Wprowadzenie do programowania

Podstawowe pojęcia

- Programowanie komputerowe – to proces tworzenia programu komputerowego, który jest zestawem instrukcji (komend, poleceń) dla komputera przygotowanym w celu uzyskania określonego rezultatu.

Proces tworzenia programu komputerowego obejmuje etapy:

- Zdefiniowanie problemu do rozwiązania
- Planowanie rozwiązania
- Kodowanie programu
- Testowanie programu
- Dokumentowanie programu

Definiowanie Problemu

Zdefiniowanie problemu do rozwiązania polega na zdefiniowaniu i określeniu wejścia oraz wyjścia programu. Wejście programu opisuje dane wejściowe, które program ma przetwarzać. Opis ten powinien uwzględniać analizę danych wejściowych pod względem jakościowym (rodzaje – typy danych, źródła), jak również ilościowym (liczba danych należących do poszczególnych grup tematycznych). Wyjście programu zaś powinno stanowić odpowiedź na pytanie: co ma być wynikiem działania programu? W tym przypadku również należy przeprowadzić analizę ilościową i jakościową

Planowanie rozwiązania

Planowanie rozwiązania problemu to drugi etap procesu tworzenia programu komputerowego. Jest on zwykle realizowany na jeden z dwóch sposobów.

Pierwszy polega na sporządzeniu schematu blokowego algorytmu rozwiązania problemu. Przy czym algorytm należy rozumieć jako ciąg czynności lub sposób postępowania prowadzący do rozwiązania problemu albo wykonania określonego zadania w skończonej liczbie kroków, a **schemat blokowy** to mapa graficzna opisująca jednoznacznie logikę i funkcjonalność programu.

Drugi z wymienionych sposobów polega na wykorzystaniu pseudokodu. Pseudokod pozwala precyzyjnie zapisać algorytm programu za pomocą konstrukcji programistycznych zapożyczonych z wybranego języka programowania (pętli, podprogramów), zgodnych z określoną konwencją i stylem

Kodowanie programu

Kodowanie programu – polega na zapisywaniu algorytmu programu w konkretnym języku programowania.

Język programowania najprościej można określić jako zbiór reguł i zasad, które zapewniają skuteczną komunikację programisty z maszyną w celu realizacji przez komputer wymaganych zadań. Operacje te są opisane w sposób jednoznaczny w programie komputerowym, który pisze programista

Kodowanie programu

- Języki programowania można podzielić według różnych kryteriów. Np. języki do tworzenia aplikacji internetowych można podzielić na zapewniające interface (interakcję) aplikacji z użytkownikiem po stronie klienta oraz za takie odpowiadają za funkcjonalność po stronie serwera.
- Innym kryterium podziału jest związany ze sposobem konwersji kodu źródłowego napisanego w języku wysokopoziomowym zrozumiałym dla programisty, na kod maszynowy zrozumiały dla komputera.
- Język kompilowany to taki, w którym proces translacji polega na kompilacji. W czasie kompilacji program źródłowy jest tłumaczony na kod pośredni, który jest kodem obiektowym, a następnie integrowany z innymi elementami aplikacji w procesie linkowania czego efektem jest program wykonywalny
- Język interpretowany pozwala na translację kodu źródłowego przez jego interpretowanie. Interpreter tłumaczy kod programu instrukcja po instrukcji i wykonuje każdą z nich zaraz po przetłumaczeniu.

Testowanie programu

- **Testowanie programu** obejmuje jego weryfikację oraz walidację. **Weryfikacja programu** polega na udowodnieniu, że spełnia on założone kryteria dotyczące funkcjonalności, które są zazwyczaj określone w formalnej specyfikacji.
- **Walidacja programu** pozwala określić czy program jest w pełni dostosowany do potrzeb zamawiającego.

Testy programów można umownie podzielić na kilka poziomów:

- Testy jednostkowe
- Testy akceptacyjne
- Testy systemowe
- Testy integracji i systemowe elementów składowych

Z etapem kodowania i testowania jest ściśle powiązany proces debugowania. Pojęcie to oznacza wykrywanie, lokalizowanie, eliminowanie i korygowanie błędów logicznych w programie.

Dokumentacja

- Dokumentacja techniczna programu obejmuje opis problemu, schematy blokowe i/lub pseudokod, opisy rekordów danych, listę modułów, opis ważniejszych podprogramów, wyniki przeprowadzonych testów itp. W skład dokumentacji wchodzą również komentarze zawarte przez programistę.
- Dokumentacja użytkownika końcowego składa się zazwyczaj ze szczegółowej instrukcji obsługi programu wraz z opisem konkretnych przykładów zastosowań.

Podstawy algorytmiki

- Algorytmika to dział informatyki, który obejmuje tematykę projektowania i analizy algorytmów.
- Algorytm to ciąg czynności lub sposób postępowania prowadzący do rozwiązania problemu albo wykonania określonego zadania w skończonej liczbie kroków.
- Zadaniem algorytmu jest zmiana – przeprowadzenie zadanego stanu początkowego w wymagany stan końcowy, reprezentowany np. przez wartości końcowe zmiennych . Wspomniane zmienne mogą być zorganizowane w strukturach danych zdefiniowanych w programie.

Podstawy algorytmiki

Każdy algorytm powinien spełniać następujące kryteria:

- Dane wejściowe powinny być określone w sposób jednoznaczny z uwzględnieniem ich typów i liczby.
- Sterowaniem przebiegiem działania programu powinno zapewniać uzyskanie wyniku
- Liczba kroków zmierzających do uzyskania rezultatu musi być skończona
- Każdy krok powinien być zdefiniowany jednoznacznie

Podstawy algorytmiki

Algorytmy programów można zapisać w różny sposób, np w postaci:

- Schematu blokowego
- Pseudokodu
- Opisu słownego
- Listy kroków
- Drzewa (grafu)

Schematy blokowe algorytmów

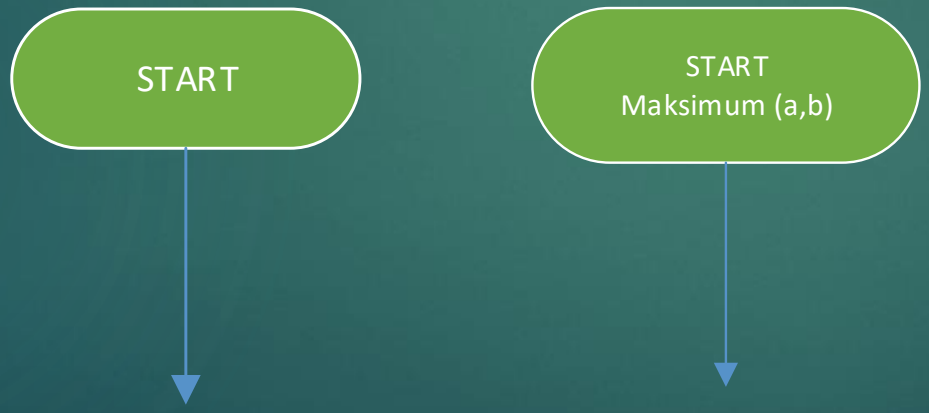
- Schemat blokowy algorytmu składa się z bloków funkcjonalnych reprezentujących operacje (polecenia, instrukcje). Różnym grupom poleceń odpowiadają różne kształty bloków. Ponadto każdy schemat blokowy zawiera linie – strzałki, które określają wzajemne powiązania funkcjonalne pomiędzy blokami oraz kolejność ich wykonywania.

Do najważniejszych bloków należą

- Blok początkowy
- Blok końcowy
- Blok wejścia/wyjścia
- Blok operacji (przetwarzania)
- Blok decyzyjny
- Blok podprogramu
- Łączniki
- Punkty koncentracji
- Blok komentarza

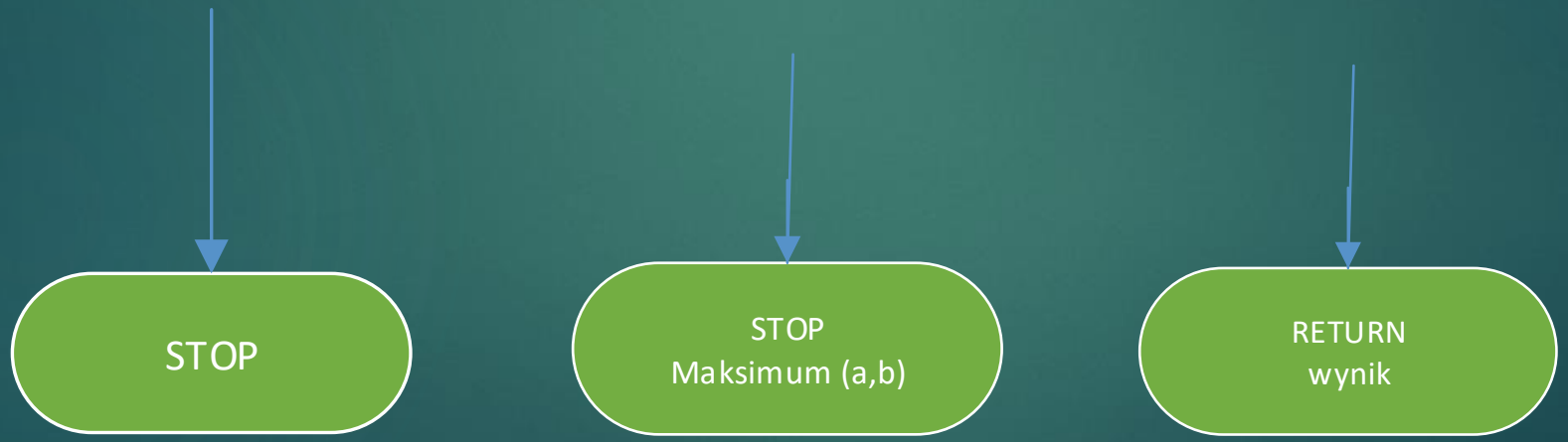
Blok początkowy

- Blok początkowy reprezentuje na schemacie blokowym początek algorytmu programu lub podprogramu, np. funkcji. Wewnątrz bloku znajduje się słowo START, jeśli algorytm dotyczy całego programu, albo słowo START z nazwą podprogramu oraz ewentualnie z jego parametrami/argumentami



Blok końcowy

- Blok końcowy oznacza koniec schematu blokowego algorytmu programu, albo algorytmu podprogramu (funkcji). Blok końcowy może występować w schemacie blokowym programu (podprogramu) więcej niż raz.

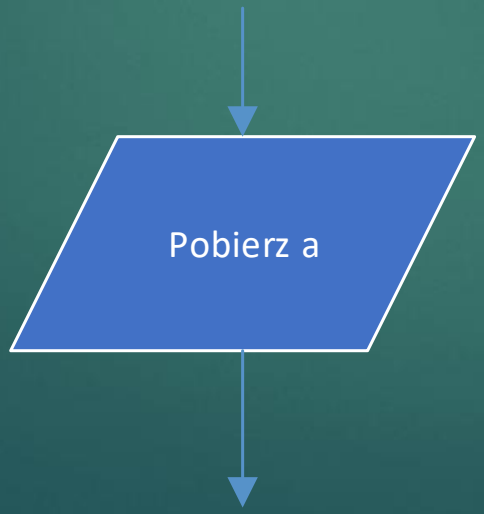


Blok końcowy

- Jeśli schemat blokowy dotyczy algorytmu podprogramu i jest niezależny od innych schematów blokowych , wówczas w nawiasach okrągłych należy podać nazwy parametrów tego podprogramu. Jeśli zaś schemat blokowy stanowi część innego schematu blokowego, wówczas w nawiasach okrągłych należy wpisać argumenty wywołania tego podprogramu.
- Jeśli podprogram zwraca na zewnątrz wartość za pośrednictwem swojej nazwy, wtedy koniec algorytmu może być oznaczony za pomocą słowa RETURN wraz z określeniem zwracanej wartości. Jest to informacja o powrocie do otoczenia podprogramu, np. programu głównego.

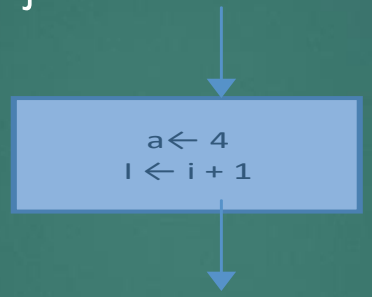
Blok wejścia/wyjścia

- Blok wejścia/wyjścia reprezentuje na schemacie blokowym algorytmu operacje pobierania (odczytu danych z urządzenia wejściowego), oraz prezentacji (wyświetlenia danych na urządzeniu wyjściowym np. ekranie monitora).
- Blok wejścia/wyjścia może również być wykorzystywany w celu wyświetlenia komunikatu informacyjnego dla użytkownika, np. dotyczącego błędu.

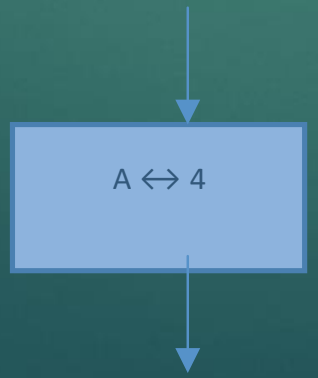


Blok operacji

- Blok operacji oznacza na schemacie operację przetwarzania danych (np. instrukcję przypisania), które nie należą do grupy operacji wejścia/wyjścia. Do oznaczania przypisania używa się strzałki blokowej skierowanej od wyrażenia znajdującego się z prawej strony do zmiennej z lewej.

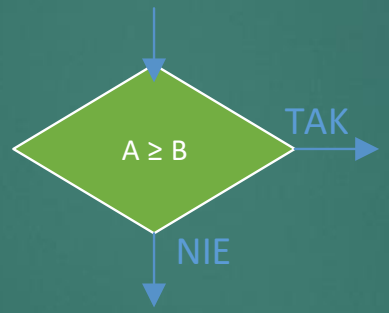


Operacje zamiany wartości są w blokach przetwarzania oznaczone za pomocą strzałki dwukierunkowej



Blok decyzyjny

- Blok decyzyjny reprezentuje na schemacie blokowym warunek. Blok ten ma jedno wejście oraz dwa wyjścia. Jedno z tych wyjść jest oznaczone jako Tak i odpowiada spełnieniu warunku (TRUE, 1), drugie zaś jest opisane jako NIE i odpowiada nie spełnieniu warunku (FALSE, 0)

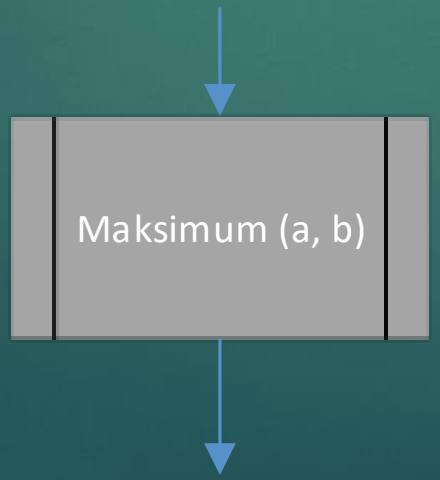


W blokach decyzyjnych nie należy używać operatorów relacyjnych znanych z kodów źródłowych programów, takich jak: ==(równy), !=(różny), <=(mniejszy lub równy), >=(większy lub równy). Na schematach blokowych warunki należy formułować przy użyci standardowego, znormalizowanego zapisu matematycznego, a więc przy wykorzystaniu operatorów: =, ≠, <, >, ≤, ≥. To samo dotyczy operacji sumy i iloczynu logicznego $\vee$, $\wedge$

W praktyce bloki decyzyjne występują na schematach blokowych zawierających opis instrukcji warunkowych (np. if, if-else, switch) oraz pętli programowych (np. while, do-while).

Blok podprogramu

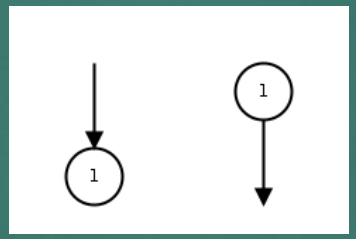
Blok podprogramu oznacza na schemacie blokowym wywołanie określonego podprogramu (np. funkcji). Przy tym podprogram traktuje się wówczas jako tzw. „czarną skrzynkę” bez zajmowania się szczegółami dotyczącymi jego algorytmu. W bloku podprogramu należy podać nazwę wywoływanego podprogramu wraz z listą argumentów. W razie potrzeby algorytm podprogramu opisujemy na oddzielnym schemacie blokowym.



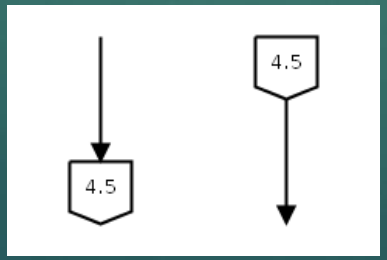
Łączniki

Dany schemat blokowy może być mało przejrzysty i czytelny, jeśli jest przedstawiony na pojedynczej stronie, np. z powodu dużej złożoności i/lub występowania przecinających się linii. Aby tego uniknąć, można przenieść jego wybrane fragmenty w inne miejsce na tej samej stronie lub na inną , dodatkową stronę

- Łączniki wewnętrzne reprezentują połączenia w obrębie jednej strony



- Łączniki zewnętrzne reprezentują łączenie w ramach dwóch stron



Punkty koncentracji i komentarze

- Bloki funkcjonalne na schematach blokowych połączone są za pomocą linii, które zakończone są strzałkami. Strzałki owe określają kolejność wykonywania działań zdefiniowanych za pomocą poszczególnych bloków. Jeśli co najmniej dwa groty strzałek koncentrują się w jednym punkcie na schemacie, wówczas należy ten punkt oznaczyć jako punkt koncentracji

Blok komentarza jest używany do skomentowania znaczenia instrukcji zawartych w wybranym bloku. Tekst komentarza należy wpisać pomiędzy kreskami pionowymi.

Zasady tworzenia schematów blokowych

- Schemat blokowy powinien być zrozumiały dla programistów wykorzystujących różne języki programowania. Dlatego też nie należy wykorzystywać składni ani operatorów charakterystycznych dla jednego języka programowania
- Jeśli schemat blokowy algorytmu jest przedstawiony na wielu arkuszach wówczas poszczególne strony należy numerować zgodnie z przyjętą konwencją. Zaleca się wykonanie spisu wszystkich stron wchodzących w skład schematu blokowego wraz z niezbędnym opisem, co zawiera każda ze stron.
- Schemat blokowy powinien być przejrzysty i czytelny. Jeśli problem do rozwiązania jest złożony, należy go podzielić na części i dla każdej z nich utworzyć oddzielny schemat blokowy. Schematy cząstkowe można łączyć ze sobą za pomocą łączników na schemacie ogólnym, zamieszczonym na osobnej stronie. Innym rozwiązaniem jest wykorzystanie na schemacie blokowym ogólnym bloków wywołania podprogramów.

Zasady tworzenia schematów blokowych

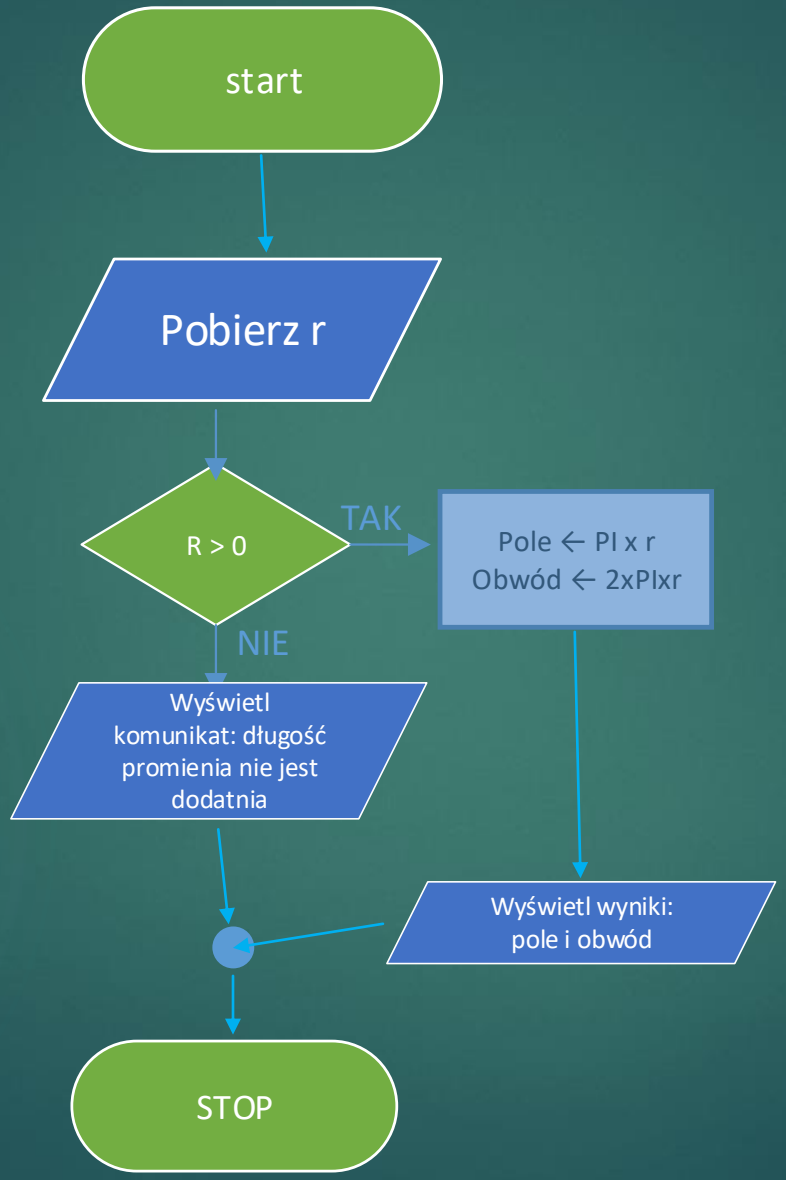
- Nie należy zbytnio zagęszczać bloków funkcjonalnych na schemacie blokowym w celu umożliwienia jego ewentualnej modyfikacji i/lub poprawy.
- Groty strzałek powinny być skierowane albo w dół, albo w prawo oraz pod kątem prostym.
- Zalecane jest, aby na schematach blokowych unikać stosowania ścieżek (linii), które się przecinają. Wynika to z faktu, że przecinające się linie znacznie zmniejszają przejrzystość i czytelność schematu. W razie konieczności należy wykorzystać łączniki wewnątrzstronicowe by podzielić schemat na mniejsze części.

Zasady tworzenia schematów blokowych

- W zależności od wymagań zewnętrznych i/lub wewnętrznych każdy blok funkcjonalny na schemacie blokowym powinien reprezentować przyjętą (stałą) liczbę odpowiadających mu operacji. Na schemacie blokowym nie powinno być tak, że np. dany blok odpowiada pojedynczej operacji prostej, a inny reprezentuje kilka operacji złożonych.
- Opis algorytmu programu w postaci schematu blokowego można łączyć z innymi formami zapisu schematów blokowych , np. opisem słownym czy pseudokodem.
- Schemat blokowy powinien być właściwie i wyczerpująco skomentowany. Komentarze można wstawiać bezpośrednio na schemacie , za pomocą bloków komentarza odpowiadających poszczególnym lub wybranym blokom funkcjonalnym, albo na dodatkowych arkuszach, na których zamieszcza się szczegółowy słowny opis danego bloku.

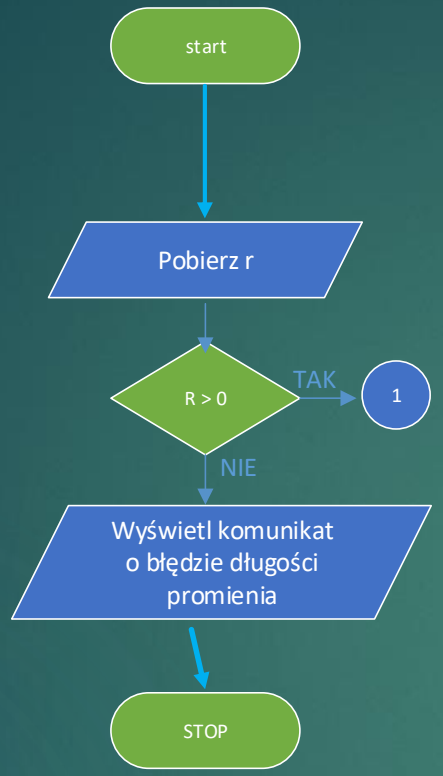
Przykłady schematów blokowych

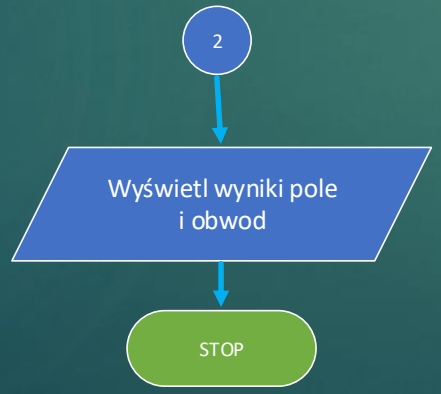
- Na rysunku poniżej przedstawiono schemat blokowy programu obliczającego pole i obwód koła. Długość promienia jest wprowadzana z klawiatury, a wyniki wyświetlane na ekranie. Obliczenia wykonywane są tylko wtedy, gdy wprowadzona wartość promienia koła jest dodatnia. W przeciwnym wypadku na ekranie monitora powinien się wyświetlić komunikat o błędzie
- W przedstawionym schemacie blokowym zilustrowano praktyczne wykorzystanie następujących bloków: początkowego i końcowego, wejścia/wyjścia, operacji, decyzyjnego i komentarza. Blok wejściowy wykorzystuje do pobrania długości promienia koła r z klawiatury. Zadaniem bloku decyzyjnego jest ustalenie czy r jest wartością dodatnią, jeśli tak wykonywane są niezbędne obliczenia, a następnie wyniki są wyświetlane na ekranie (pole i obwód koła). W przeciwnym wypadku wyświetlany jest komunikat o błędzie.

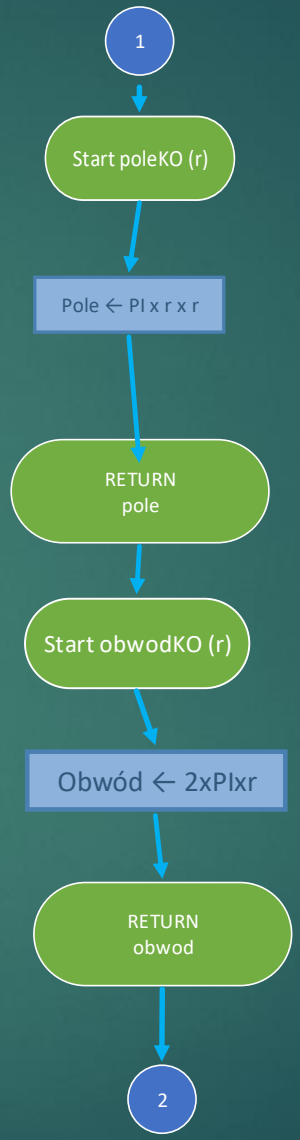

start
Pobierz r
R > 0
TAK
Pole ← PI x r
Obwód ← 2xPIxr
NIE
Wyświetl komunikat: długość promienia nie jest dodatnia
Wyświetl wyniki: pole i obwód
STOP

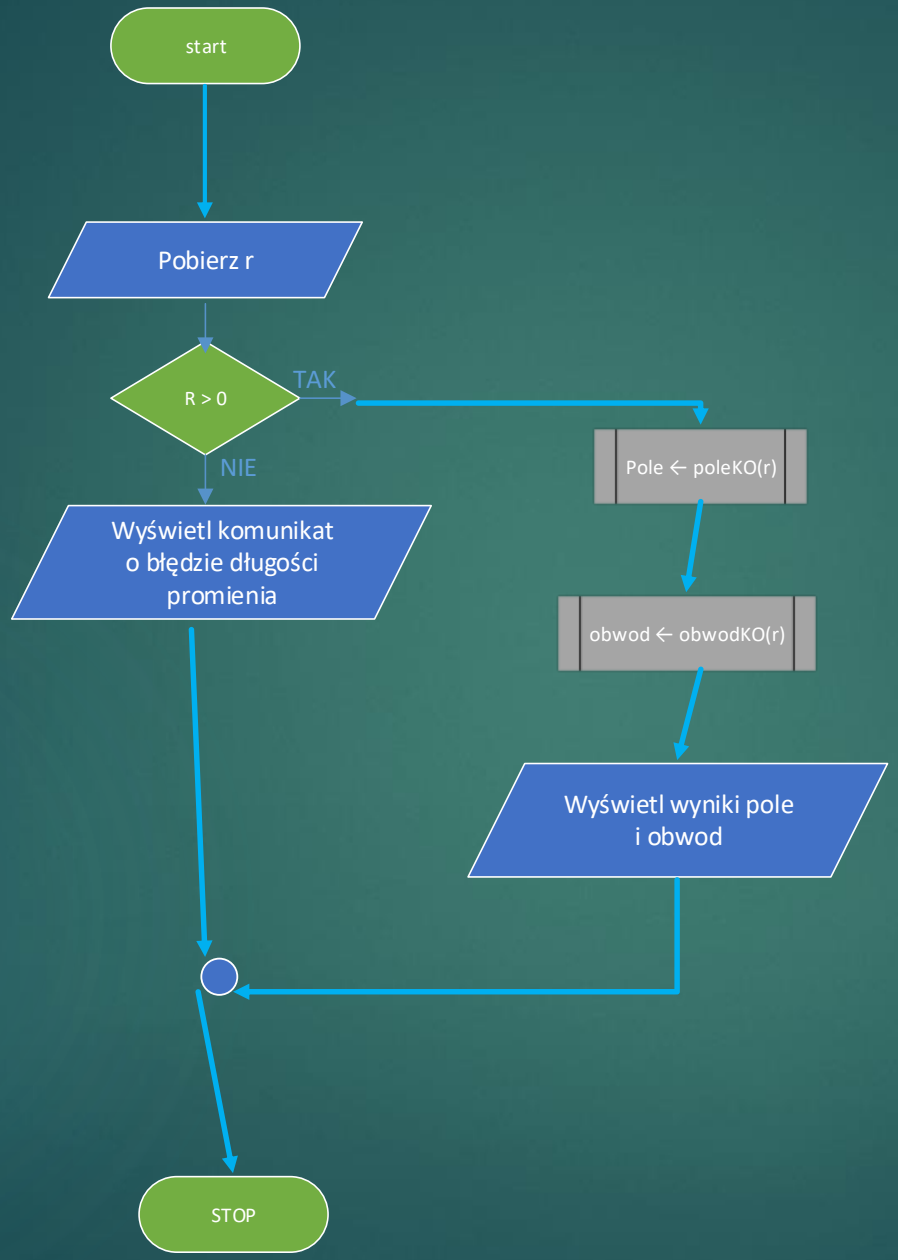
Przykłady schematów blokowych

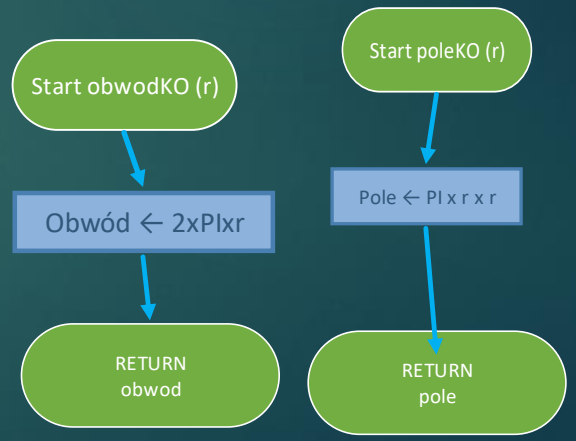
- Poniżej przedstawiono dwa przykłady schematu blokowego programu, którego treść przedstawiona została powyżej
- Program różni się od poprzedniego wykorzystaniem dwóch podprogramów o nazwach : poleKO, obwodKO. Zadaniem podprogramu poleKO jest obliczenie pola koła, zaś obwodKo oblicza jego obwód. Do każdego z tych programów dostarczane są pojedyncze dane wyjściowe – długość promienia koła reprezentowaną prze zmienną r. Wyniki działania są zwracane za pośrednictwem nazw podprogramów.
- W wariancie pierwszym definicje są zawarte bezpośrednio na schemacie blokowym ogólnym, dotyczącym całego programu. Bloki wspomnianych definicji zawierają nazwy podprogramów oraz ich argumenty. Bloki końcowe są opisane za pomocą słowa return


start
Pobierz r
R > 0
TAK
1
NIE
Wyświetl komunikat
o błędzie długości
promienia
STOP


2
Wyświetl wyniki pole
i obwod
STOP


1
Start poleKO (r)
Pole ← PI x r x r
RETURN
pole
Start obwodKO (r)
Obwód ← 2xPIxr
RETURN
obwod
2


start
Pobierz r
R > 0
TAK
NIE
Wyświetl komunikat
o błędzie długości
promienia
Pole ← poleKO(r)
obwod ← obwodKO(r)
Wyświetl wyniki pole
i obwod
STOP


Start obwodKO (r)
Obwód ← 2xPIxr
RETURN
obwod

Start poleKO (r)
Pole ← PI x r x r
RETURN
pole

Przykłady schematów blokowych

- Schemat blokowy algorytmu całego programu zawiera jedynie bloki wywołania podprogramów ***poleKO i obwodKO,*** a nie ich kompletne definicje. We wspomnianych blokach wywołań podprogramów są zawarte ich nazwy wraz z argumentem wywołania ***r***, a także przypisanie ich działań do konkretnych zmiennych.
- Schemat przedstawiony powyżej jest bardziej czytelny i przejrzysty od schematu przedstawionego na slajdzie 28, pomimo, że oba dotyczą algorytmu tego samego programu.

Pseudokod

- Algorytmy programów oraz ich części składowych można również zapisywać przy użyciu tzw. pseudokodu. Podobnie jak schemat blokowy pseudokod to opis algorytmu niezależny od systemu operacyjnego, języka programowania czy środowiska programistycznego.
- Najważniejszą różnica pomiędzy algorytmem programu określonym za pomocą pseudokodu, a samym algorytmem zapisanym w konkretnym języku programowania jest to, że pseudokod jest zrozumiały także dla kogoś, kto nie umie programować w żadnym języku. W pseudokodzie reprezentowane są wszystkie elementy składowe algorytmów, np. operacje wejścia/wyjścia, przypisania i struktury sterujące, takie jak konstrukcje warunkowe i pętle programowe
- W ogólności pseudo kod to logiczne połączenie elementów konkretnego języka naturalnego z elementami języka programowania. W przeciwieństwie do języka programowania pseudokod nie ma standardu. Mimo to, pisząc pseudokod należy się stosować do ogólnie przyjętych konwencji i zaleceń dotyczących tej formy zapisu algorytmu.

Zasady zapisu algorytmów w postaci pseudokodu

- Pseudokod powinien być zrozumiały zarówno dla doświadczonych programistów jaki początkujących. Dlatego też zaleca się, aby w pseudokodzie wykorzystywać jedynie powszechnie znane konstrukcje językowe (np. struktury warunkowe ***if, if-else***, pętle programowe ***while, do-while***), występujące w większości popularnych języków programowania.
- Algorytm zapisany w pseudokodzie powinien być czytelny. Dlatego w celu uzyskania jak największej przejrzystości można stosować „wcięcia" – podobnie jak wcina się w kodzie źródłowym programu instrukcje składowe konstrukcji warunkowych oraz pętli programowych. Takie podejście pomaga programiście zrozumieć wykorzystane mechanizmy sterujące.
- Łatwość zrozumienia i analizy algorytmu zapisanego w pseudokodzie można w niektórych przypadkach zwiększyć przez zastosowanie numeracji linii.

Zasady zapisu algorytmów w postaci pseudokodu

- W całym pseudokodzie należy stosować te same konwencje nazewnicze dotyczące nazw stałych, zmiennych, podprogramów, parametrów/argumentów podprogramów itp.
- Pseudokod powinien być kompletny. Innymi słowy, żadna wykorzystywana konstrukcja nie powinna być dla programisty zagadką przez zbyt dużą ogólność zapisu. Wszystkie operacje powinny być opisane na tyle szczegółowo, ażeby programista mógł je zinterpretować w sposób jednoznaczny.
- Jeśli to potrzebne, poszczególne lub wybrane operacje w algorytmie powinny być skomentowane. W tym celu można wykorzystywać styl pisania komentarzy obowiązujący w popularnych językach programowania
- Zaleca się, aby algorytmy programów rozpoczynać od podania nazwy programu oraz jego parametrów. Jeżeli podprogram zwraca na zewnątrz jakąś wartość, wówczas można go zakończyć słowem return wraz z określeniem wyrażenia, którego wartość jest zwracana.
- Dobrym nawykiem jest na początku każdego algorytmu opisać problem, który należ rozwiązać za jego pomocą.

Przykłady pseudokodu

```
//Obliczanie pola i obwodu koła dla długości promienia wprowadzonego z klawiatury
// Wyniki (pole i obwód koła) są prezentowane na ekranie monitora
Wprowadź z klawiatury długość promienia koła r
If r > 0
    then
        //obliczanie pola koła
        pole ← PI x r x r
        //Oblicznie obwodu koła
        obwod ← 2 x PI x r
        // wyświeltlenie wyników na ekranie monitora
        wyświetl pole
        wyświetl obwod
    else
        wyświetl komunikat o błędzie : długość promienia nie jest dodatnia!
```