Temat 1

Podstawowe pojęcia.

Codziennie, na każdym kroku człowiek musi rozwiązywać rozmaite problemy. Problemem jest tutaj każda, nawet najprostsza czynność – na przykład obejrzenie filmu, otwarcie okna, czy zaparzenie herbaty. Czynności te dla nikogo nie stanowią problemu, jednak spróbuj sobie wyobrazić, że musisz zbudować robota, który rozumie tylko kilka podstawowych czynności np. krok w przód, obróć, podnieś itd. Musisz mu podać listę zadań, aby otworzył okno. Nie jest to już takie proste, zwłaszcza, że Twój robot powinien umieć otwierać każde okno jednoskrzydłowe, dwuskrzydłowe, małe duże, z różnymi klamkami Taką listę czynności, którą należy wykonać aby osiągnąć określony cel nazywamy algorytmem.

***Algorytm** to przepis rozwiązania zadania, zawierający opis danych wraz z opisem czynności, które należy w określonym porządku wykonać, aby osiągnąć określony cel.*

Bardzo dobrym przykładem algorytmu w życiu codziennym jest **przepis kulinarny**. Przykładowo przepis na placek zawiera wszystkie elementy klasycznego algorytmu: są dane początkowe, czyli składniki i jest opis czynności jakie należy na tych składnikach wykonać , aby uzyskać pożądany efekt. Jest również podana kolejność, której należy bezwzględnie przestrzegać. Jest i opis wyników – czyli np. ciasto z kremem czekoladowym.

Nie każde czynności są algorytmem. **Algorytm musi posiadać następujące cechy:**

- **poprawność** – dla każdego popularnego zestawu danych, po wykonaniu skończonej liczby czynności, prowadzi do poprawnych wyników;
- **jednoznaczność** – w każdym przypadku jego zastosowania, dla tych samych danych uzyskamy ten sam wynik
- **szczegółowość** – aby wykonawca algorytmu rozumiał opisane czynności i potrafił je wykonać
- **uniwersalność** – aby służył do rozwiązywania pewnej grupy zadań, a nie tylko jednego zadania: np. algorytm jest przepisem na rozwiązanie równania postaci *ax+b=0* dla dowolnych współczynników *a* i *b*, a nie – jednego konkretnego równania, np*. 2x+3=0*

W rozwiązywaniu problemów należy zwracać uwagę na szereg po sobie występujących **akcji**, które realizują w skończonym czasie pewien **proces** lub obliczenie. Każda z akcji wymaga istnienia **obiektów**, do których się odnosi. Efektem wystąpienia akcji jest osiągnięcie **wyniku.**

Opis akcji nazywamy działaniem, **instrukcją** albo **operacją**. Natomiast ich zbiór jest algorytmem. **Algorytm** jest gotowym rozwiązaniem pewnego problemu. Rozwiązanie to można zapisać za pomocą jakiegoś języka programowania, tak aby dany problem mógł być rozwiązany przez komputer. Algorytm zapisany w języku programowania nazywamy programem. Dane są to pewne informacje, które należy dostarczyć programowi na pewnym etapie jego działania – na przykład podać jakieś liczby, które program przetworzy.

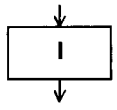
Różnica między ogólnym algorytmem a programem komputerowym jest taka, że program jest algorytmem przedstawionym ściśle według pewnych reguł językowych (języka programowania). **Językiem programowania** jest jakiś zbiór zdań zbudowanych ze słów, które są ciągami liter i cyfr oraz znaków specjalnych. Każdy język ma swoja składnię i budowę. Słowa kluczowe takich języków są zbliżone do języka naturalnego, najczęściej jest to kombinacja słów i skrótów z języka angielskiego. Języki takie nazywamy **językami wysokiego poziomu**. Program zapisany w takim języku nie jest jeszcze takim programem, który komputer mógłby wykonać, ponieważ rozumie on tylko specjalne sekwencje rozkazowe, które w gruncie rzeczy są zestawem zer i jedynek. Aby było możliwe wykonanie takiego programu musi on zostać **skompilowany**, czyli przetłumaczony na postać binarną, zrozumiałą dla komputera. Do najpopularniejszych języków programowania wysokiego poziomu należą: Turbo Pascal, Basic, C++, Algol, Fortran, Cobol, Delphi i inne.

Druga grupę stanowią języki maszynowe tzw. **asemblery**, które przeznaczone są dla informatyków zajmujących się podstawowym oprogramowaniem komputera (tworzeniem systemów operacyjnych). Każdy mikroprocesor posiada swój język maszynowy. Języki te nazywa się tez często **językami niskiego poziomu**. Są one bardzo trudne, gdyż ich składnia bliższa jest samemu komputerowi niż człowiekowi, który myśli za pomocą pojęć języka naturalnego, a nie sekwencji zer i jedynek. Inna grupa języków programowania to **języki zorientowane problemowo** – przeznaczone dla specjalistów z jakiś dziedziny, na przykład konstruktorów mostów, układów scalonych, systemów zarządzania bazami danych itd.

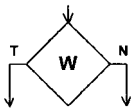
Temat 2

Schematy blokowe jako metoda prezentacji algorytmów.

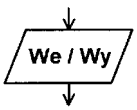
IIstnieje kilka sposobów prezentacji algorytmów. Algorytmy możemy przedstawiać za pomocą języka naturalnego – w postaci listy kolejnych kroków wraz z opisem czynności, za pomocą umownego języka strukturalnego albo za pomocą schematów blokowych. Ta ostatnia metoda jest najpowszechniej używana. **Schematy blokowe są graficzną formą prezentacji algorytmu, obrazującą przebieg jego działania, ułatwiającą zapis algorytmu w języku programowania. Służą również do zapisu modułów algorytmicznych, czyli fragmentów programu zwanych procedurami**. Schematy konstruuje się ze specjalnych, ściśle określonych elementów graficznych, zwanych skrzynkami oraz połączeń między nimi wyznaczających przebieg algorytmu.



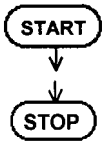
Skrzynka operacyjna – zwana też instrukcyjną, jest prostokątem, w którym znajdują się instrukcje. Ze skrzynki operacyjnej wychodzi tylko jedno połączenie. W takiej skrzynce dokonujemy przypisania wartości zmiennym. Do tego celu używa się operatora „**: =**" Przykładowo: „a : = b+2" oznacza: zmiennej „a" przypisz (albo weź podstaw) wartość zmiennej „b + 2".



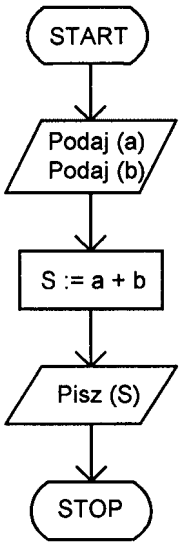
Skrzynka warunkowa, która wyznacza kierunek przebiegu algorytmu w zależności od spełnienia, bądź nie spełnienia warunku zapisanego wewnątrz rombu. Jeżeli warunek jest spełniony, algorytm przebiega w kierunku zgodnym ze strzałką oznaczoną litera „T" (TAK), a jeżeli nie, to w kierunku wyznaczonym przez strzałkę „N" (NIE).



Skrzynka wejścia / wyjścia. Ma kształt równoległoboku. Znajdują się w niej instrukcje pobierania danych do algorytmu lub wyprowadzania wyniku. Przykładowo: „podaj a" – oznacza pobranie do programu wartości, która będzie przypisana zmiennej „a"; „pisz wynik" – oznacza instrukcję wyprowadzająca z programu wartość zmiennej „wynik".



Skrzynka początku i końca – wyznaczają początek i koniec algorytmu. W całym algorytmie może znajdować się tylko jedna skrzynka START, która ma tylko jedno wyjście oraz tylko jedna skrzynka STOP, która ma tylko jedno wejście.



Przykład 1

Skonstruuj algorytm obliczania sumy dwóch liczb.

Rozwiązanie:

Rozwiązanie w postaci schematu blokowego pokazuje przebieg algorytmu. Pobierane są wartości dwóch liczb i zapisywane odpowiednio w zmiennej „a" oraz „b". W konstrukcji algorytmu została też użyta zmienna „s" oznaczająca sumę. W skrzynce instrukcyjnej nastąpiło przypisanie wartości zmiennej „s" sumy wartości zmiennych „a" i „b". Następnie w skrzynce wejścia / wyjścia został wyprowadzony wynik i algorytm kończy swój przebieg.

Przykład ten jest bardzo prostym przykładem. Nie została tutaj użyta skrzynka warunkowa. Spróbuj przekształcić ten algorytm tak, aby obliczał iloraz liczb a i b. Pamiętaj, iż dzielnikiem nie może być zero! Wobec tego powinieneś użyć skrzynki warunkowej, tak aby działanie mogło się odbyć na prawidłowych wartościach.

Temat 3-4

Schematy blokowe – ćwiczenia w konstrukcji algorytmów.

Przykład 1

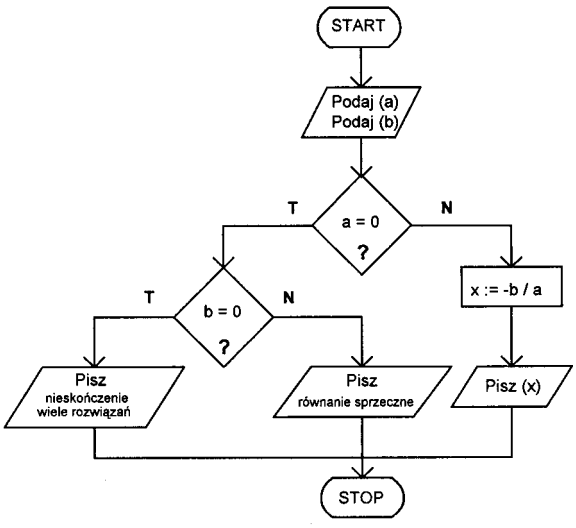
Spróbujmy skonstruować algorytm wyznaczania pierwiastków równania liniowego ax+b=0. Aby rozwiązać to równanie, należy wziąć pod uwagę następujące przypadki:

- a=0, b=0 – wtedy równanie posiada nieskończenie wiele rozwiązań
- a=0; b≠0 – wtedy równanie nie posiada rozwiązania – mówimy, że jest sprzeczne
- a≠0 – wtedy równanie posiada jedno rozwiązanie x= -b/a

Spróbujmy teraz ułożyć algorytm rozwiązania posługując się językiem naturalnym:

1. Podaj wartości współczynników a i b
2. Sprawdź, czy a=0.
   - Jeżeli a=0, to sprawdź, czy b=0
   - jeżeli a=0, b=0, to napisz, że x jest dowolne, czyli nieskończenie wiele rozwiązań
   - jeżeli a=0, b≠0 to napisz, że równanie jest sprzeczne i brak jest odpowiedzi
3. Jeżeli a ≠ 0, to wykonaj obliczenie: x= -b/a

A oto graficzne przedstawienie powyższego algorytmu:



Przykład 2

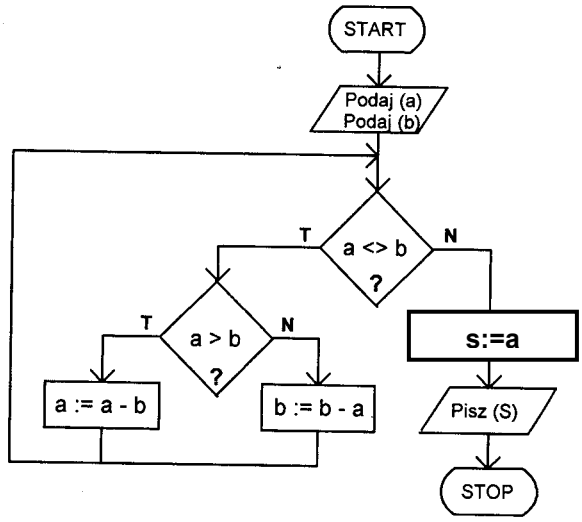
Korzystając z algorytmu Euklidesa wyznacz największy wspólny dzielnik (NWD) dwóch liczb naturalnych.

Analiza problemu:

Algorytm Euklides polega na sprawdzeniu, czy dwie liczby naturalne są sobie równe. Jeżeli tak, to za wartość NWD przyjmuje się wartość jednej z tych liczb, jeżeli nie, to dalej algorytm polega na odejmowaniu większej liczby od mniejszej i przypisaniu wyniku zmiennej reprezentowanej przez odjemną. Czynność tę wykonuje się tak długo, dopóki nie otrzyma się dwóch liczb sobie równych. Wtedy jest to wspólny dzielnik. Przykładowo a=25, b=15. Wtedy algorytm Euklidesa będzie miał następujący przebieg:

1. a>b → a:25-15=10
2. a<b → b:=15-10 = 5
3. a>b → a:= 10-5 =5
4. a=b → S :=5

A oto graficzna prezentacja tego algorytmu:



zadania:

1. Przygotuj algorytm sprawdzania parzystości podanej liczby. Algorytm zapisz za pomocą schematu blokowego
2. Dane są trzy liczby a, b c. Przygotuj algorytm wyznaczania największej z nich. Zapisz go za pomocą schematu blokowego.
3. Skonstruuj algorytm obliczający średnią arytmetyczną sumy 10 kolejnych liczb parzystych (od 2 do 20). Algorytm zapisz za pomocą schematu blokowego
4. Dane są trzy odcinki a, b, c. Sprawdź, czy można z nich zbudować trójkąt. Algorytm zapisz za pomocą schematu blokowego
5. Skonstruuj algorytm obliczającego sumę „n" kolejnych liczb naturalnych począwszy od liczby „a".
6. Zbuduj algorytm prostej gry – zgadywanki, która polega na tym, ze komputer losuje jakąś liczbę, a gracz zgaduje jaka to liczba. Następnie komputer daje odpowiedz „za dużo" albo „za mało", a gdy liczba zostanie odgadnięta, to daje komunikat „Brawo! odgadłeś!" Niech komputer sprawdza tez ilość prób, jakie gracz wykonał, żeby odgadnąć liczbę.