Wydział Zarządzania AGH
Katedra Informatyki Stosowanej

Programowanie komputerowe

Program przedmiotu

- **Wprowadzenie do programowania**
- **Język VBA:**
  - **stałe i zmienne – deklarowanie zmiennych**
  - **instrukcja przypisania, wyrażenia arytmetyczne, instrukcja komentarza**
  - **podstawowe instrukcje wejścia – wyjścia**
  - **instrukcja warunkowa, instrukcja wyboru**
  - **instrukcje iteracyjne**
  - **funkcje i procedury, rekurencja**

www.kis.zarz.agh.edu.pl

Program przedmiotu

Literatura:

- Carmen T. H., Leiserson C. E., Riverst R. L.: *Wprowadzenie do algorytmów*, WNT, Warszawa 2002.
- Jacobson R.: *Microsoft Excel 2002. Visual Basic for Applications krok po kroku*, Oficyna Wydawnicza READ ME, Warszawa 2003.
- Osyczka A., Jankowski R., Skalna I., Krajewski P.: *Visual Basic dla Aplikacji*, Uczelniane Wydawnictwa Naukowo-Dydaktyczne AGH, Kraków 2006.
- Orwis W. J.: *Visual Basic dla aplikacji w przykładach. Sztuka programowania*, Oficyna Wydawnicza READ ME, Warszawa 1995.
- Wróblewski P.: *Algorytmy, struktury danych i techniki programowania*, Helion, Gliwice 2001.
- VB Mania - witryna poświęcona Visual Basic dla aplikacji.

Program wykładu

- Wprowadzenie
- Algorytmy
- Języki programowania
- Tworzenie programów
- Schematy

Wprowadzenie

Dlaczego warto umieć programować

- Drzwi do kariery nie otwiera już automatycznie żaden dyplom. Kiedy Gabriela Jabłońska kończyła UJ, bezrobotnych socjologów było już na pęczki. – *Dla potencjalnych pracodawców są za bardzo humanistyczni* – uważa Gabriela. Żeby więc poprawić swoje szanse na zdobycie pracy, musiała się douczyć tego, czego w programie socjologii jest za mało: statystyki i obsługi programów komputerowych. W zdobycie tych umiejętności trzeba zainwestować, ale bez nich trudno dziś marzyć o jakiejkolwiek pracy. – *W Stanach Zjednoczonych ludzie zaczynają sobie zdawać sprawę, że umiejętność programowania staje się równie ważna jak obsługa komputera* – twierdzi Jabłońska. Przydatna jest w wielu zawodach. Burmistrz Nowego Jorku, Michael Bloomberg, ogłosił, że w 2012 r. ma zamiar się tego nauczyć. Udało się?

Źródło: Joanna Solska, *Tajemniczy zawód*, Polityka, nr 7, 2012.

Wprowadzenie

Dlaczego warto umieć programować

- Programowanie to umiejętność, która w dzisiejszym świecie jest niezwykle przydatna i może bardzo ułatwić nam życie. Nawet gdy nie zamierzamy pracować na stanowisku programisty, umiejętność samodzielnego napisania kilku programów na własny użytek może być wielkim atutem. Coraz większa popularyzacja telefonów komórkowych, na które można pisać własne aplikacje, możliwość stworzenia programu dostępnego z poziomu przeglądarki internetowej czy zwykłe aplikacje desktopowe, w których zawrzemy funkcje potrzebne nam i dostosowane do naszych potrzeb, mogą być nieocenioną pomocą w codziennym życiu i pracy.
- Nie trzeba być specem od programowania, by zbić na tym fortunę: jeśli tylko jesteśmy w stanie wpaść na ciekawy program, aplikację na popularnego iPhona czy inny telefon możemy z powodzeniem sprzedawać za pośrednictwem udostępnianej przez producenta telefonu platformy właśnie dla takich projektów. Sprzedanie jednego programu nie da nam dużych pieniędzy, jeśli jednak uda się zdobyć na niego kilkadziesiąt czy nawet kilkaset tysięcy chętnych?

Źródło: http://programy.einfo.waw.pl/dlaczego-warto-umiec-programowac

Wprowadzenie

Dlaczego warto umieć programować

- MP-ror [Generał] Programowanie - co warto umieć
  Jaki język warto szlifować ? Który język programowania ma przyszłość ? C++, C# a może jakiś zupełnie inny? Co jest obecnie na topie i co rokuje nadzieje na pracę w przyszłości ? Znacie się na tym ?
- moneo [Nikon Sevast]
  MP-ror---> zalezy, co chcialbys programowac.
  jesli oprogramowanie biznesowe, to na pewno ucz sie C# i Javy - zdania sa podzielone, co bedzie bardziej popularne w przyszlosci.
- cotton_eye_joe [maniaq]
  teraz najwieksza kase chyba mozna skukac jak umiesz VBA i C#. C++ jest chyba bardziej wykorzystywany do pisania np. gier. natomiast z tego co widze w ogloszeniach o prace przewaza VBA i c#. i PHP ale to bardziej podchodzi pod bazo-danowcow razem z sql. java - do aplikacji w telefonach.
- ExistoR [Sigma Private Hitman]
  Nic kompletnie nie wiem o programowaniu ale w przyszłości zamierzam w tym na poważnie pogmerac. Tylko powiedzcie mi czy nauka takiego np. C++ wymaga na prawdę dużo czasu? Wiem, że nie będzie to jeden wieczór ani 2, 3 czy nawet 5 ale czy po prostu musze nad tym siedziec ponad rok czy 2 żeby w miare dobrze go opanowac?

Źródło: http://forumarchiwum.gry-online.pl/S043archiwum.asp?ID=4476135

Wprowadzenie

Dlaczego warto umieć programować

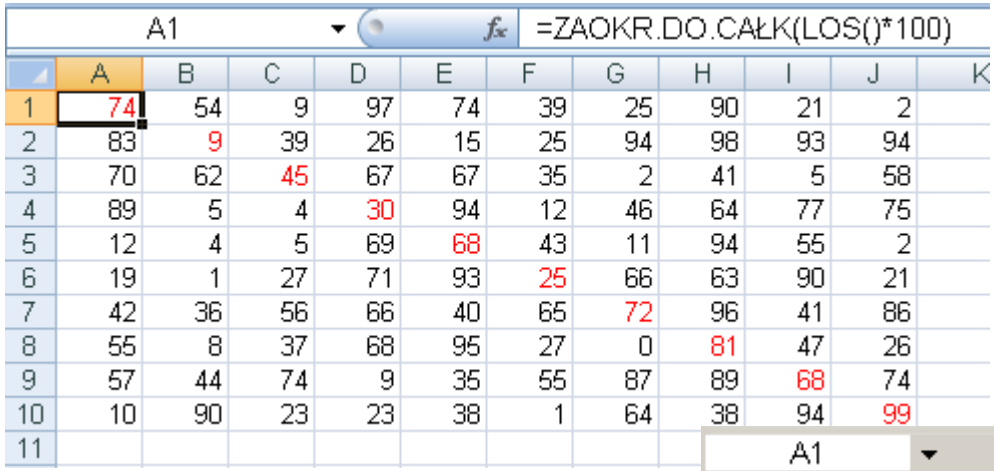
- Na naszych oczach zmienia się sposób myślenia, sposób pracy, nawet mózgi kolejnych pokoleń przedstawicieli naszego gatunku mają strukturę połączeń nerwowych nieco inną niż rodzice, aby lepiej radzić sobie z wszechobecną elektroniką. Kto spróbował kiedyś programować, ten wie, że czynność ta wymaga specyficznego podejścia do problemów i myślenia abstrakcyjnego w oderwaniu od języka naturalnego, przypominającego nieco to znane matematykom. Na temat przydatności takich umiejętności w życiu codziennym można się spierać — w końcu wykonywanie operacji na macierzach nie jest większości z nas do szczęścia potrzebne — ale zmieniają one na zawsze sposób rozumienia gadżetów i postrzegania otoczenia, pozwalają rozłożyć na mniejsze, analizować i optymalizować nawet zwykłe, domowe problemy. Poza tym programowanie „wycieka” do innych dziedzin nauki — do fizyki, matematyki, biologii, chemii czy socjologii.
- Za kilka lat i w Polsce pisanie kodu może stanąć w szkołach na równi z matematyką i językiem. Na razie wśród Polaków w wieku 16-24 lata program napisało 16% — jesteśmy pod tym względem na 21 pozycji w Unii Europejskiej.

Źródło: www.dobreprogramy.pl/Pisanie-kodu-zamiast-nauki-Officea-w-szkole-podstawowej,Aktualnosc,38278.html

Wprowadzenie

Dlaczego warto umieć programować

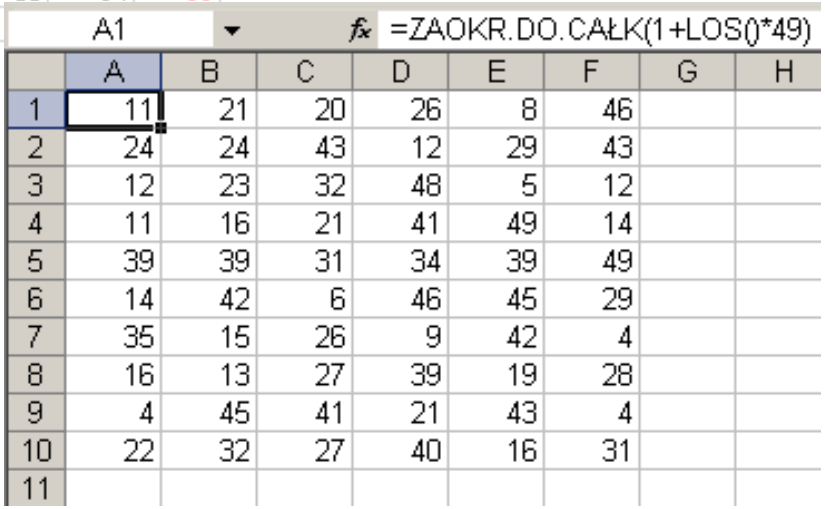
- Zsumować wszystkie wartości na przekątnej tabeli.



A1 =ZAOKR.DO.CAŁK(LOS()*100)

	A	B	C	D	E	F	G	H	I	J	K
1	74	54	9	97	74	39	25	90	21	2	
2	83	9	39	26	15	25	94	98	93	94	
3	70	62	45	67	67	35	2	41	5	58	
4	89	5	4	30	94	12	46	64	77	75	
5	12	4	5	69	68	43	11	94	55	2	
6	19	1	27	71	93	25	66	63	90	21	
7	42	36	56	66	40	65	72	96	41	86	
8	55	8	37	68	95	27	0	81	47	26	
9	57	44	74	9	35	55	87	89	68	74	
10	10	90	23	23	38	1	64	38	94	99	
11											

- Wylosować *n* zakładów Lotto.



A1 =ZAOKR.DO.CAŁK(1+LOS()*49)

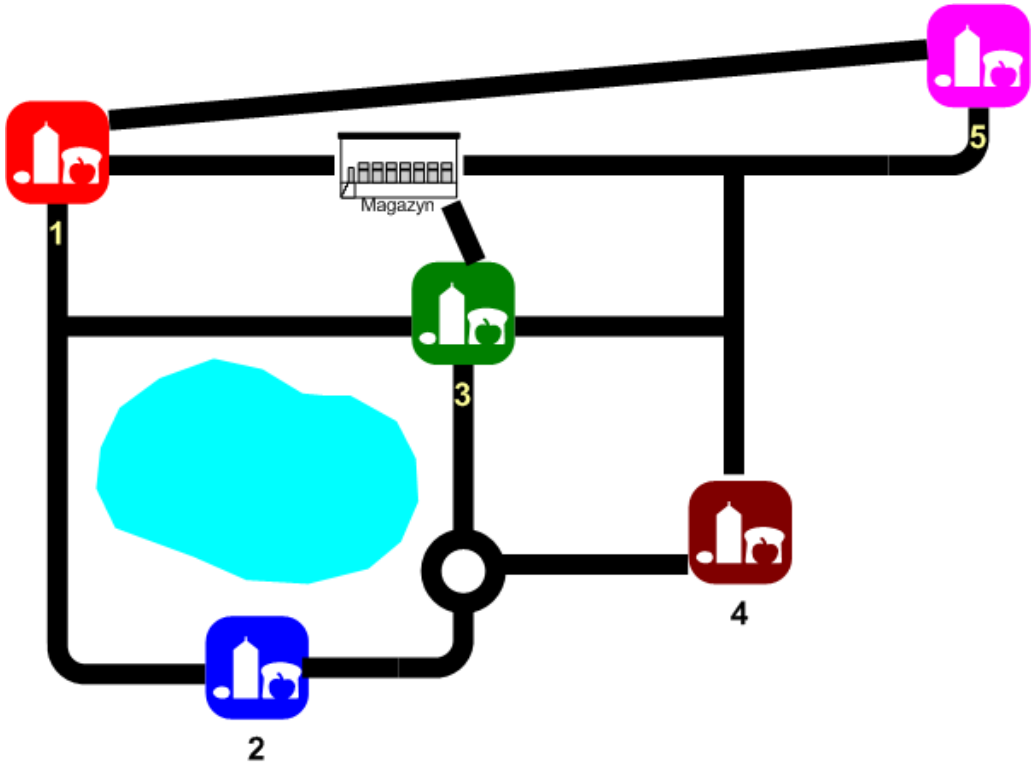
	A	B	C	D	E	F	G	H
1	11	21	20	26	8	46		
2	24	24	43	12	29	43		
3	12	23	32	48	5	12		
4	11	16	21	41	49	14		
5	39	39	31	34	39	49		
6	14	42	6	46	45	29		
7	35	15	26	9	42	4		
8	16	13	27	39	19	28		
9	4	45	41	21	43	4		
10	22	32	27	40	16	31		
11								

Źródło: życie.

Wprowadzenie

Dlaczego warto umieć programować

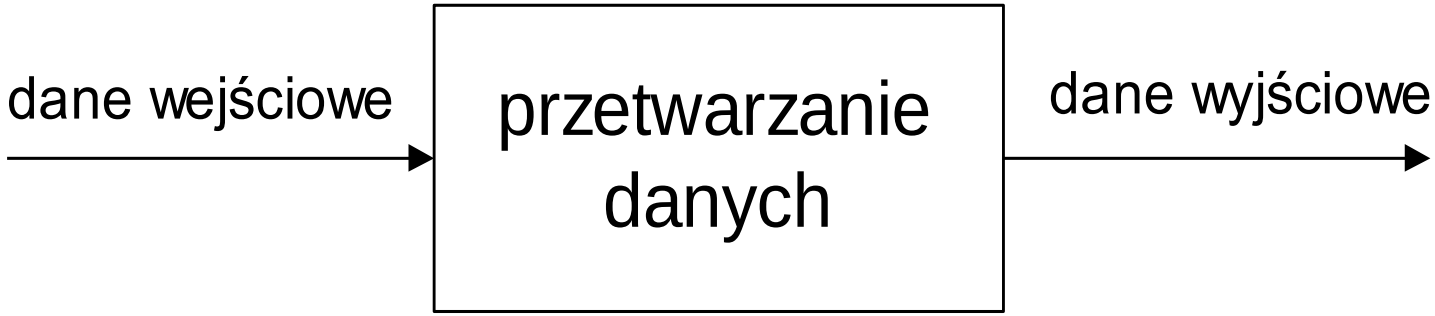
- Zaproponować trasę samochodu dostawczego z pieczywem. Kierowca ma rozwieźć ustalone ilości do 15 punktów tak, by przejechać najkrótszą trasą, odwiedzić tylko jeden raz po drodze wszystkie sklepy, i wrócić do punktu startowego.



Źródło: życie.

Wprowadzenie

- We wszystkich systemach przetwarzania danych występują zawsze 3 elementy:



- Dokładną specyfikację działania w systemie przetwarzania danych uzyskuje się przez podanie algorytmu tych działań.

Algorytmy

- **Algorytm jest to zbiór reguł rozwiązania problemu w skończonej liczbie kroków.**
- Nie dla wszystkich zagadnień można określić algorytm rozwiązujący.
- Jednym ze skutków powszechnego stosowania informatyki jest rozpowszechnienie stosowania algorytmów w innych dziedzinach jako bardzo wygodnej metody prezentacji.

Języki programowania

- Algorytmy przedstawia się w postaci skończonego ciągu symboli w pewnym języku.
- **Język** to skończony zbiór symboli (alfabet) wraz ze zbiorem reguł łączenia poszczególnych symboli w ciągi (składnia) i zbiorem reguł nadających formalnym ciągom symboli znaczenie merytoryczne (semantyka).
- Inaczej: język to ogólna nazwa zdefiniowanego zbioru znaków oraz reguł określających sposoby i kolejność, w jakich znaki te mogą być łączone dla nadania im znaczenia w tym języku.
  Przykład: instrukcja przypisania (zmienna = wyrażenie)

$$x = x*y/4 + \text{potega}(z,3)$$

Języki programowania

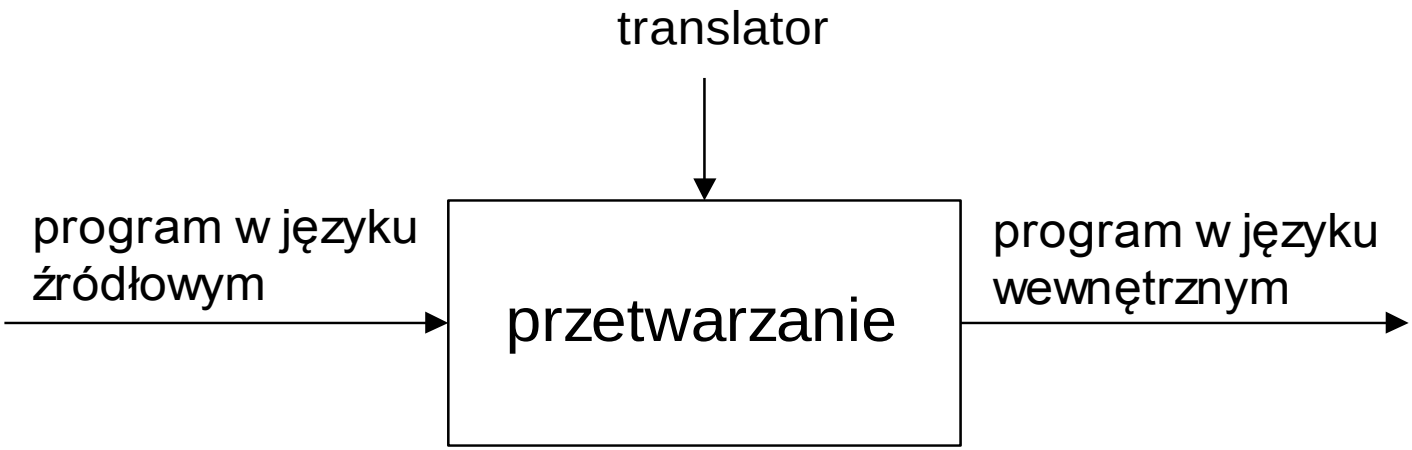
- Język stosowany do celów programowania nazywamy **językiem programowania**.
- Algorytm przedstawiony w języku programowania nazywa się **programem**.
- Szczególnym językiem programowania jest **język wewnętrzny** maszyny (język maszynowy). Elementami tego języka są rozkazy maszyny. Wszystkie rozkazy tworzą listę rozkazów maszyny, czyli opis wszystkich działań, które mogą być wykonane przez komputer.
- Obiektami działania rozkazów są dane, identyfikowane przez podanie adresu określającego miejsce ich przechowywania w pamięci.

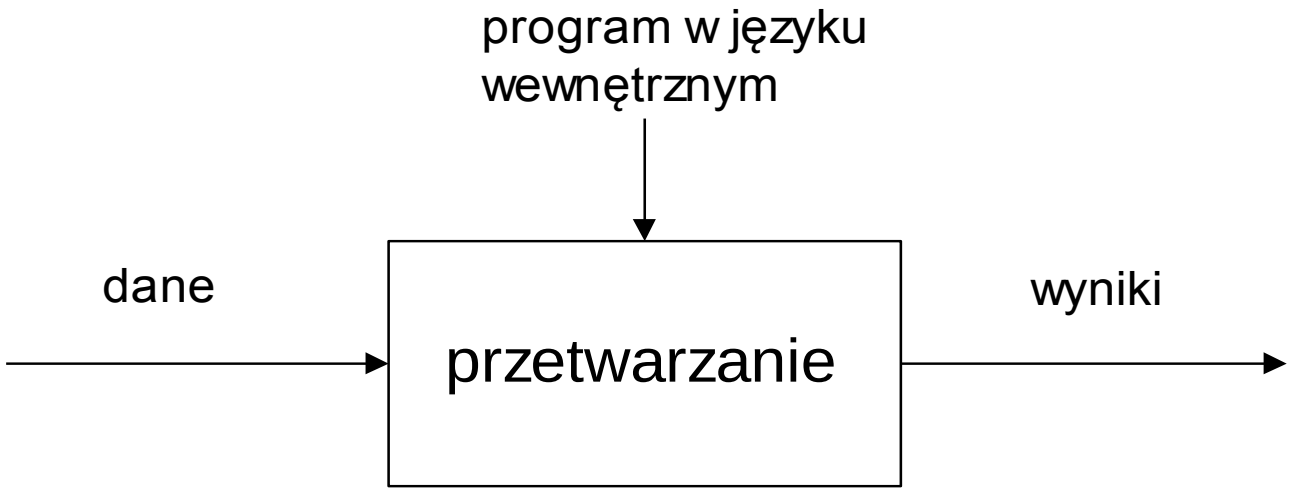
Języki programowania

- Wykonując rozkazy maszyna realizuje program w języku wewnętrznym (tylko taki program komputer może wykonać).
- W celu zrealizowania programu napisanego w języku różnym od języka maszyny, należy przekształcić go (przetłumaczyć) na równoważny program (przedstawiający ten sam algorytm) w języku maszynowym.

  Tłumaczenie takie nazywane jest **translacją**.

Języki programowania





Języki programowania

- Istnieje wiele języków programowania, które mają różne cechy:
  - przeznaczenie,
  - łatwość nauczenia się,
  - zwięzłość,
  - czytelność (łatwo zrozumiałe instrukcje),
  - łatwość pisania i zmiany programów,
  - szybkość procesu translacji,
  - szybkość wykonywania przetłumaczonych programów.

Języki programowania

- Najprostszym, i historycznie pierwszym, językiem programowania komputerów był **język bezpośredni**, w którym ciągom symboli języka zewnętrznego odpowiadały (w jednoznacznej relacji jeden-do-jednego) poszczególne rozkazy maszyny. Programista musiał nie tylko wymyślić algorytm, ale również zaplanować rozmieszczenie programu i danych.

Języki programowania

- W **języku symbolicznym** ciągi symboli odpowiadają rozkazom maszynowym również w jednoznacznej relacji 1-1, natomiast części adresowe poszczególnych rozkazów nie są podawane w postaci bezpośredniej (liczbowej), lecz w postaci symbolicznych nazw złożonych z kilku symboli języka.
  Przyporządkowanie poszczególnym nazwom rzeczywistych adresów jest dokonywane przez odpowiedni program umieszczony uprzednio w pamięci (tzw. program adresujący).
- Wykorzystywanie języka symbolicznego znacznie skróciło czas uruchamiania programu tzn. czas przeznaczony na wykrywanie i usuwanie pomyłek występujących w programie już wprowadzonym do komputera.

Języki programowania

- Dalszy rozwój języków programowania wynikał z kilku czynników:
  - zaczęto budować coraz szybsze komputery, których efektywne wykorzystanie wymagało dostarczenia im coraz większej ilości informacji, stąd przyśpieszenie procesu programowania stało się nagląca koniecznością,
  - większe możliwości komputerów pozwały przerzucić bezpośrednio na nie więcej pracy związanej z programowaniem,
  - nacisk coraz szerszego kręgu ludzi, którzy chcieli korzystać z komputerów, lecz programowanie w językach symbolicznych wymagało od nich opanowania wielu szczegółowych wiadomości całkowicie zbędnych z ich zawodowego punktu widzenia (np. konieczna była znajomość listy rozkazów),
  - programy napisane w języku symbolicznym nie mogły być przenoszone między różnymi maszynami.

Języki programowania

- Pojawił się nowy typ języków programowania - **autokody**, w których:
  - nie obowiązuje zasada odpowiedniości typu 1-1 między ciągami symbolów języka a rozkazami maszynowymi,
  - wprowadzono zapis wyrażeń arytmetycznych zbliżony do stosowanego w tradycyjnej notacji matematycznej,
  - ułatwiono korzystanie z gotowych podprogramów.
- Zmiany:
  - konieczność stosowania bardziej złożonych programów tłumaczących,
  - łatwość opanowania zasad programowania,
  - znaczne skrócenie czasu układania programu,
  - mniej okazji do popełniania pomyłek.

Tworzenie programów

- **Programowanie** jest sztuką(?) pisania programów przy użyciu jakiegoś języka programowania.
- Podstawową sprawą przy tworzeniu programów jest właściwe sformułowanie problemu. Chodzi tu zarówno o rozwiązanie ostateczne jak i cele pośrednie prowadzące do tego rozwiązania.
- Przystępując do formułowania problemu programista musi być zorientowany w ilości i rodzaju danych wejściowych i wynikowych.

Tworzenie programów

Dalsze etapy przygotowania programu:

- wybór metody rozwiązania,
- redakcja programu,
- kodowanie,
- uruchamianie i testowanie.

Po pomyślnym zakończeniu tych wszystkich etapów program jest gotowy do wykorzystania: po wczytaniu go do pamięci komputera i dostarczeniu odpowiednich danych można oczekiwać, że uzyskamy poprawne rezultaty w procesie obliczeniowym.

Wybór metody rozwiązania

- Z reguły istnieje więcej niż jedna metoda rozwiązania problemu. Zazwyczaj stosuje się metody standardowe (znane np. z literatury), jednak niejednokrotnie należy opracować własne rozwiązanie uwzględniające w miarę konieczności specyfikę rozwiązania maszynowego (np. wykorzystanie dużej szybkości obliczeniowej komputera, co preferuje metody iteracyjne).

Wybór metody rozwiązania

Złożoność rzeczywistych zadań:

1. Dane wejściowe – sytuacja na szachownicy.
2. Wynik – najlepszy ruch białych.
3. Algorytm?

1. Dane wejściowe – rozwieźć towar do 50 punktów.
2. Wynik – trasa o minimalnej długości.
3. Algorytm?

Redakcja programu

- Poza najprostszymi problemami, w etapie tym konieczne jest opracowanie **schematu działania algorytmu**.
- Schemat jest przedstawieniem ciągu operacji wykonywanych podczas realizacji programu z uwzględnieniem możliwych (w zależności od uzyskanych wyników cząstkowych) kolejności ich wykonywania.
- Schemat może być przedstawiony w postaci:
  - graficznej (schemat blokowy, flowdiagram),
  - pseudokodów,
  - opisu słownego.
- Ostatecznym rezultatem procesu redagowania programu jest ciąg instrukcji, który w zamierzeniu programisty stanowi zapis algorytmu rozwiązywanego problemu.

Kodowanie

- Kodowanie polega na przeniesieniu rozkazów napisanego uprzednio programu do pliku, który może być odczytany bezpośrednio przez maszynę.
- Niedokładne kodowanie może być przyczyną licznych błędów.

Uruchamianie i testowanie programu

- **Uruchamianie** programu ma za zadanie wykrycie wszystkich błędów i pomyłek powstałych przy redagowaniu i kodowaniu programu.
- Uruchomienie dowodzi jedynie usunięcia podstawowych błędów (przede wszystkim składniowych), nie musi jednak świadczyć o tym, że program działa zgodnie z założeniami i że uzyskane wyniki w rzeczywistości reprezentują wyniki, dla których program został napisany.

Uruchamianie i testowanie programu

- W celu sprawdzenia, czy program nie zawiera błędów logicznych stosuje się **testowanie programu**.
- Testowanie polega zazwyczaj na przekazaniu do programu danych i porównaniu uzyskanych wyników z wynikami uzyskanymi z obliczeń poza komputerem. Obliczenia takie wykonywane są dla serii określonych danych, a ich poprawność pozwala przypuszczać, że program zachowuje się poprawnie dla wszystkich możliwych przypadków.

Schematy

- umożliwiają szybkie zorientowanie się w algorytmie,
- stanowią środek wymiany informacji między wieloma programistami opracowującymi dany program równocześnie,
- stanowią część dokumentacji programu.
- Schemat stanowi obraz działania programu, w którym dokładność przedstawienia szczegółów zależy od określonych potrzeb. Ten sam algorytm może być przedstawiony za pomocą schematu o niewielkiej liczbie elementów (wtedy każdy element obrazuje ciąg wielu rozkazów) lub w ten sposób, że każdemu elementowi schematu odpowiada dokładnie jedna instrukcja.

Schematy – sieć działań

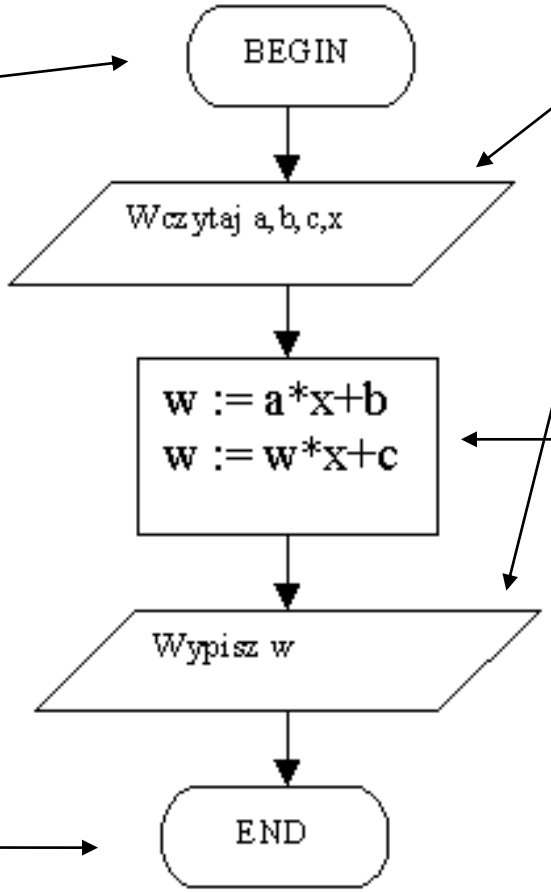
- **Flowdiagram** jest graficznym przedstawieniem algorytmu.
- Elementami sieci działań są graficzne symbole operacji i decyzji podejmowanych w programie; elementy te są łączone ze sobą za pomocą zorientowanych linii, które wskazują kolejność wykonywania poszczególnych operacji.

Schematy – sieć działań

- Sieci działań tworzone są wg następujących reguł:
  - sieć składa się ze skrzynek połączonych zorientowanymi liniami,
  - skrzynki obrazują ciąg (niekiedy jednoelementowy) operacji,
  - kolejność wykonywania operacji w programie jest określona przez zorientowane linie łączące poszczególne skrzynki; zazwyczaj skrzynki są łączone od górnej do dolnej i od lewej do prawej (wtedy strzałki mogą być opuszczone),
  - do każdej skrzynki może wchodzić co najwyżej jedna linia,
  - linie mogą się łączyć w tzw. punktach zbiegu.

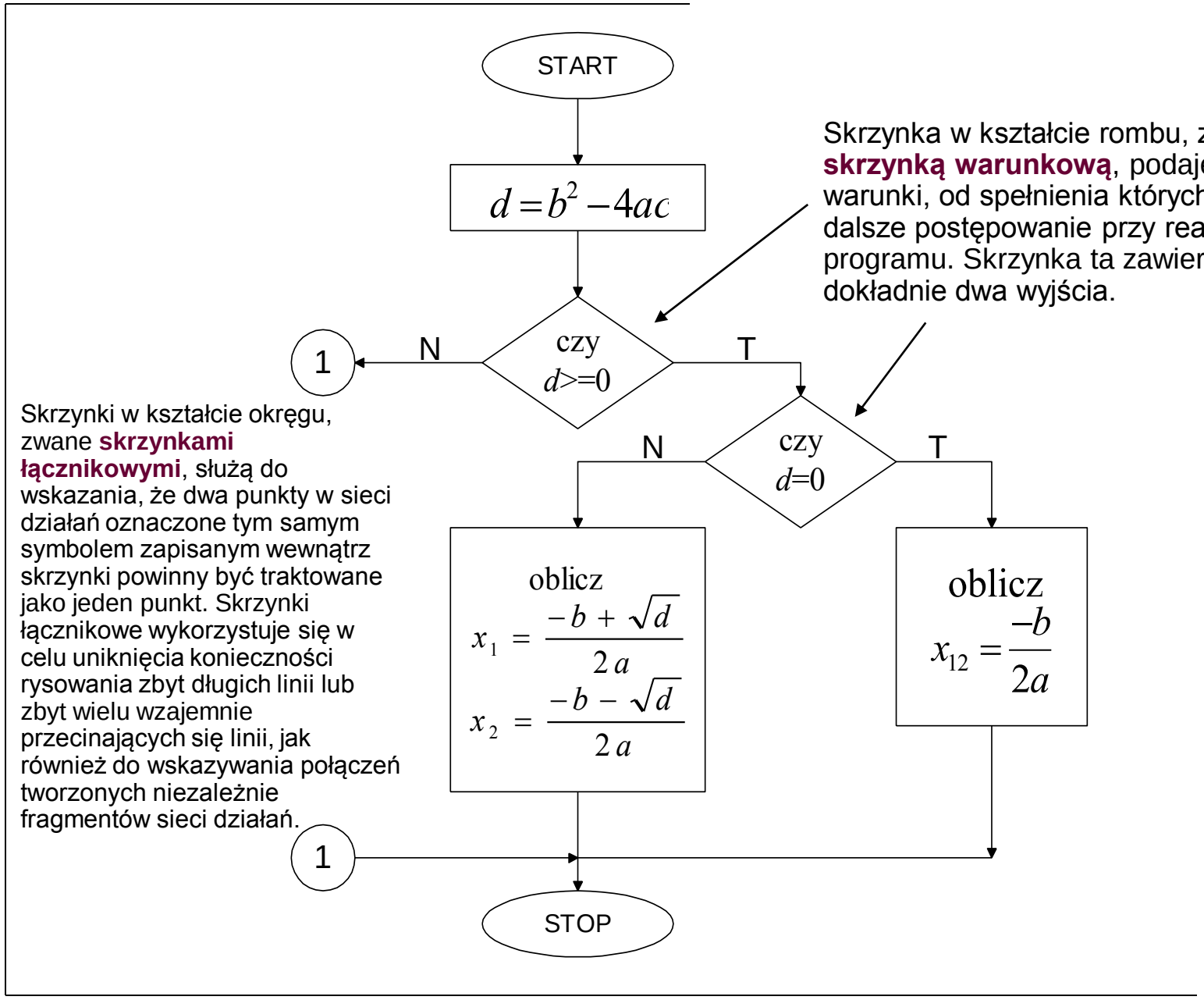
Schematy – sieć działań

Skrzynka w kształcie owalnym, zwana **skrzynką graniczną**, zawiera słowa **START** albo **STOP** i wskazuje miejsca, z których analiza sieci działań jest rozpoczynana i w których jest kończona.



Blok wprowadzania/wyprowadzania danych (równoległobok z listą wprowadzanych lub wyprowadzanych danych). Wchodzi do niego jedna linia i wychodzi jedna. Symbolizuje operacje wprowadzania lub wyprowadzania danych.

Skrzynka prostokątna (operacyjna) zawiera zapis jednej lub kilku operacji obliczeniowych albo wejścia-wyjścia. Ze skrzynki obliczeniowej wychodzi tylko jedna linia i w związku z tym przejście do następnej operacji jest z góry jednoznacznie określone. Wewnątrz skrzynki zapisywana jest operacja(cje) realizowana przez tę skrzynkę. Postać zapisu operacji jest w zasadzie dowolna.



Skrzynka w kształcie rombu, zwana **skrzynką warunkową**, podaje warunki, od spełnienia których zależy dalsze postępowanie przy realizacji programu. Skrzynka ta zawiera dokładnie dwa wyjścia.

Skrzynki w kształcie okręgu, zwane **skrzynkami łącznikowymi**, służą do wskazania, że dwa punkty w sieci działań oznaczone tym samym symbolem zapisanym wewnątrz skrzynki powinny być traktowane jako jeden punkt. Skrzynki łącznikowe wykorzystuje się w celu uniknięcia konieczności rysowania zbyt długich linii lub zbyt wielu wzajemnie przecinających się linii, jak również do wskazywania połączeń tworzonych niezależnie fragmentów sieci działań.

Algorytmy – pseudokody

- Schematy mogą być zapisywane w sposób opisowy, przy użyciu języka naturalnego. Często wykorzystuje się przy tym elementy wybranego języka programowania np. PASCAL, C (mówimy wtedy o pseudokodach języka PASCAL, C).
- Przy tworzeniu schematów w pseudokodach obowiązują podobne zasady jak w przypadku sieci działań.

Algorytmy – pseudokody

```
BEGIN
    R := R0 {rozwiązanie początkowe};
    T := T0 {temperatura początkowa > 0};
    wybierz funkcję redukcji temperatury α;
    WHILE (nie spełniony warunek zatrzymania) DO
    BEGIN
      WHILE (nie osiągnięty stan równowagi) DO
      BEGIN
        R' := f(R) {nowe, sąsiednie rozwiązanie};
        dE:=E(R') - E(R);
        IF (dE < 0) THEN R:= R'
        ELSE IF (RANDOM < exp(-dE/T)) THEN R:= R'
      END;
      T:= α(T);
    END;
END.
```

Algorytmy – opis słowny

1. Start
2. Podaj wartości *a*, *b*, *c*
3. Oblicz deltę: $d=b^2-4ac$
4. Jeśli delta jest mniejsza od zera, to idź do kroku 8, jeśli delta jest równa zero, to idź do kroku 6
5. Oblicz dwa pierwiastki: $x_1=(-b+sqrt(d))/2a$, $x_2= (-b-sqrt(d))/2a$, idź do kroku 7
6. Oblicz pierwiastek $x_{12}=-b/2a$
7. Wypisz pierwiastki
8. Stop

Kodowanie, uruchamianie i testowanie

- Istnieją dwa rodzaje programów tłumaczących:
  - kompilatory,
  - interpretatory.
- Interpretator (interpreter) wczytuje fragment kodu źródłowego programu, tłumaczy na kod wynikowy i od razu wykonuje. Operacja ta jest powtarzana dla kolejnych elementów programu. Przy każdym uruchomieniu programu trzeba od początku tłumaczyć jego kod źródłowy na kod wynikowy zrozumiały dla komputera. Zatem do każdego wykonania programu musimy użyć interpretatora.
- Kompilator tłumaczy program źródłowy, przy okazji wykrywa i wyświetla błędy składniowe, które muszą być poprawione zanim przejdziemy do uruchamiania programu. Program może być kompilowany wielokrotnie zanim usuniemy wszystkie tego typu błędy (syntax errors).

Kodowania, uruchamianie i testowanie

- Kompilator nie tłumaczy całego programu, lecz tylko jego szkielet: reszta (preprogramowane procedury) jest trzymana w bibliotekach na dysku. Gdy proces tłumaczenia przez kompilator przebiegnie pomyślnie, uruchamiany jest program łączący (linker), który odszukuje w bibliotekach brakujące fragmenty i łączy je z programem wynikowym (object module) utworzonym przez kompilator. Tworzy się wtedy program wynikowy (load module) w języku maszynowym.

Kodowania, uruchamianie i testowanie