

Zaawansowane metody programowania

Dr Zbigniew Koziol - wykład
Mgr Mariusz Woźny - laboratorium

Wykład IV

Algorytmy

Drzewa, grafy, etc... Najpierw o algorytmach



General Feldmarschall
Albrecht Theodor Emil
Graf von Roon

Algorytmy

In mathematics and computer science, an algorithm is a self-contained step-by-step set of operations to be performed. Algorithms exist that perform calculation, data processing, and automated reasoning.

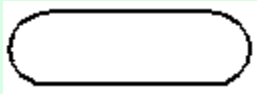
An algorithm is an effective method that can be expressed within a finite amount of space and time and in a well-defined formal language for calculating a function. Starting from an initial state and initial input (perhaps empty), the instructions describe a computation that, when executed, proceeds through a finite number of well-defined successive states, eventually producing "output"

Algorytm – skończony ciąg jasno zdefiniowanych czynności, koniecznych do wykonania pewnego rodzaju zadań.

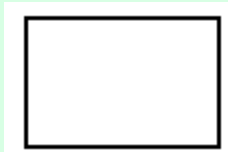


Schematy blokowe

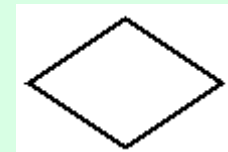
- Poszczególne elementy schematu łączy się za pomocą strzałek. W większości przypadków blok ma jedną strzałkę wchodzącą i jedną wychodzącą, lecz są także wyjątki



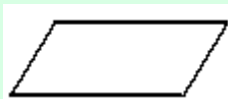
Początek lub koniec algorytmu.



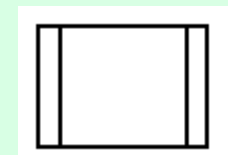
Symbol procesu. W jej obrębie umieszczamy wszelkie obliczenia lub podstawienia. Proces ma dokładnie jedną strzałkę wchodzącą i dokładnie jedną strzałkę wychodzącą.



Blok decyzyjny. Umieszcza się w nim jakiś warunek (np. " $x > 2$ "). Każdy romb ma dokładnie jedną strzałkę wchodzącą oraz dokładnie dwie strzałki wychodzące.



Odczyt lub zapis danych



Proces, który został już kiedyś zdefiniowany. Można ją porównać do procedury, którą definiuje się raz w programie, by następnie móc ją wielokrotnie wywoływać. Warunkiem użycia jest więc wcześniejsze zdefiniowanie procesu. Podobnie jak w przypadku zwykłego procesu i tu mamy jedno wejście i jedno wyjście.

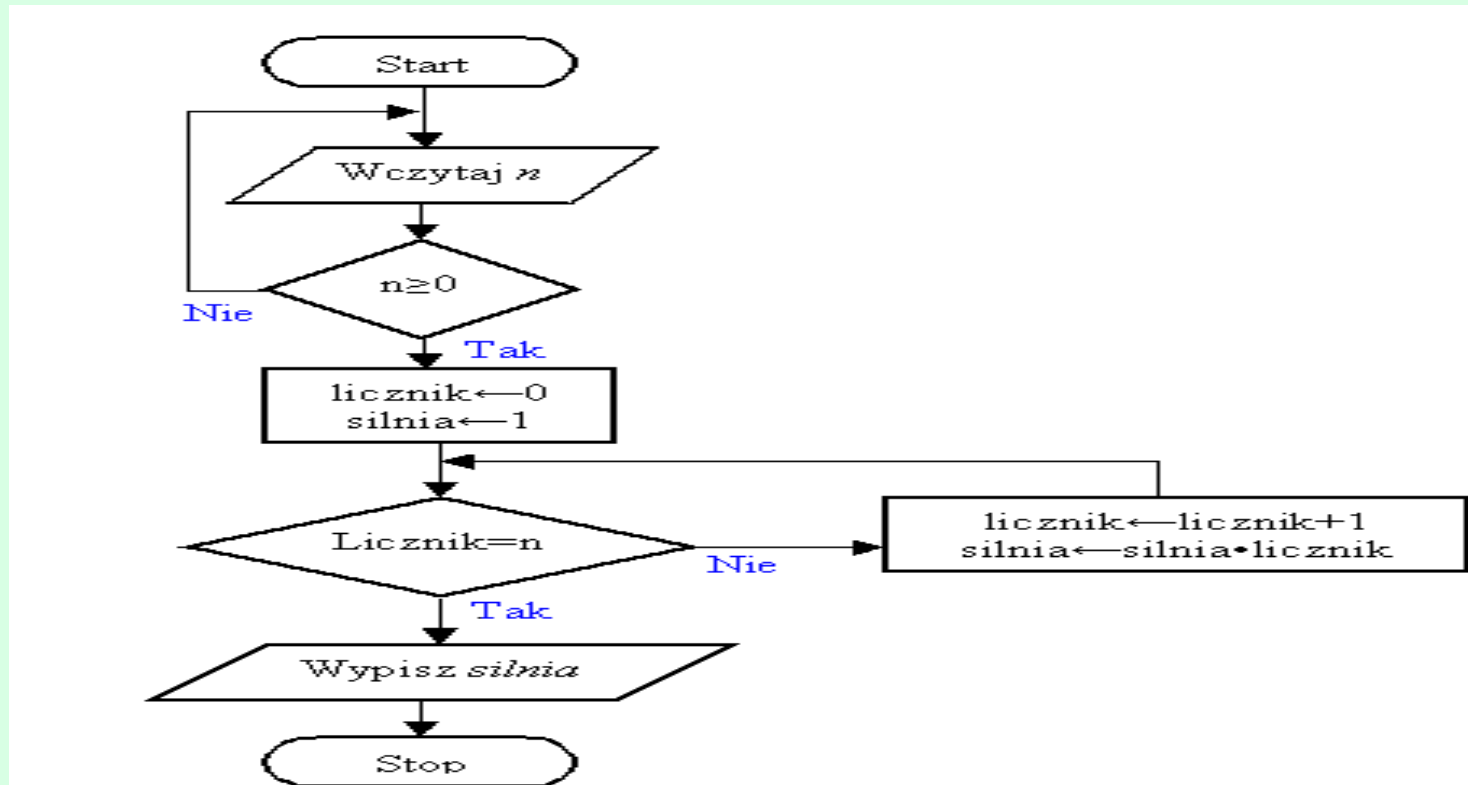
Schematy blokowe

Przykładowy algorytm za pomocą schematu blokowego.

Zdefiniujmy iteracyjną wersję silni. Dla przypomnienia: rekurencyjna definicja silni:

$$0! = 1$$

$$n! = n * ((n-1)!)$$



Najważniejsze techniki implementacji algorytmów komputerowych

proceduralność – algorytm dzielimy na szereg podstawowych procedur, wiele algorytmów współdzieli wspólne biblioteki standardowych procedur, z których są one wywoływane w razie potrzeby;

praca sekwencyjna – wykonywanie poszczególnych procedur algorytmu, według kolejności ich wywołań, naraz pracuje tylko jedna procedura;

praca wielowątkowa – procedury wykonywane są sekwencyjnie, lecz kolejność ich wykonania jest trudna do przewidzenia dla programisty;

praca równoległa – wiele procedur wykonywanych jest w tym samym czasie, wymieniają się one danymi;

rekurencja – procedura lub funkcja wywołuje sama siebie, aż do uzyskania wyniku lub błędu;

algorytm probabilistyczny – działa poprawnie z bardzo wysokim prawdopodobieństwem, ale wynik nie jest pewny.

Przykład rekurencji (Pascal, `recurrent.pas` ← [link](#))

```
Function Factorial(n : Integer) : Integer;
Var
    my_string: String;
Begin

    STR(n:3,my_string);
    writeln('in='+my_string);

    If n = 1 then
        Factorial := 1 Else
        Factorial := n*Factorial(n-1);

    STR(Factorial:3,my_string);
    writeln('out='+my_string);

End;

BEGIN
    Factorial(7);
END.
```

Przykład algorytmu. Znajdywanie największej liczby w zbiorze danych.

One of the simplest algorithms is to find the largest number in a list of numbers of random order. Finding solution requires looking at every number in the list.

Algorithm:

1. If there are no numbers in the set then there is no highest number.
2. Assume the first number in the set is the largest number in the set.
3. For each remaining number in the set: if this number is larger than the current largest number, consider this number to be the largest number in the set.
4. When there are no numbers left in the set to iterate over, consider the current largest number to be the largest number of the set.

Przykład implementacji algorytmu znajdowania największej liczby w zbiorze danych, w języku perl.

```
my $maxA # Inicjowanie $maxA

my $index = 0; # $index będzie trzymać numer linii danych wejściowych

while(my $a=<>) {

    $index++; # zwiększamy index o 1

    $a =~ s/\n//; # usuwamy znak końca linii

    if ($index == 1) {
        $maxA = $a;
    }

    if ($a > $maxA) {
        $maxA = $a;
    }
}

print "$maxA\n";
```

Przykład implementacji algorytmu znajdowania największej liczby w zbiorze danych, w języku perl.

Program `maximum.pl` ([← link](#)) z danymi w pliku `maximum.dat` ([← link](#)) uruchamiamy z okna terminala komendą: `perl ./maximum.pl < maximum.dat`

**Przykład implementacji algorytmu
probabilistycznego, w języku perl.
Poszukujemy minimum funkcji $y=x^2$**

```
my $minY, $minX; # Inicjowanie $minY, $minX; Wynik, którego szukamy
my $Xmin = -1.5; my $Xmax = 0.5; # Przedział X
my $DX = $Xmax - $Xmin;
my $No = 10; # liczba przypadkowych liczb generowanych

for (my $i=0; $i<$No; $i++) {
    my $randomX = rand(); # a random number between 0 and 1
    my $x = $Xmin + $DX * $randomX;
    my $y = $x*$x; # funkcja, której minimum szukamy
    if ($i==0) {
        $minX = $x;    $minY = $y;
    } else {
        if ($y < $minY) {
            $minX = $x; $minY = $y;
        }
    }
}
print "X: $minX\tY:$minY\n";
}
```

Algorytm probabilistyczny. Własności.

1. Wynik za każdym razem inny.
2. Wynik tym dokładniejszy im więcej prób. Ale... niekoniecznie zawsze!
3. Wbrew pozorom, metoda jest bardzo użyteczna. W większości symulacji / modelowania komputerowego wcale bowiem nie musimy poznać dokładnego wyniku!

Algorytm probabilistyczny, w szczególnej odmianie, znany jest jako metoda Monte Carlo.

Monte Carlo, Monaco



Algorytm probabilistyczny, w przykładzie opisanym, nadaje się także świetnie do implementacji w formie algorytmu wielowątkowego, lub równoległego.

In computer science, a parallel algorithm, as opposed to a traditional serial algorithm, is an algorithm which can be executed a piece at a time on many different processing devices, and then combined together again at the end to get the correct result.

Algorytmy wielowątkowe lub równoległe. Algorytmy rozprzestrzenione (distributed).

GPU (Nvidia), threads, Google (maps and searching), C++

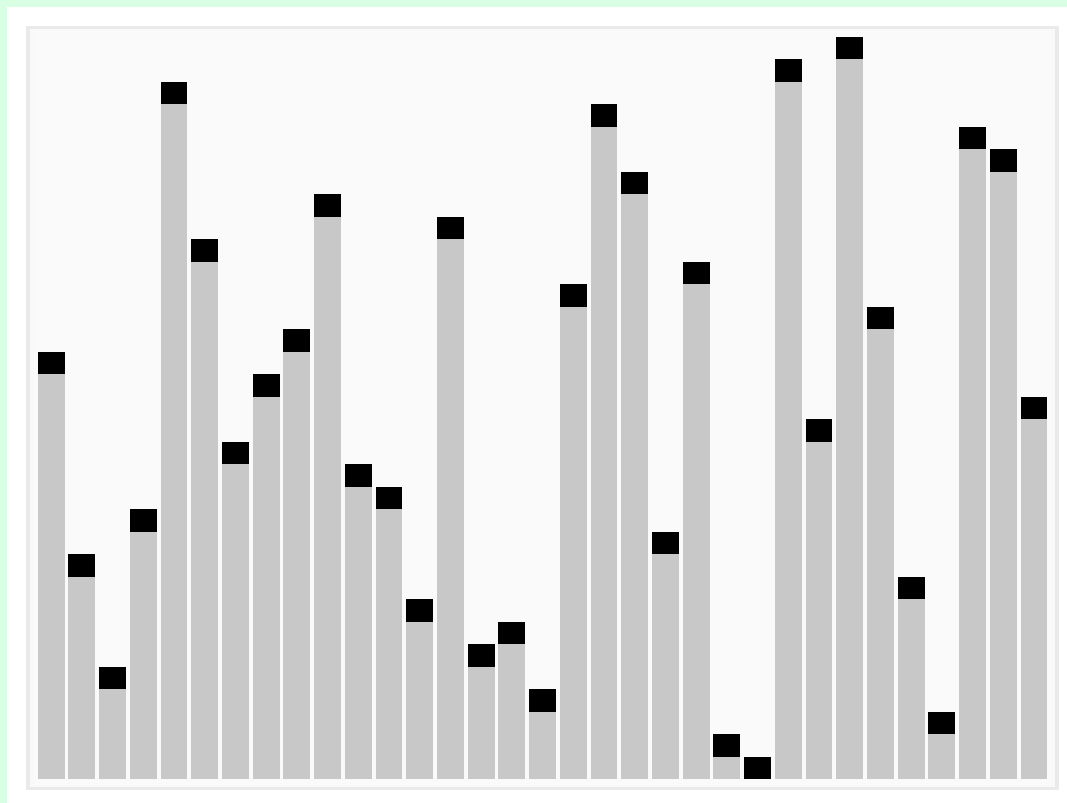


© Universal Pictures

**Przykład innego algorytmu znajdowania największej liczby w zbiorze liczb przypadkowych.
Algorytm dokonuje również sortowania liczb.**

An animation of the **quicksort** algorithm sorting an array of randomized values. The red bars mark the pivot element; at the start of the animation, the element farthest to the right hand side is chosen as the pivot.

[Quicksort - Wikipedia](#) ←
[link \(ANG\)](#)



Literatura / Źródła

- [Handbook of Algorithms and Data Structures](#) ← link (ANG)
- [Dictionary of Algorithms and Data Structures](#) ← link (ANG)
- [Archiwum gotowych rozwiązań zadań algorytmicznych](#) ← link (PL)
- [Algorytmy i struktury danych](#) ← link (PL)
- [Algorytm - Wikipedia](#) ← link (PL)
- [Algorithm - Wikipedia](#) ← link (ANG)
- [Pascal Programming](#) ← link (ANG)
-
-
-