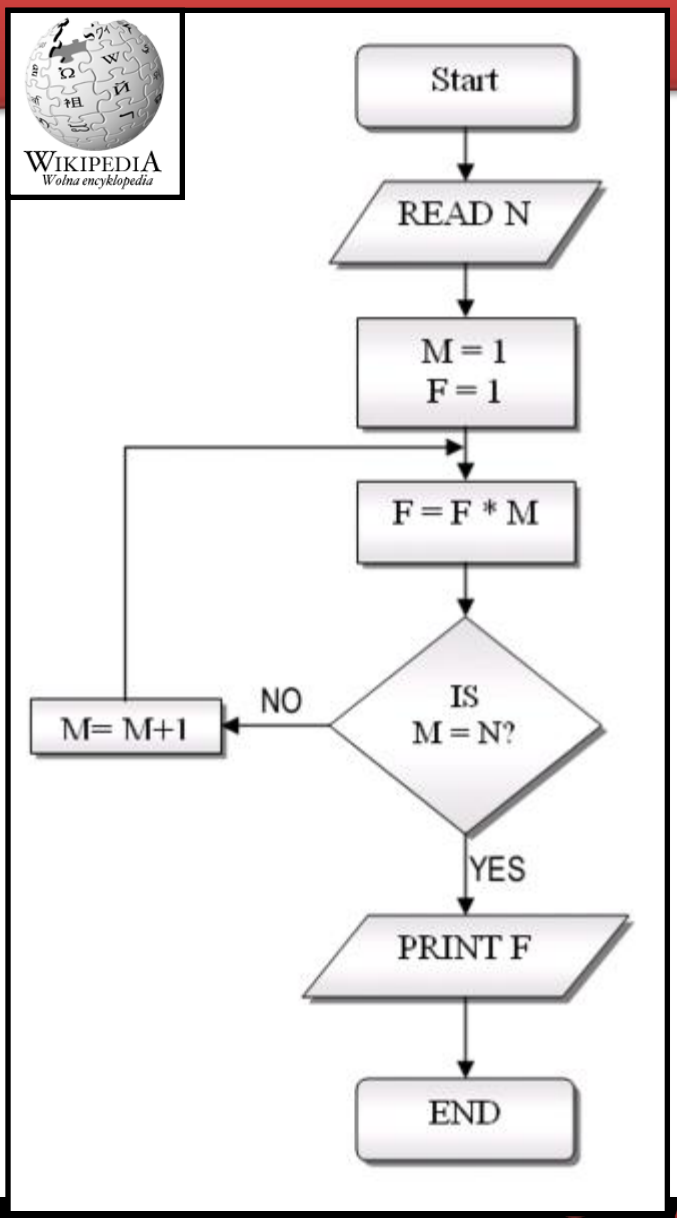
Schemat blokowy

N!



N! rekurencyjnie





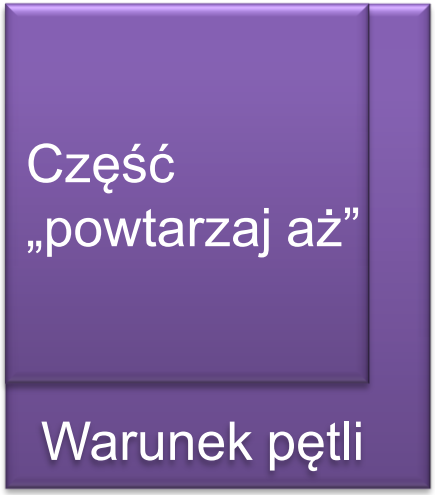
Diagramy ramkowe

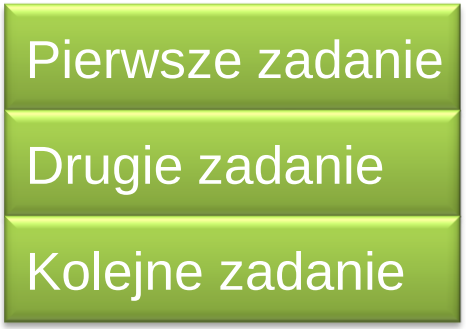
- Nie można nimi opisać programów łamiących zasady programowania strukturalnego
- Diagramy ramkowe (Box diagrams)
  - Schematy Nassiego-Shneidermana
  - Schematy N-S
  - Diagramy Chapina

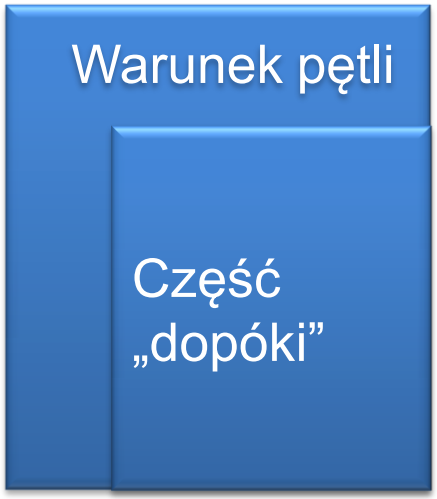
Diagramy ramkowe

- Dobrze określona i widoczna dziedzina funkcjonalna
  - Zasięg warunków i powtórzeń
- Nie da się opisać dowolnego rodzaju przepływu sterowania
- Łatwo określić zasięg widoczności lokalnych i globalnych danych
- Łatwo opisać rekurencję

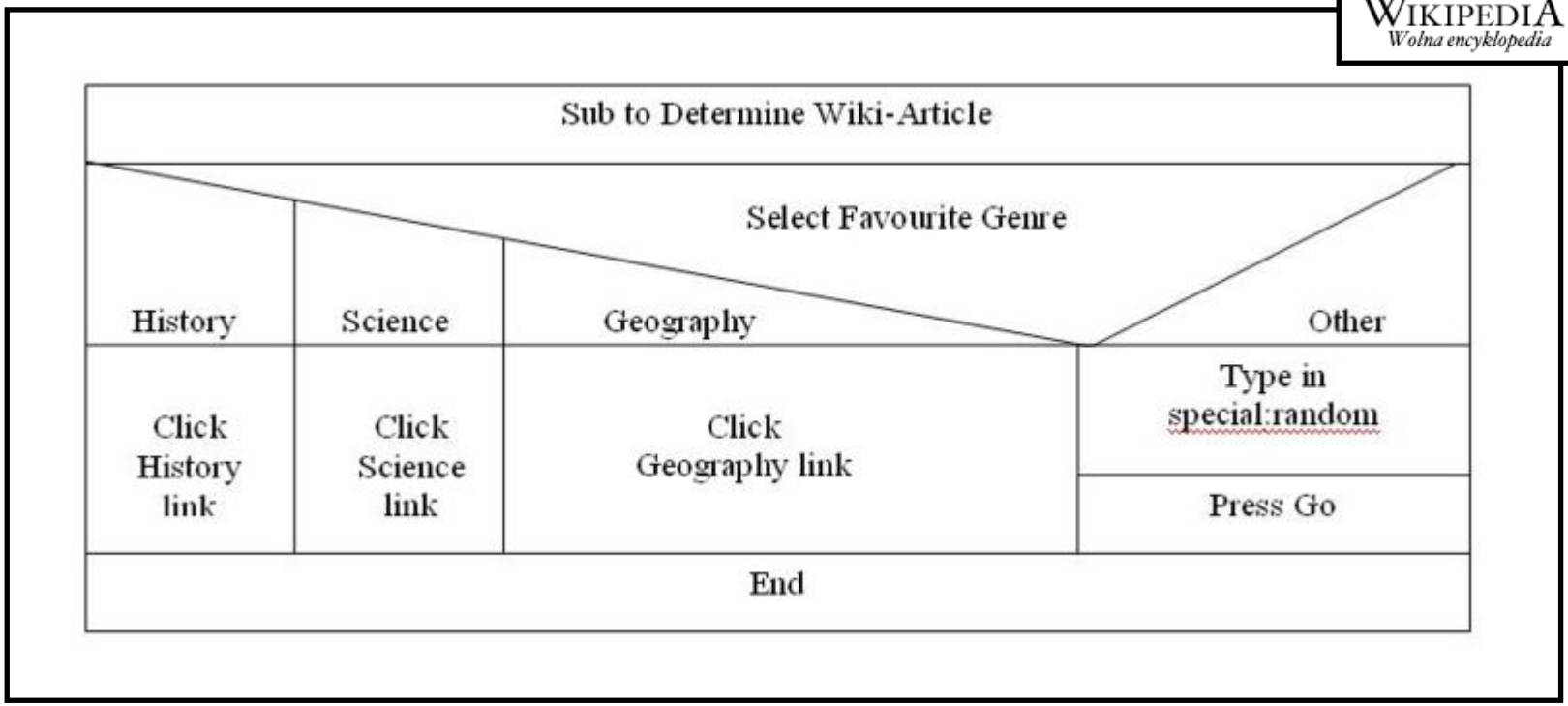
Diagramy ramkowe



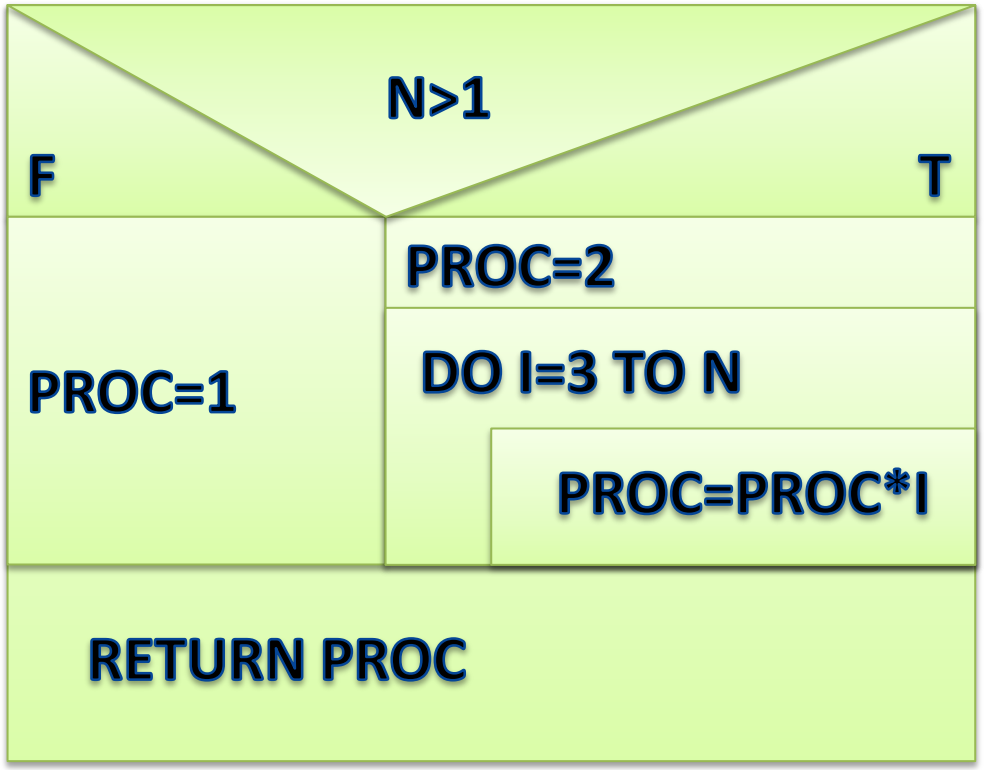


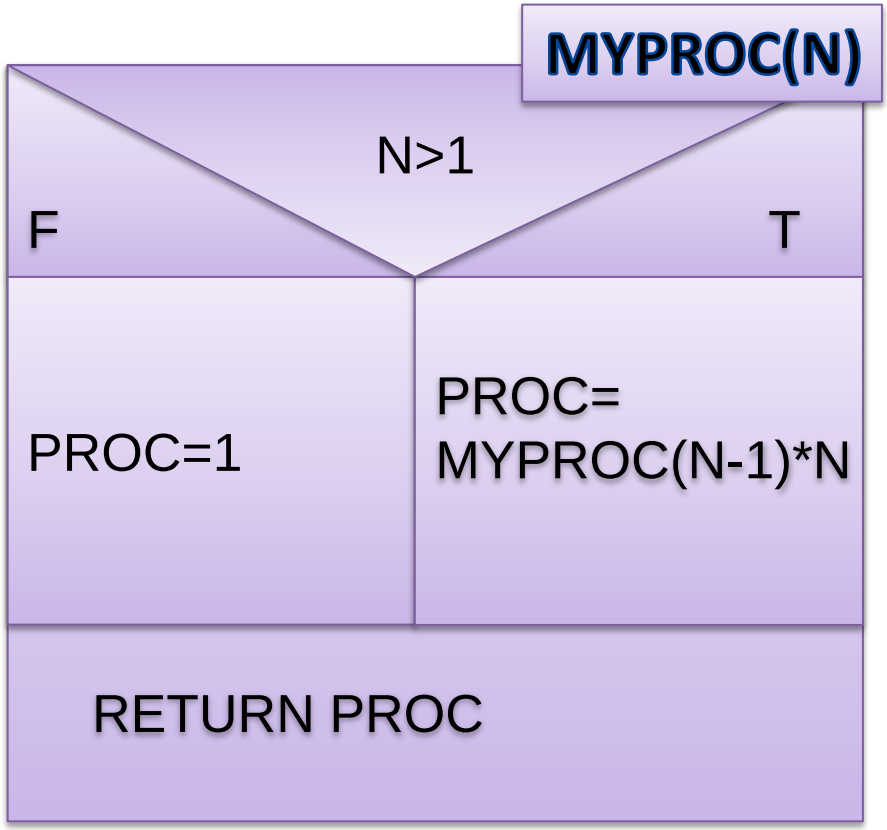


Diagramy ramkowe



Diagramy ramkowe





Tablicowa notacja projektowa

- Dodatkowy opis projektowanych procedur
- Służą do opisywania kombinacji warunków i czynności, które na ich podstawie należy wykonać

Tablicowa notacja projektowa

	Reguły				
Warunki	**1**	**2**	**3**		**n**
Warunek 1	√		√		
Warunek 2		√			
Warunek 3		√	√		
Czynności					
Czynność 1	√	√			
Czynność 2		√	√		
Czynność 3	√				

Tablicowa notacja projektowa

- Jeśli klient płaci rachunki w systemie stałych stawek, to za zużycie energii do 100 kWh nalicza się minimalną opłatę miesięczną. W przeciwnym razie stosuje się schemat płatności A. Jeśli jednak klient płaci w systemie zmiennych stawek, to za zużytą energię do 100 kWh stosuje się schemat płatności A, a powyżej 100 kWh – schemat płatności B.

Tablicowa notacja projektowa

	Reguły						
Warunki	**1**	**2**	**3**				**n**
Stałe stawki	T	T	F	F	F		
Zmienne stawki	F	F	T	T	F		
Zużycie do 100 kWh	T	F	T	F			
Zużycie ponad 100 kWh	F	T	F	T			
Czynności							
Min opłata miecięczna	√						
Schemat A		√	√				
Schemat B				√			

Pseudokod

- Zachowuje strukturę charakterystyczną dla kodu zapisanego w języku programowania
- Rezygnuje ze ścisłych reguł składniowych na rzecz prostoty i czytelności
- Nie zawiera szczegółów implementacyjnych
- Nietrywialne kroki algorytmu opisywane są przy pomocy formuł matematycznych lub zdań w języku naturalnym

Pseudokod

KOD

```
<?php
if (is_valid($cc_number)) {
    execute_transaction($cc_number, $order);
} else {
    show_failure();
}
?>
```

PSEUDOKOD

```
if credit card number is valid
    execute transaction based on number and order
else
    show a generic failure message
end if
```

Programowanie strukturalne

- Algorytmy są reprezentowane przy pomocy ograniczonego zestawu konstrukcji logicznych
- Jego istotą jest umożliwienie projektantom tworzenia mniej złożonych algorytmów, które łatwiej
  - Czytać
  - Testować
  - Pielęgnować

KONIEC