INFORMATYKA

ALGORYTMY

CEL LABORATORIUM

Celem zajęć jest zapoznanie uczestniczki/uczestnika z algorytmami i strukturami danych. Przedstawione zostaną pojęcia danych, zmiennych, struktur danych, algorytmów oraz ich przykłady w informatyce i w świecie rzeczywistym. Zaprezentowane zostaną sposoby zapisu algorytmów: lista kroków, pseudokod, schemat blokowy.

ALGORYTMY, DANE, ZMIENNE, STRUKTURY DANYCH

Od strony formalnej, przez algorytm rozumiemy **skończony ciąg reguł, które zastosowane na skończonej liczbie danych, pozwalają rozwiązać zbliżone do siebie klasy problemów**. Z algorytmami mamy do czynienia w codziennym życiu, chociażby pod postacią przepisów kulinarnych. Algorytmem jest opis postępowania na przejściu dla pieszych, głoszący „najpierw spójrz w lewo, później w prawo, a później znów w lewo".

Przy takiej interpretacji, za **dane** może być uznane wszystko co może posłużyć do uzyskania informacji. Dane są nierozerwalnie powiązane ze zmiennymi, przy czym **zmienne** rozumiemy jako wszelkie rzeczy, których stan może ulegać zmianie. W omawianym powyżej algorytmie postępowania na przejściu dla pieszych, zmienną jest obszar drogi, po lewej i prawej stronie przechodzącego. Danymi jest natomiast obecność (lub brak) samochodów w rzeczonym obszarze oraz prędkość i kierunek ich ewentualnego ruchu.

Dane mogą być zorganizowane w **struktury**. Przez **strukturę danych** rozumiemy pewną grupę danych, które są ze sobą powiązane. W świecie rzeczywistym, strukturą danych może być pojedynczy samochód, w którym może znajdować się jeden pasażer (dana) lub więcej. Wszystkich pasażerów cechuje wspólna cecha: są ludźmi (lub ogólniej: zwierzętami). Strukturą danych może też być człowiek gdyż każdego człowieka tyczą się pewne dane: rok urodzenia, kolor włosów, wysokość oraz wiele innych. Stąd wynika jasno, że dane (i struktury danych), mogą być **zagnieżdżone**.

W informatyce, pod postacią danych, najczęsciej występują różnego rodzaju liczby i znaki lub ciągi znaków. Zmiennymi będą specjalne miejsca w pamięci komputera (na dysku twardym lub w pamięci ulotnej), którym przypisano nazwę i można się do nich odnieść. Strukturami będą zbiory, tablice, listy i klasy. Algorytmy służą natomiast gotowymi rozwiązaniami na najczęstsze problemy i pozwalają programistom szybko wdrożyć rozwiązanie.

JAK SĄ PRZECHOWYWANE DANE?

Na poprzednich laboratoriach poznaliśmy system binarny. Wszystkie współczesne urządzenia elektroniczne korzystają z systemu binarnego do przechowywania danych. A jednak uzyskujemy nie tylko liczby, ale teksty, pliki graficzne, audio, filmy i wiele innych. Jak to się dzieje?

Kluczem jest standaryzacja. Tak, jak istnieje standard IEEE 754, który opisuje sposób zapisu w pamięci 32-bitowych liczb zmiennoprzecinkowych, tak też istnieją standardy opisujące sposób przechowywania innych rodzajów danych.

Najbardziej podstawowym standardem kodowania znaków, jest standard ASCII, który każdy znak zapisuje 7 bitach. Do każdej wartości, przypisany jest konkretny znak. Stąd, niezależnie czy korzystamy z komputera w domu czy telefonu komórkowego, widzimy takie same znaki.

SPOSOBY ZAPISU ALGORYTMÓW

Istnieją trzy główne sposoby zapisu algorytmów: lista kroków, schemat blokowy i zapis za pomocą pseudokodu. Każdy z tych sposobów zapisu jest **równoważny**, co znaczy, że kod opracowany przez programistę[1] powinien być taki sam, niezależnie od sposobu reprezentacji algorytmu.

Rozważmy więc przykładowy algorytm, opisujący zamianę liczby dziesiętnej na binarną. Z taką zamianą zapoznaliśmy się na poprzednich zajęciach laboratoryjnych, warto więc sobie ją przypomnieć i utrwalić. Zacznijmy od listy kroków:

1. Utwórz zmienną pomocniczą, która będzie przechowywać liczbę binarną. Nazwiemy ją „binarna".
2. Dopisz na początku zmiennej „binarna" resztę z dzielenia liczby dziesiętnej przez 2.
3. Liczbę dziesiętną podziel przez 2.
4. Jeżeli liczba dziesiętna jest większa niż 1, wróć do kroku 2.

I to tyle. Po zakończeniu („**na wyjściu**") wykonywania algorytmu, zmienna „binarna", powinna zawierać liczbę w systemie dwójkowym, która odpowiada liczbie w systemie dziesiętnym, która była dana na początku („**na wejściu**"). W rzeczywistych zastosowaniach, chcielibyśmy na końcu algorytmu dodać jeszcze jeden krok, który kazałby liczbę wypisać albo przekazać dalej („**zwrócić**").

Ten sam algorytm możemy zapisać za pomocą pseudokodu:

```
1. binarna := pusta lista
2. front(binarna) := dziesiętna/2
3. dziesiętna := dziesiętna/2
4. Jeżeli dziesiętna>1: wróć do kroku 2
```

Zwrócić należy uwagę, że zapis w pseudokodzie jest dużo bardziej formalny i pozbawiony opisu słownego. Korzysta też z symboli, w szczególności z symbolu „:=", oznaczającego przypisanie zmiennej pewnej wartości.

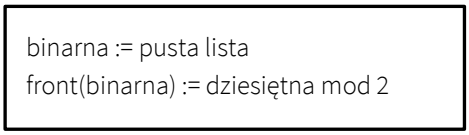
Ostatnią formą reprezentacji algorytmu jest schemat blokowy. Żeby móc korzystać ze schematów blokowych, konieczne jest najpierw zapoznanie się z elementami tworzącymi schemat.

[1] Zakładając oczywiście, że programista umie programować i rozumie algorytm.

SCHEMATY BLOKOWE

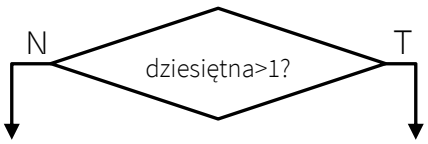
Schemat blokowy składa się z bloków, które są połączone strzałkami. Strzałki wskazują kolejny element, który należy rozpatrzeć. Najbardziej bazowymi blokami są bloki oznaczające początek i koniec schematu:

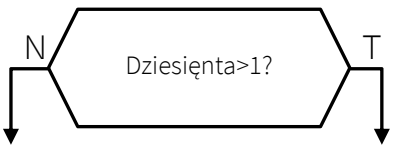
Kolejnym bardzo często występującym rodzajem bloku jest blok instrukcji, który zawiera tekst, opisujący polecenia, które należy wykonać. W przeciwieństwie do bloków START i KONIEC, które mają zaokrąglone rogi, blok instrukcji, jest prostokątem:



Spośród instrukcji, wyróżniają się dwa szczególne rodzaje: **zapis** i **odczyt** danych. Wyróżniamy je również w schematach blokowych, oznaczając je równoległobokami o pochylonych bokach, które dodatkowo mają linie po lewej (odczyt) lub prawej (zapis) stronie:

Ostatnim często występującym rodzajem bloków, są bloki instrukcji warunkowych. Blok ten pojawia się, gdy zachodzi konieczność podjęcia decyzji. Przyjmuje on kształt rombu lub sześciokątu, którego dwa boki są poziome i równe sobie, a pozostałe krótsze i równe sobie. W bloku instrukcji warunkowej, notuje się warunek, zakończony pytajnikiem, a linie prowadzi od boków, podpisując literami T (gdy warunek jest spełniony) oraz N (gdy nie jest):

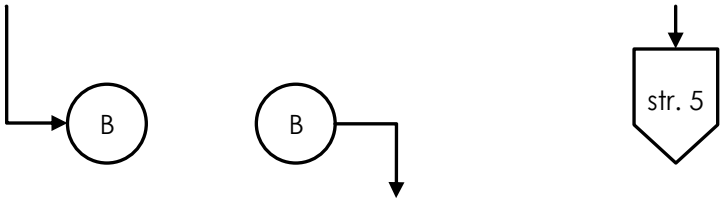




Oprócz tego, istnieją jeszcze bloki, które będziemy używać rzadziej, ale możesz się z nimi spotkać w literaturze i praktyce. Pierwszym z takich elementów jest blok procesu, którzy służy do zaznaczenia, że w tym miejscu powinien być wywołany osobny proces, zazwyczaj zdefiniowany w innym miejscu, za pomocą osobnego schematu blokowego. Blok procesu przypomina blok instrukcji, ale ma pionowe linie po obu stronach:

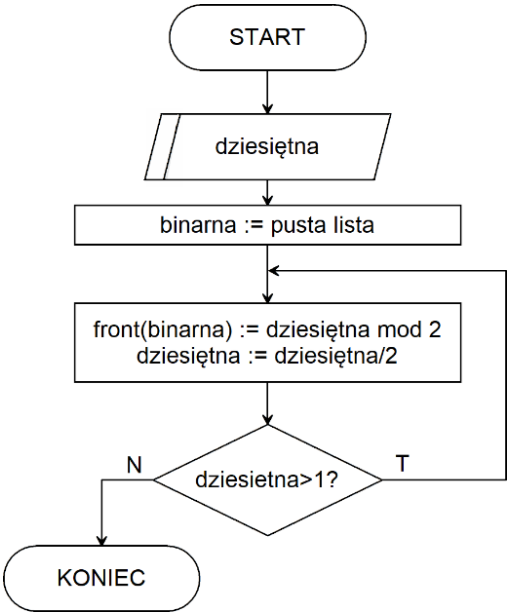
Istnieją również elementy, które są pomocne, gdy schemat blokowy stanie się zbyt duży, by można go było pomieścić na jednej stronie. Elementami tymi są łączniki stronicowe

(o kształcie owalnym) i międzystronicowe (o kształcie pięciokątów). W obu z nich umieszcza się informacje o tym, gdzie kierują czytelnika:



PRZYKŁADOWY SCHEMAT BLOKOWY

Po zapoznaniu się z wyglądem bloków, które tworzą schematy blokowe, możemy opracować schemat blokowy algorytmu, który wcześniej przedstawiliśmy w postaci listy i pseudokodu:



Poświęć chwilę na analizę tego schematu blokowego. Porównaj go z listą kroków i pseudokodem. Czy podane w instrukcji trzy formy zapisu faktycznie są równoważne? Być może jakiegoś elementu gdzieś brakuje lub jest nadmiarowy?

PĘTLE

Wróćmy do algorytmu przedstawionego za pomocą listy kroków. Krok czwarty brzmi: „Jeżeli liczba dziesiętna jest większa niż 1, wróć do kroku 2". Takie kroki nazywamy **pętlami**, gdyż (przy spełnieniu pewnego warunku), zapętlają one algorytm.

Koncept pętli jest w informatyce bardzo istotny. Zawsze, gdy w liście kroków spotkasz się z krokiem „jeżeli (…) to wróć…" czy „Dopóki (…), wykonuj...", masz do czynienia z pętlą. Podobnie sprawa wygląda w przypadku pseudokodu, chociaż w literaturze stosuje się

zazwyczaj angielskie sformułowania: „**for**" („dla danych warunków"), „**while**" („dopóki") i „**if (..) go to**" („jeżeli (…), wróć do…").

Równie łatwo rozpoznać pętlę w schemacie blokowym. Jeżeli strzałka, zamiast do bloku, kieruje nas do innej strzałki, to mamy do czynienia z pętlą.

ZADANIA DO SAMODZIELNEGO WYKONANIA

ZADANIE 1

Algorytm opisany przez pseudokod poniżej, przedstaw za pomocą listy kroków:

```
1. Pobierz a, b
2. Jeżeli a>b: przejdź do kroku 4
3. Zwróć „a jest mniejsze niż b"
4. Zwróć „a jest większe niż b"
```

ZADANIE 2

Algorytm opisany przez pseudokod poniżej, przedstaw za pomocą listy kroków:

```
1. Pobierz x
2. temp := x
3. wynik := 1
4. wynik := wynik * temp
5. temp := temp -1
6. Jeżeli temp >1: Wróć do kroku 4
7. Zwróć wynik
```

ZADANIE 3

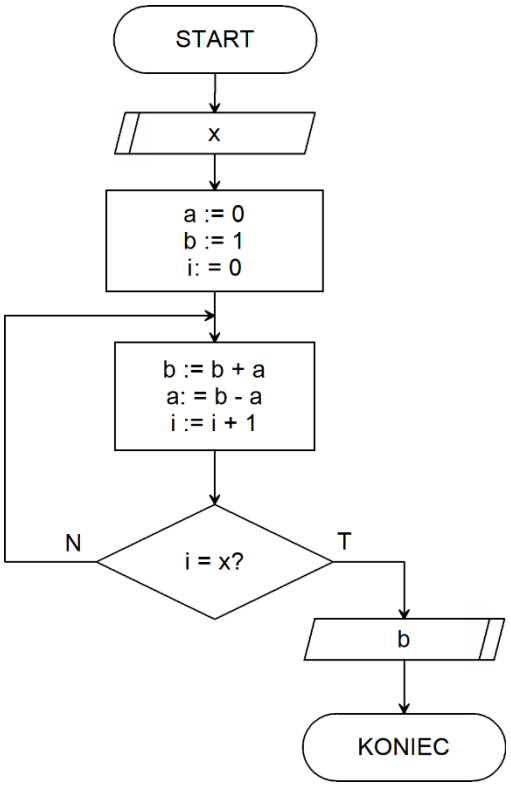
Zakładając, że na wejściu algorytmu z zadania 2, podano wartość 5 (czyli w pierwszym kroku x := 5), przedstaw w postaci tabeli, w jaki sposób będą ulegać zmianie wartości wszystkich zmiennych w algorytmie, aż do jego zakończenia.

ZADANIE 4 Ⓟ

Algorytm z zadania 2 przedstaw za pomocą schematu blokowego.

ZADANIE 5 Ⓟ

Zakładając, że na wejściu algorytmu opisanego schematem blokowym poniżej, podano wartość 6 (czyli w pierwszym kroku x := 6), przedstaw w postaci tabeli, w jaki sposób będą ulegać zmianie wartości wszystkich zmiennych w algorytmie, aż do jego zakończenia.



ZADANIE 6 Ⓟ (NAGRADZANE DWOMA PLUSAMI)

Opracuj algorytm, który pobiera trzy liczby rzeczywiste: a, b i c. Algorytm ma zwracać najbliższe zera miejsce zerowe funkcji $y = ax^3 - 3bx^2 + \frac{c}{2}$, w przedziale (1,100). Jeżeli funkcja nie ma miejsca zerowego w tym przedziale, zwróć komunikat „brak miejsca zerowego". Algorytm przedstaw w formie schematu blokowego.

Autor:	Mgr inż. Paweł Stawarz, 5.10.2021
Korekta:	-