1 Algorytmy

Algorytm – jest to skończony, uporządkowany ciąg czynności mających na celu przetworzenie informacji wejściowych na informacje wyjściowe- informacje wejściowe często nazywane są danymi a wyjściowe wynikami.

inna definicja:
Algorytmem nazywamy uporządkowany zbiór operacji, taki, ze po ich wykonaniu otrzymuje się rozwiązanie dowolnego zadania z określonej klasy zadań

Dla tak sprecyzowanego pojęcia algorytmu możemy podać zestaw cech, jakie powinien on posiadać:

- ogólność-co oznacza, ze algorytm przeznaczony jest do rozwiązywania określonej klasy zadań, a nie tylko pojedynczego szczególnego przypadku (czyli np. rozwiązywanie równań różniczkowych 2 rzędu, a nie konkretnego równania).
- Skończoność -możliwość uzyskania rozwiązania problemu w skończonej ilości kroków.
- Określoność - jednoznaczność wszystkich wykonywanych w nim operacji.
- efektywność - przez co rozumiemy czas potrzebny na wykonanie tego algorytmu (najczęściej liczony w umownych jednostkach np. ilości wykonywanych operacji w zależności od danych wejściowych) lub zapotrzebowanie algorytmu na pamięć.

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.

Istnieje co najmniej kilka sposobów zapisu algorytmów Poniżej zostaną przedstawione najważniejsze z nich.

- **Język naturalny** - najbardziej rozpowszechniony w życiu sposób zapisu algorytmów Jego zaletami są: prostota, brak potrzeby przyswajania specyficznych konstrukcji formalnych, szeroki zasób możliwego do użycia słownictwa; wadami: mała precyzja oraz możliwość błędnej interpretacji (mała ścisłość zapisu), mała zwięzłość

- **Schematy blokowe** - sformalizowany zapis algorytmów Algorytm prezentowany jest w postaci bloków operacyjnych oraz linii ilustrujących przepływ sterowania. Jest to sposób graficzny. Wadą jest brak (technicznie) możliwości przedstawiania dużych algorytmów
- **Języki formalne**- najczęściej używany w praktyce sposób zapisu algorytmu. Językiem formalnym możne być dowolny z języków programowania lub specjalnie stworzona w tym celu notacja. Zapewnia ścisłość zapisu.
- **Lista kroków**

Etapy rozwiązywania problemów:

1. Sformułowanie zadania
2. Określenie danych wejściowych
3. Określenie celu, czyli wyniku
4. Poszukiwanie metody rozwiązania, czyli algorytmu
5. Przedstawienie algorytmu w postaci:
- opisu słownego
- listy kroków
- schematu blokowego
- jednego z języków programowania
6. Analiza poprawności rozwiązania
7. Testowanie rozwiązania dla różnych danych - ocena efektywności przyjętej metody

Symbole graficzne w schematach blokowych programów według PN-75/E-01226

Opis podstawowych symboli – plik „opis_symboli_blokowych.pdf”

Fragmenty artykułu: ALGORYTMY, ALGORYTMIKA I ALGORYTMICZNE MYŚLENIE W SZKOLE - *Maciej M. Sysło, Instytut Informatyki, Uniwersytet Wrocławski*

Do pewnego momentu, gdzieś w połowie zeszłego stulecia, słowo algorytm w potocznym znaczeniu nierozerwalnie pojawiało się w zestawieniu **algorytm Euklidesa**. Co najmniej dziwnym, bo przecież przepis Euklidesa na znajdowanie największego dzielnika dwóch liczb ma grubo ponad 2000 lat, a słowo algorytm zaczęło się wykluwać gdzieś pod koniec pierwszego tysiąclecia. Uzasadnienie jest jednak proste – algorytm Euklidesa jest najprostszą, a jednocześnie najbardziej pojemną konstrukcją umożliwiającą wyjaśnienie niemal wszystkich podstawowych cech konstrukcji algorytmicznych. Nieco później, w latach pięćdziesiątych, algorytmem był przede wszystkim przepis na obliczenia numeryczne, gdyż pierwsze komputery służyły głównie do takich rachunków.
Obecnie, na dobrą sprawę, każdą czynność, w jakimś sensie celową i składającą się z uporządkowanych etapów, można by nazwać i często nazywa się algorytmem.

Pojęcie „algorytm” swoją popularność i rozpowszechnienie zawdzięcza przede wszystkim informatyce. A powodem są komputery. W bardzo uproszczony sposób, związek algorytmów z komputerami, czyli pośrednio z informatyką, można ująć następująco – komputery głównie wykonują programy, a każdy program jest zapisem jakiegoś algorytmu.
Algorytmy można zapisywać na papierze, na tablicy lub opisać słownie. Tak zresztą często postępujemy również na lekcjach informatyki. Ale na tym nie koniec. Dopiero uzmysłowienie sobie, że ten przepis ma być wykonany przez komputer czyni z niego twór informatyczny. Najlepiej ujmuje ten fakt powiedzenie, którego autorem jest jeden z największych żyjących informatyków, Donald E. Knuth:

Mówi się często, że człowiek dotąd nie zrozumie czegoś, zanim nie nauczy tego – kogoś innego.

W rzeczywistości,
człowiek nie zrozumie czegoś naprawdę,
zanim nie zdoła nauczyć tego – komputera.

A zatem **algorytm**, to **tak skonstruowany przepis, że może go zrozumieć i wykonać komputer**. Oczywiście do wszystkiego jest potrzebny człowiek, który ma przygotować i podać ten przepis komputerowi. Stąd, ze względu na wykonawcę-komputer, o algorytmie mówi się m.in., że składa się z uporządkowanych kroków, jednoznacznie opisanych, wykonalnych za pomocą prostych i znanych operacji, ma skończone działanie itd. Dodatkowo, opis kroków algorytmu powinien być poprzedzony **specyfikacją**, czyli precyzyjnym opisem wejścia i wyjścia, a więc danych i wyników. To wszystko po to, by siadając do tłumaczenia komputerowi, co ma zrozumieć i wykonać, ani tłumaczący, ani później komputer nie mieli żadnych wątpliwości, o co chodzi.

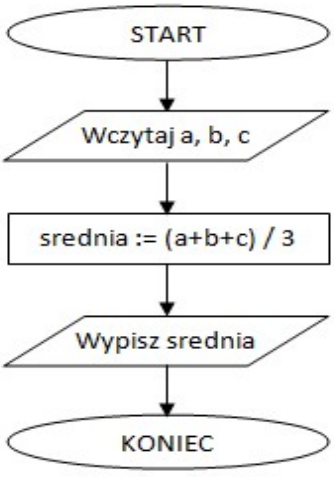
Rodzaje algorytmów:

a) **Algorytm liniowy (sekwencyjny)** – kolejność wykonywanych czynności nie jest zależna od wartości danych wejściowych

Przykład :
Algorytm obliczający średnią arytmetyczną podanych liczb:

Dane wejściowe: trzy liczby **a**, **b**, **c**.
Dane wyjściowe: liczba **srednia**, będąca średnią arytmetyczną tych liczb.

1. Start
2. Wczytaj **a**, **b**, **c**
3. srednia:=(a+b+c)/3
4. Wypisz **srednia**
5. Koniec.



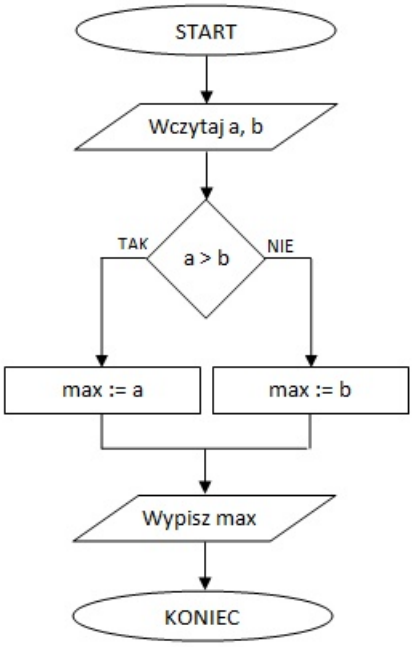
b) **Algorytm z rozgałęzieniami (warunkowy)**– kolejność wykonywania operacji zależy od wartości danych wejściowych

Przykład :

Algorytm zwracający wartość maksymalną z dwóch wprowadzonych wartości

Dane wejściowe: dwie liczby **a**, **b**.
Dane wyjściowe: liczba **max**, będąca wartością maksymalną.

1. Start
2. Wczytaj **a**, **b**
3. Jeśli **a>b** to **max:=a**, w przeciwnym razie **max:=b**
4. Wypisz **max**
5. Koniec.



c) **Algorytm iteracyjny**-zawiera instrukcje, które nakazują wielokrotne powtarzanie pewnych czynności. Iteracje występują w dwóch podstawowych odmianach:
iteracja z określoną liczbą powtórzeń : wykonuj czynność dokładnie N razy, iteracja warunkowa: wykonaj czynność, dopóki jest spełniony warunek. Algorytm iteracyjny może działać na danych o dowolnej długości (wielkości).

d) **Algorytm rekurencyjny-** Charakterystyczną cechą funkcji (procedury) rekurencyjnej jest to, że wywołuje ona samą siebie.

2. Instrukcja warunkowa

if (warunek) instrukcja1;
else instrukcja2;

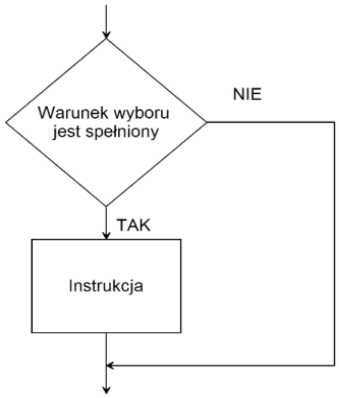
warunek - jest wyrażeniem logicznym, które może przyjmować wartość 0 – gdy nie jest spełnione, albo wartość różną od 0 - wtedy jest traktowane jako spełnione, prawdziwe. W warunkach często stosuje się operatory porównań: < <= == >= > != - różny

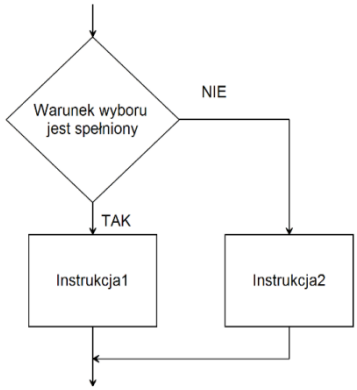
instrukcja1 - instrukcja wykonywana, gdy warunek ma wartość różną od 0, czyli gdy jest prawdziwy (true)

instrukcja2 - instrukcja wykonywana, gdy warunek ma wartość równą 0, czyli gdy jest fałszywy (false).

Człon z else może być pominięty, jeśli nasz algorytm nie przewiduje wykonania żadnej operacji przy fałszywym warunku. Wtedy instrukcja warunkowa upraszcza się do postaci:

if(*warunek*) *instrukcja;*





Jeśli w instrukcji warunkowej **if** musimy wykonać więcej niż jedną operację, to stosujemy tzw. instrukcję złożoną, czyli grupę instrukcji objętych klamrami:

```
if (warunek)  {instrukcja1_1;
               instrukcja1_2;
              }
                        else   {instrukcja2_1;
                                instrukcja2_2;}
```

Zagnieżdżanie instrukcji warunkowych:

```
if (warunek1)  instrukcja1;
                        else  if(warunek2)   instrukcja2;
                                    else instrukcja3;
```

Przykład 1
Program obliczający wartość bezwzględną podanej na wejściu liczby rzeczywistej.
Do wprowadzania liczby wykorzystamy okno dialogowe prompt.

Rozwiązanie:

```
<!DOCTYPE html>
```

```
<html>
<head>
<meta charset="UTF-8">
</head>
<body>
<script>
var liczba=Number(prompt("podaj liczbę",2));
//konwersja typu string na liczbę rzeczywistą
 if (liczba>=0) {document.write("wartość bezwzgledna liczby "+liczba+ "  wynosi"+liczba);}
else {document.write("wartość bezwzgledna liczby "+liczba+ "  wynosi  "+ (-1*liczba));}
</script>
</body>
</html>
```

Sprawdź działanie programu. Jaki będzie wynik działania aplikacji, gdy wprowadzimy ciąg znaków nie będący liczbą?
W jaki sposób można zabezpieczyć program przed wprowadzeniem niepoprawnych danych?

Przykładowe rozwiązanie:

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
</head>
<body>
<script>
var liczba=Number(prompt("podaj liczbę",2));
//konwersja typu string na liczbę rzeczywistą
if (!isNaN(liczba))  {
   //jezeli wprowadzony ciąg znaków można przekonwertować na liczbę
   //to wykonujemy porównanie
 if (liczba>=0) {document.write("wartość bezwzgledna liczby "+liczba+ "  wynosi"+liczba);}
else {document.write("wartość bezwzgledna liczby "+liczba+ "  wynosi  "+ (-1*liczba));}
   }
    else  {
      //jeżeli nie to wypisujemy stosowny komunikat
      document.write("wprowadzono niepoprawne dane");}
</script>
</body>
</html>
```

Przykład 2

Napisać skrypt, który dla dowolnych wartości współczynników a, b, c równania : $ax^2+bx+c=0$ zwróci informację o liczbie rozwiązań równania, a w przypadku, gdy równanie posiada rozwiązanie, wypisze to rozwiązanie. Przedstaw algorytm za pomocą schematu blokowego. Podaj specyfikację

Przykładowy skrypt

```
<!DOCTYPE html>
<html lang="pl">
<head>
<meta charset="UTF-8">
```

```
</head>
<body>
<script>
var a=Number(prompt("podaj parametr a", 1));
var b=Number(prompt("podaj parametr b", 1));
var c=Number(prompt("podaj parametr c", 1));
if(a!=0)
      {delta=b*b-4*a*c;
       if(delta>0)  { var x1=(-b-Math.sqrt(delta))/(2*a);
                    var x2=(-b+Math.sqrt(delta))/(2*a);
                   document.write("<br> równanie ma dwa rozwiązania  x1="+x1+",   x2="+x2);
                  }
                       else if (delta==0){ var x0=-b/(2*a);
                                    document.write("<br> równanie ma jedno rozwiązanie  x0="+x0);
                                                }
                                  else {document.write("<br> Równanie nie ma rozwiązania ");}
              }
                       else if(b!=0)    {var x0=-c/b;
                                     document.write("<br> równanie ma jedno rozwiązanie  x0="+x0);
                                    }
                 else if (c==0) {document.write("<br> Równanie ma nieskończenie wiele rozwiązań ");}
                       else       {document.write("<br> Równanie nie ma rozwiązania");}
</script>
</body>
</html>
```

3 Operator warunkowy „?” Instrukcja przetwarzania warunkowego- stanowi skrócona wersję instrukcji warunkowej if...else

- Operator warunkowy ?:

```
arg > 0 ? 1 : −1;
```

SKŁADNIA

(warunek)? instrukcja_1:instrukcja_2

przykład:

Zadaniem skryptu jest sprawdzenie, czy wprowadzona przez użytkownika liczba jest parzysta, czy nieparzysta

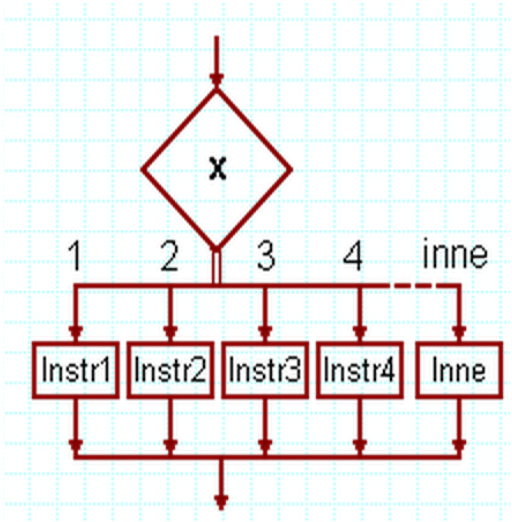
```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
</head>
<body>
<script>
var a=prompt("podaj liczbę całkowitą",0);  //Uwaga! Nie  ma rzutowania na typ Number
var x;
x=(a%2==0)?"parzysta":"nieparzysta";  // a%2 - automatyczna konwersja do typu Number
```

```
document.write("<br>"+a+"  to liczba  "+x);
</script>
</body>
</html>
```

4 Instrukcja wyboru (szczególny rodzaj instrukcji warunkowej)

składnia:

```
switch (wyrażenie){
    case wartość1 :
    instrukcja1;
    break;
    case wartość2 :
    instrukcja2;
    break;
    case wartość3:
    instrukcja3;
    break;
    default:
    instrukcja4;
    }
```



Przykład

I sposób – wykorzystanie *if*

```
if (a == 0)
    alert("a jest równe zero");
else if (a == 1)
    alert("a jest równe jeden");
else if (a == 2)
    alert("a jest równe dwa");
else if (a == 3)
    alert("a jest równe trzy");
else
    alert("a ma inną wartość");
```

II sposób – wykorzystanie *instrukcji wyboru*

```
switch (a) {
case 0:
    alert("a jest równe zero");
    break;
case 1:
    alert("a jest równe jeden");
    break;
case 2:
    alert("a jest równe dwa");
    break;
case 3:
    alert("a jest równe trzy");
    break;
default:
```

```
    alert("a ma inną wartość");
    break;
}
```

Po instrukcjach case i default można umieścić kilka instrukcji bez konieczności otaczania ich nawiasami klamrowymi { }.

Zwróć także uwagę na słowo kluczowe break umieszczone na końcu każdej z sekcji wewnątrz instrukcji switch. Instrukcja ta mówi w którym momencie ma zostać przerwane wykonywanie instrukcji switch. W przypadku gdyby jej nie było, wykonane byłyby także instrukcje z znajdującej się poniżej sekcji case (lub default). Zapomnienie o wstawieniu break jest przyczyną błędów w skryptach, które niekiedy mogą być trudne do wykrycia.

- Instrukcja wyboru - składniowo podobne do instrukcji z C++ i Javy,
  - Wyrażenia etykiet są wartościowane w czasie uruchomienia.
  - Etykieta może być wyrażeniem dowolnego typu.

```
switch(dzien) {
    case "poniedziałek":
        return 1;
    case "wtorek"
        return 2;
    //...
    default:
        return -1;
}
```

Przykład:
Opracuj skrypt obliczający stopień na podstawie liczby otrzymanych punktów
Kryteria: 0.. 39 pkt. - 1
40..49pkt -2
50..59 pkt. - 3
60.. 69 pkt. - 3.5
70.. 79 pkt. - 4
80.. 89 pkt. - 4.5
90..100 pkt. - 5

Przykładowe rozwiązanie:

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
</head>
```

```
<body>
<script>
var lp=prompt("podaj liczbę punktów(0<=lp<=100)",0);
/*Nie zastosujemy rzutowania na typ Number
Ze względu na działania matematyczne nastąpi automatyczna konwersja typu String do typu Number
*/
var stopien;
lp=Math.floor(lp/10);
/*Obiekt Math przechowuje wartości matematyczne, zgromadzone jako własności i metody tego
obiektu
Metoda floor(liczba) - wraca największą liczbę całkowitą mniejszą lub równą argumentowi liczba
*/
switch(lp)
{case 4: stopien=2;
        break;
case 5: stopien=3;
        break;
 case 6: stopien=3.5;
        break;
case 7: stopien=4;
        break;
case 8: stopien=4.5;
        break;
case 9,10: stopien=5;
        break;
default: stopien=1;
}
document.write("<br>Otrzymałeś ocenę   "+stopien);

</script>
</body>
</html>
```

Zadania

1) (fragment zadania egzaminacyjnego, czerwiec 2014)
Napisać skrypt, który umożliwi użytkownikowi wprowadzenie liczby rzeczywistej. Wynikiem działania programu będzie informacja, czy wprowadzona liczba jest dodatnia, ujemna czy równa zero.
2) Napisz skrypt sprawdzający, czy wprowadzona przez użytkownika wartość jest większa niż 100 i podzielna przez 3.
3) Opracuj skrypt, który wypisze trzy liczby wprowadzone przez użytkownika w kolejności malejącej

4)

- Opracuj program obliczania podatku dochodowego według zasad podanych w tabeli. Użytkownik podaje podstawę obliczania podatku.

Podatek dochodowy od osób fizycznych

Skala podatkowa - 2007r

Podstawa obliczenia podatku (w zł)		Wartość podatku
poniżej	43 405	19% podstawy obliczenia minus kwota 572 zł 54 gr.
43 405	85 528	7 674 zł 41 gr. + 30% nadwyżki ponad 43.405 zł
85 528		20 311 zł 31 gr. + 40% nadwyżki ponad 85.528 zł

Kwota wolna od podatku **3.013 zł**

Opracuj stronę internetową, na której będzie zamieszczona powyższa tabelka oraz wynik działania skryptu. Formatowanie elementów strony umieść w pliku .css.