Algorytmy, reprezentacja algorytmów.

Wprowadzenie do algorytmów

Najważniejszym pojęciem algorytmiki jest **algorytm** (ang. algorithm). Nazwa pochodzi od nazwiska perskiego astronoma, astrologa, matematyka i geografa **Muhameda ibn Musa Al-Kwarizmiego** (arab. محمد بـ ن مو سى خوارزمى)żyjącego w latach od około 780 do 850 n.e. Był on autorem dzieła pt. **Algebra**, opisującego sposoby rozwiązywania różnych problemów matematycznych, np. równań wielomianowych.

Określenie algorytmu:

Algorytm jest ściśle określonym, skończonym i uporządkowanym ciągiem operacji, których wykonanie nad dobrze zdefiniowanym zbiorem danych prowadzi do rozwiązania określonej klasy problemów.

Z podanej definicji wynika kilka istotnych własności algorytmów.

- **Jednoznaczność** - operacje wykonywane przez algorytm nie można dowolnie interpretować - musi być określony jeden sposób ich wykonania.
- **Wykonalność** - operacje muszą być wykonalne.
- **Skończoność** - liczba operacji może być bardzo duża, ale skończona. Wykonanie każdej operacji zajmuje pewien czas. Nieskończona liczba operacji będzie się wykonywała w nieskończoność, to co nam po takim algorytmie?
- **Porządek** - operacje muszą być wykonywane w ustalonej kolejności, aby było wiadome, którą operację wykonać jako następną.
- **Ogólność** - algorytm powinien rozwiązywać klasę problemów, a nie jeden przypadek szczególny. Np. algorytm obliczania sumy 2 + 2 = 4 jest złym algorytmem - lepszym jest algorytm

dodawania dowolnych liczb, którego uczyłeś się na lekcjach matematyki w szkole podstawowej.

- **Efektywność** - algorytm powinien dochodzić do rozwiązania najkrótszą drogą - czasem jest to trudne do spełnienia, gdyż możemy nie znać najkrótszej drogi.

Dalej w podanej definicji czytamy, iż ciąg operacji wykonywany jest nad dobrze zdefiniowanym zbiorem danych. Zatem algorytm, oprócz definicji operacji, wymaga również definicji danych, nad którymi pracuje. Definicję danych nazywamy **specyfikacją algorytmu** lub **specyfikacją problemu**. Specyfikacja składa się zwykle z dwóch podstawowych części:

- **Definicja danych wejściowych** - określa ona jakie informacje potrzebne są algorytmowi do znalezienia rozwiązania. Np. dla równania liniowego ax + b = c algorytm potrzebuje współczynników a, b i c do znalezienia rozwiązania. W definicji podajemy również ograniczenia dla danych wejściowych, jeśli takowe istnieją. Np. w powyższym przykładzie liczby a, b i c mogą być dowolne, z zastrzeżeniem, iż a jest różne od 0.
- **Definicja danych wyjściowych** - określa efekt pracy algorytmu, czyli co jest jego wynikiem, co otrzymamy po zastosowaniu algorytmu dla danych wejściowych. Np. dla powyższego równania liniowego ax + b = c, wynikiem będzie wartość x, która po wstawieniu do równania spełnia je.
- **Definicja danych pomocniczych** - ta część nie jest obowiązkowa, jednakże znacząco ułatwia implementację algorytmu definiując pomocnicze struktury danych, które są niezbędne w trakcie przetwarzania przez algorytm danych wejściowych.

Algorytm można przedstawić graficznie w sposób następujący:

Sposoby reprezentacji algorytmów

Dobrze udokumentowany algorytm powinien składać się ze specyfikacji oraz opisu operacji, który spełnia podaną na początku rozdziału definicję. Istnieje kilka użytecznych sposobów opisywania operacji, poniżej podajemy te najbardziej popularne:

Opis słowny

Operacje do wykonania opisujemy zwykłym tekstem opisowym. Tę formę prezentacji algorytmu stosuje się najczęściej w fazie wstępnej, gdy chcemy w sposób ogólny opisać operacje bez wdawania się w szczegóły techniczne. Zwykle na podstawie opisu słownego tworzymy w następnym kroku bardziej ścisłą reprezentację.

Przykład:

Algorytm obliczania obwodu i pola prostokąta.

Dane wejściowe:

a,b - długości boków prostokąta

Dane wyjściowe:

pole, obwód

Oblicza pole jako iloczyn boku a przez bok b prostokąta.

Oblicz obwód jako sumę boków a i b pomnożoną przez 2.

Lista kroków

Każdą operację zapisujemy w osobnym, numerowanym kroku algorytmicznym. Stosuje się ograniczoną liczbę operacji elementarnych. Lista kroków pozwala precyzyjnie zdefiniować cały algorytm.

Przykład:

Algorytm obliczania obwodu i pola prostokąta.

Dane wejściowe:

a,b - długości boków prostokąta

Dane wyjściowe:

pole, obwód

K01: Czytaj a,b

K02: pole ← a × b

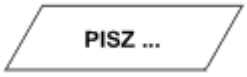
K03: obwód ← 2 × (a + b)

K04: Pisz pole, obwód

K05: Zakończ

Schemat blokowy

Operacje przedstawiamy w sposób graficzny za pomocą następujących symboli:

	oznacza początek algorytmu, czyli punkt startowy, od którego rozpoczynamy wykonywanie operacji.
	oznacza koniec algorytmu.
	określa następną operację do wykonania
	wewnątrz tego symbolu umieszczamy operację do wykonania nad danymi
	operacja wprowadzania danych do algorytmu - określa dane wejściowe, które algorytm musi odczytać.
	operacja wyprowadzania wyników - określa dane wyjściowe, które produkuje algorytm.
	test umożliwiający tworzenie rozgałęzień w algorytmie. Jeśli test jest spełniony, to z symbolu wychodzimy drogą oznaczoną strzałką TAK. W przeciwnym wypadku idziemy po strzałce oznaczonej NIE. Często umieszcza się przy symbolu testu tylko jeden napis TAK lub NIE. Drugi jest oczywisty.

Przykład:

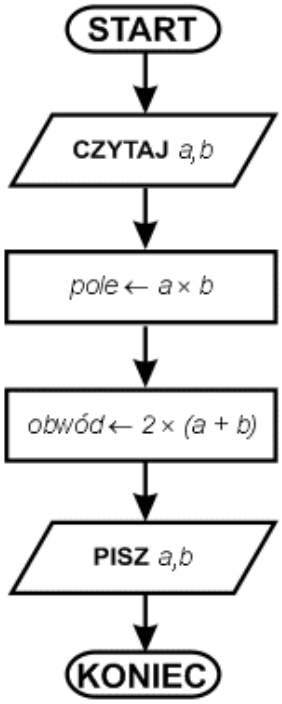
Algorytm obliczania obwodu i pola prostokąta.

Dane wejściowe:

a,b - długości boków prostokąta

Dane wyjściowe:

pole, obwód



Program komputerowy

Tak, nie pomyliliśmy się. Program komputerowy jest niczym innym, jak jednym ze sposobów zapisu algorytmu przy pomocy środków dostarczanych przez język programowania. Dlatego Algorytmika jest tak ważna przy nauce programowania. Poprzednie sposoby reprezentacji algorytmów są ogólne, nie wymagają znajomości konkretnego języka programowania. Jest to zaletą, gdyż zwykle algorytm jest niezależny od konkretnego języka programowania i zapis algorytmu w postaci ogólnej pozwala później zaimplementować go w dowolnym, wybranym przez programistę języku programowania. Dlatego autorzy opracowań często wybierają jedną z poprzednich form przedstawiania algorytmów, aby uniezależnić się od zmian popularności tego, czy innego narzędzia programistycznego - np. znany autorytet w dziedzinie informatyki, profesor Donald Knuth, stworzył na potrzeby swojego dzieła "Sztuka programowania komputerów" fikcyjną maszynę cyfrową o dobrze zdefiniowanej liście poleceń i wszystkie prezentowane algorytmy przedstawiał w

asemblerze tej maszyny. W ten sposób uniezależnił się od zmian stosowania języków programowania, a było ich trochę - Fortran, C, Pascal, C++, Java, PHP, Python, Ruby i tysiące innych.

Znane jest słynne powiedzenie Knutha:

"Languages come and go, but algorithms stand the test of time."
"Języki programowania przychodzą i odchodzą, lecz algorytmy opierają się próbie czasu"

```
#include <iostream>

using namespace std;

int main()
{
  unsigned int a,b,pole,obwod;

  cout << "Obliczanie pola i obwodu prostokata\n"
          "-----------------------------------\n\n";
// odczytujemy długości boków a i b

  cout << "Bok a = "; cin >> a;
  cout << "Bok b = "; cin >> b;
  cout << endl;

// wykonujemy obliczenia obwodu i pola

  obwod = 2 * (a + b);
  pole  = a * b;

// wyświetlamy wyniki obliczeń

  cout << "Obwod = " << obwod << endl
       << "Pole  = " << pole << endl << endl;

  system("pause");
  return 0;
}
```