Uniwersytet Zielonogórski
Instytut Sterowania i Systemów Informatycznych

Laboratorium Algorytmów i Struktur Danych

Laboratorium 1
Schematy blokowe

1 Cel ćwiczenia

Ćwiczenie ma na celu zapoznanie z zagadnieniami związanymi z projektowaniem blokowych schematów algorytmicznych, symbolicznym przedstawieniem elementów algorytmów oraz metodologii procesu tworzenia blokowych schematów opisujących algorytmy.

2 Wstęp teoretyczny

Elementem poprzedzającym pisanie kodu programu powinno być przygotowanie schematu jego działania (algorytmu). Jednym ze sposobów przedstawiania algorytmów są schematy blokowe. Schemat blokowy jest to układ figur geometrycznych (nazywanych skrzynkami lub blokami) połączonych ze sobą odcinkami prostymi lub łamanymi (ścieżki sterujące).

Figury służą do przedstawiania rodzaju działań zaprojektowanych w algorytmie, zaś linie wskazują kolejność wykonywania tych działań. Każda figura w schemacie blokowym prezentuje określony rodzaj operacji. Zasadniczą zaletą schematów blokowych jest to, że graficznie prezentują one algorytm zarówno od strony występujących w nim działań, jak i ich kolejności.

3 Elementy schematu blokowego

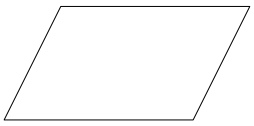
Elementami schematu blokowego są:

- Strzałki — określają kierunek przepływu danych lub kolejność wykonywania działań. Powinny składać się z linii prostych, należy unikać łuków oraz krzyżowania się linii.
- Operacja START (rys. 1) — oznacza punkt startu algorytmu (programu). Od tego bloku rozpoczyna się wykonywanie algorytmu. Występuje dokładnie jeden raz w każdym schemacie blokowym.
- Operacja STOP (rys. 2) — oznacza punkt zakończenia algorytmu (programu). Na tym bloku kończy się wykonywanie algorytmu. Najczęściej występuje jeden raz, jednakże dla zwiększenia czytelności schematu może zostać powtórzony wielokrotnie.

Rysunek 1: Operacja START

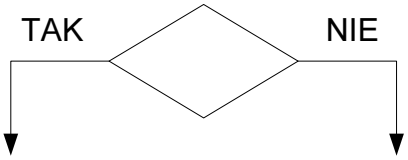
Rysunek 2: Operacja STOP

- Operacje odczytywania danych oraz wprowadzania wyników (rys. 3) — oznaczone za pomocą równoległoboków. W środku równoległoboku wpisuje się odpowiedni komentarz, odpowiadający żądanej operacji. Należy podkreślić, że **inne operacje, nie będące operacjami wejścia / wyjścia oznaczane są innym blokiem** — blokiem operacji.



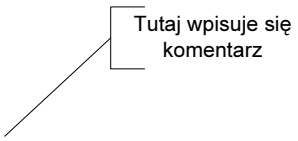
Rysunek 3: Operacja wejścia/wyjścia

- Operacja warunkowa JEŻELI (rys. 4) — oznaczona za pomocą rombu. Wewnątrz rombu znajduje się odpowiedni warunek. Słowo JEŻELI zazwyczaj pomija się. Operacje warunkowe **zawsze** prowadzą do konieczności rozważenia dwóch dróg: jednej (TAK) kiedy rozpatrywany warunek jest spełniony i drugiej kiedy rozpatrywany warunek nie jest spełniony (NIE). Blok ten odpowiada instrukcji warunkowej (*if*). Przy zapisie warunku powinno się używać operatorów matematycznych $=$, $\neq$, $<$, $\leqslant$, $>$, $\geqslant$, nie zaś zapisu znanego z języków programowania (np. ”>=”).



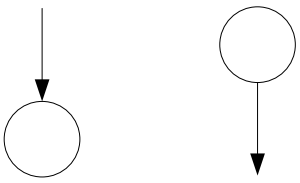
Rysunek 4: Operacja warunkowe

- Komentarz (rys. 5) — blok używany do wyjaśnienia znaczenia instrukcji zawartych w blokach. Tekst komentarza wpisuje się między kreskami. Ukośna kreska wskazuje (dotyka) komentowany blok.



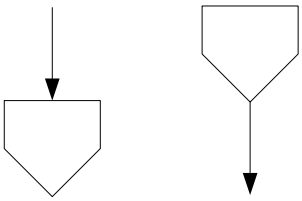
Rysunek 5: Komentarz

- Łącznik stronnicowy (rys. 6) — oznacza łączenie w obrębie jednej strony. Używany w przypadku, gdy użycie strzałki znacząco zmniejszyłoby czytelność schematu. Poszczególne łączniki oznacza się liczbami całkowitymi. Zawsze istnieją dwa łączniki oznaczone tym samym numerem: jeden źródłowy, drugi — docelowy. Strzałka skierowana jest do łącznika źródłowego oraz od łącznika docelowego.



Rysunek 6: Łącznik stronnicowy — źródłowy (po lewej stronie) i docelowy (po prawej stronie)

- Łącznik międzystronnicowy (rys. 7) — używany w przypadku, gdy konieczne jest przeniesienie sterowania między stronami (arkuszami papieru) schematu. Zasady jego użycia są analogiczne do zasad użycia łącznika stronnicowego. Poszczególne łączniki oznacza się liczbami całkowitymi. Zawsze istnieją dwa łączniki oznaczone tym samym numerem: jeden źródłowy, drugi — docelowy. Strzałka skierowana jest do łącznika źródłowego oraz od łącznika docelowego.



Rysunek 7: Łącznik międzystronnicowy — źródłowy (po lewej stronie) i docelowy (po prawej stronie)

- Blok operacji (rys. 8) — oznaczany za pomocą prostokąta, w którym wpisuje się komentarz określający daną operację. Jeśli kilka operacji tworzy logiczną całość, to wszystkie one mogą być umieszczone w jednym bloku. Nie zaleca się umieszczania tam zbyt dużej liczby operacji nawet wtedy, kiedy są one powiązane ze sobą bezpośrednio, ponieważ może to zmniejszyć czytelność schematu. Instrukcje przypisania oznacza się przy pomocy strzałki ($\leftarrow$). Instrukcję Pascalową "$x := y$;" na schemacie powinno się oznaczyć jako "$x \leftarrow y$". Jeśli zachodzi potrzeba użycia tablic, n-ty element można oznaczyć jako $tablica[n]$. Należy podkreślić, że **operacje wejścia / wyjścia oznaczane są innym blokiem** — blokiem operacji odczytywania danych oraz wprowadzania wyników.

Rysunek 8: Blok operacji

- Blok proceduralny (rys. 9) — proces określony poza programem i z tego powodu nie wymagający zdefiniowania w rozpatrywanym programie (podprogram), przy definiowaniu tych elementów należy pamiętać, że ich wykonanie nie rozpoczyna się od START i nie kończy STOP (najczęściej podprogram kończy wykonanie instrukcji powrotu (RETURN) do głównego programu)

Rysunek 9: Blok proceduralny

- Punkt koncentracji (rys. 10) — blok używany dla podniesienia czytelności schematu. Oznacza miejsce, do którego wpływa kilka strzałek. Rysuje się go w postaci punktu

Rysunek 10: Punkt koncentracji

4 Zasady tworzenia schematów blokowych

1. Schemat powinien być prosty i co za tym idzie czytelny. W razie złożonego rozwiązania schemat należy podzielić na mniejsze części (podprogramy) i zamieścić je na osobnych arkuszach.

2. Schemat blokowy powinien być jednakowo dobrze rozumiany przez programistów posługujących się różnymi językami programowania. Z tego powodu nie powinno się używać w schematach składni jakiegokolwiek języka programowania. Preferowane jest użycie operatorów matematycznych, rozumianych przez wszystkich (np. zamiast "$y := SQRT(X)$" należy napisać "$y \leftarrow \sqrt{x}$ ").

3. Do rysowania schematów dobrze jest używać szablonów. Dzięki takiemu rozwiązaniu polepsza się czytelność schematu.

4. W blokach niezbędne jest komentowanie zarówno zaprojektowanych operacji, jak i kolejności ich wykonywania. Komentarze powinny być krótkie, lecz dostatecznie dokładnie wyjaśniające znaczenie opisywanych elementów.

5. Należy unikać rysowania przecinających się ścieżek sterowania. W razie konieczności lepiej jest wprowadzić odpowiednie łączniki, które pozwolą na wyeliminowanie niektórych linii wskazujących kolejność działań w algorytmie.

6. Należy dokładnie numerować arkusze, na których został rozrysowany schemat blokowy.

7. Trzeba liczyć się z możliwością wystąpienia konieczości wprowadzenia poprawek do schematu, dlatego wskazane jest tak tworzyć arkusze, aby możliwe było naniesienie poprawek bez konieczności przerysowywania całego schematu.

8. Należy unikać zarówno zbyt dużej szczegółowości jak i zbytniej ogólności schematów. Oczywiste operacje można zapisać w formie pseudokodu (np. "Odczytaj z pliku liczbę x").

9. Nie należy umieszczać zbyt dużej liczby operacji w jednym bloku.

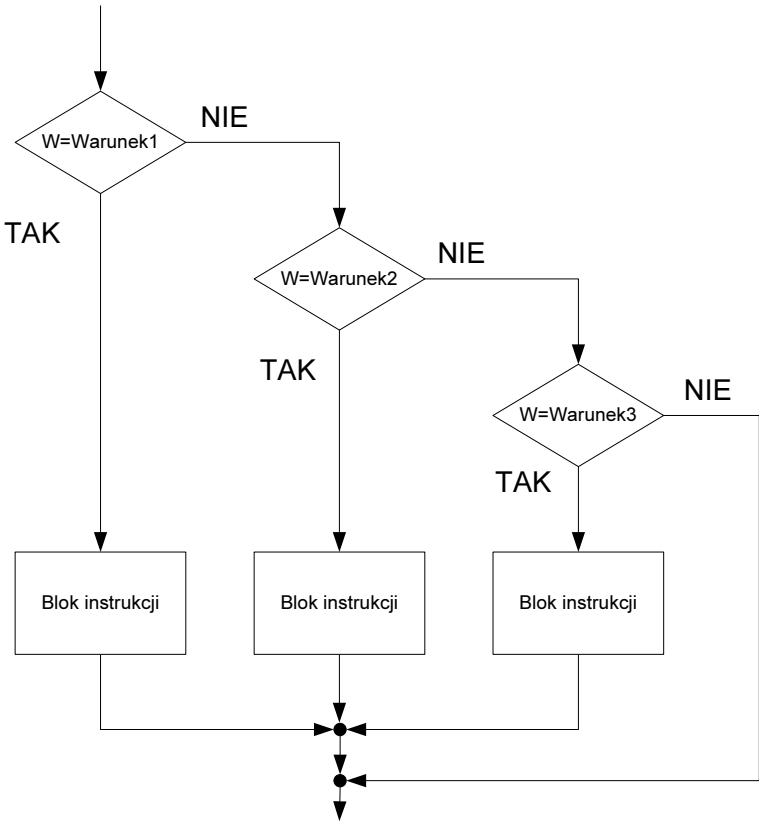
10. Operacja warunkowa IF zawsze prowadzi do konieczości rozważenia dwóch dróg: gdy warunek jest spełniony i gdy nie jest.

5 Realizacje pętli i instrukcji warunkowych

Wszystkie fragmenty programów zostały napisane w języku Pascal.

5.1 Instrukcja case

Schemat blokowy postaci



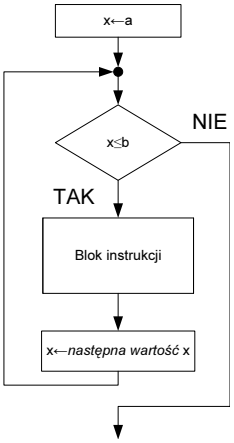
Rysunek 11: Instrukcja CASE

Odpowiada następującemu fragmentowi programu

```
case W of
  Warunek1 : begin
               {Blok instrukcji}
             end;
  Warunek2 : begin
               {Blok instrukcji}
             end;
  Warunek3 : begin
               {Blok instrukcji}
             end;
end;
```

5.2 Pętla FOR

Schemat blokowy postaci



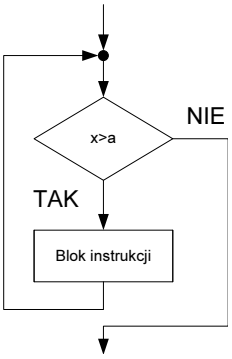
Rysunek 12: Pętla FOR

Odpowiada następującemu fragmentowi programu

```
for x:=a to b do begin
  {Blok instrukcji}
end;
```

5.3 Pętla WHILE

Schemat blokowy postaci



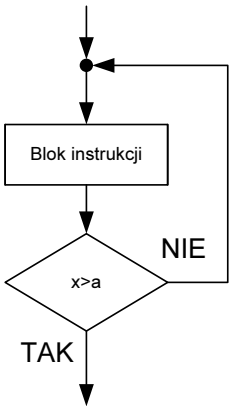
Rysunek 13: Pętla WHILE

Odpowiada następującemu fragmentowi programu

```
while x>a do begin
  {Blok instrukcji}
end;
```

5.4 Pętla REPEAT—UNTIL

Schemat blokowy postaci



Rysunek 14: Pętla REPEAT—UNTIL

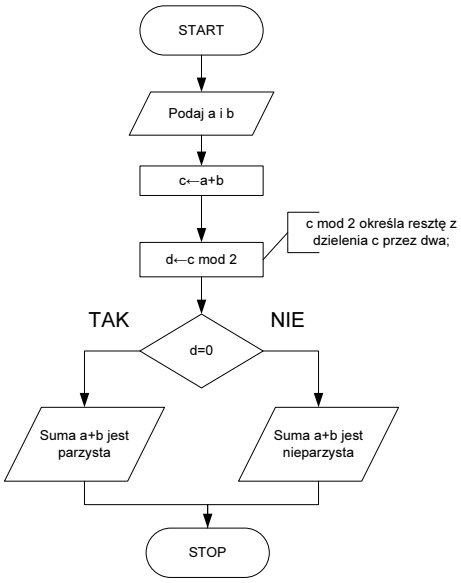
Odpowiada następującemu fragmentowi programu

```
repeat
  {Blok instrukcji}
until x>a;
```

Należy zwrócić uwagę na różnice pomiędzy pętlami WHILE oraz REPEAT—UNTIL. W przypadku pętli WHILE, jeśli warunek nie jest spełniony, instrukcje wewnątrz pętli nie są wykonywane ani razu. Pętla REPEAT—UNTIL wykonuje się co najmniej jeden raz; warunek sprawdzany jest na końcu. Ponadto należy zwrócić uwagę na odwrócenie warunku wykonania pętli. Ciało pętli WHILE **wykonywane** jest, jeśli warunek jest spełniony. Pętla REPEAT—UNTIL **przerywana** jest w przypadku spełnienia warunku — odwrotnie do pętli WHILE.

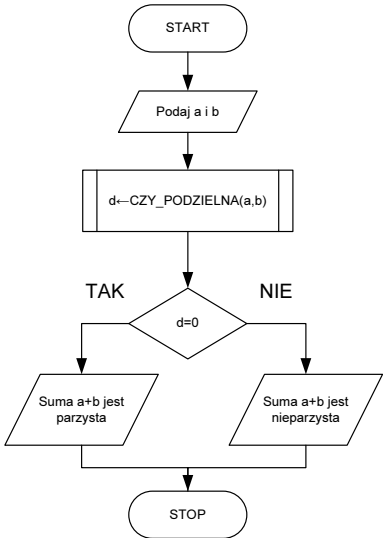
6 Przykład

Algorytm sprawdzający, czy suma dwóch liczb podanych przez użytkownika jest parzysta.

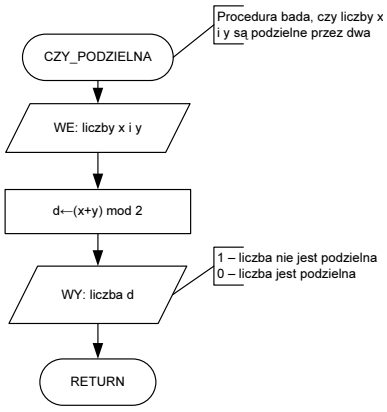


Rysunek 15: Przykładowy algorytm

Ten sam algorytm zapisany przy użyciu bloczka proceduralnego przedstawiono na rys. (16) i (17):



Rysunek 16: Program główny



Rysunek 17: Procedura

7 Pytania sprawdzające

Poniższe pytania mają pomóc w przyswojeniu materiału.

1. Ile razy w schemacie blokowym może wystąpić blok "START"? A blok "STOP"?
2. Przy pomocy jakiego bloku oznaczyć operację "odczytaj liczbę x z pliku"? (Wskazówka: jest to operacja wejścia/wyjścia)
3. Narysować schemat blokowy programu wyznaczającego średnią arytmetyczną dwóch liczb.
4. Narysować schemat blokowy programu wyznaczającego sumę liczb podawanych z klawiatury tak długo, aż użytkownik nie wprowadzi liczby zero.

Literatura

[1] P. Wróblewski. Algorytmy, struktury danych i techniki programowania, Helion, 1996.

[2] T.H.Cormen i in. Wprowadzenie do algorytmów, WNT 2000.