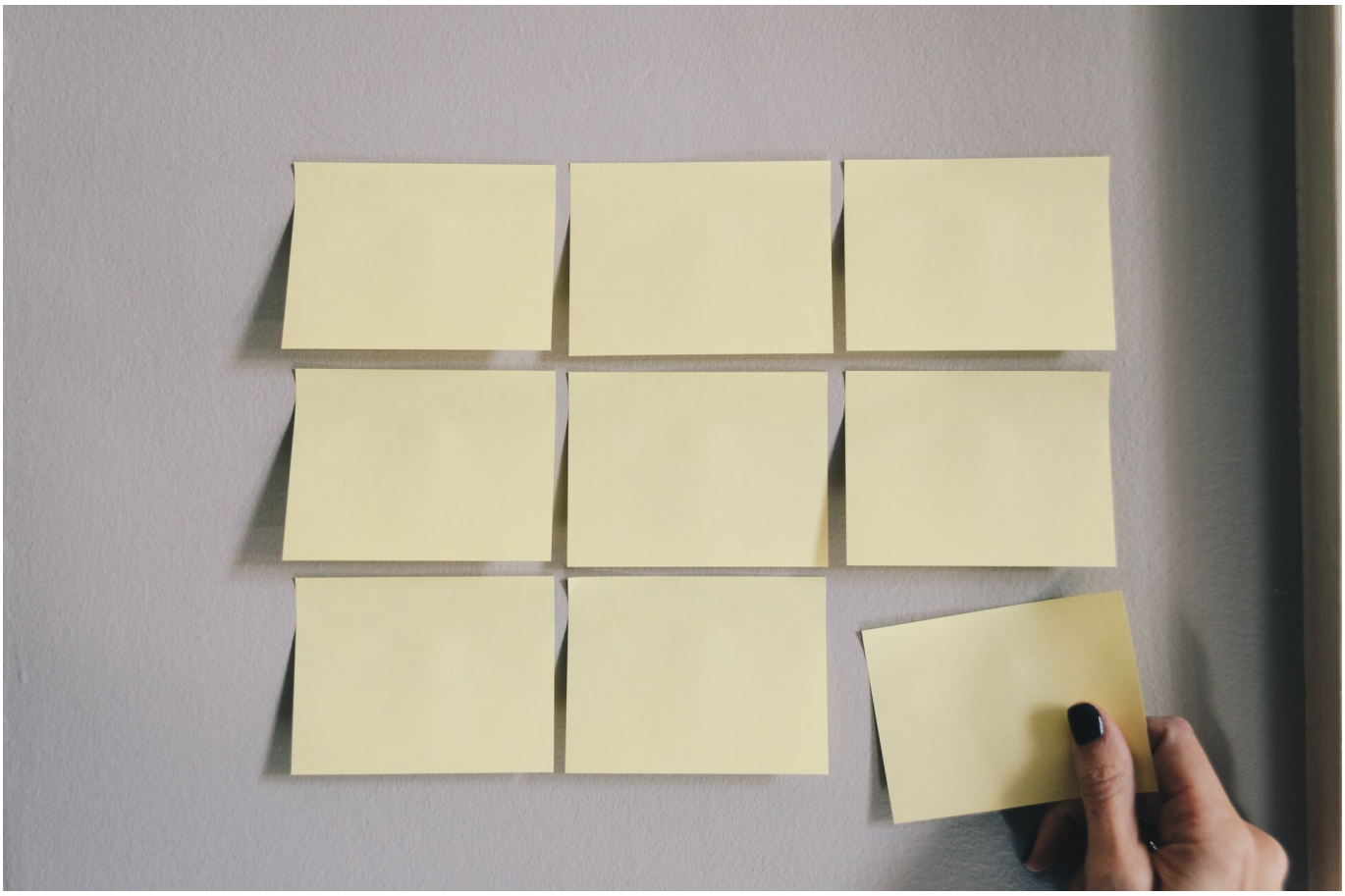


Podstawowe struktury danych

- Wprowadzenie
- Przeczytaj
- Prezentacja multimedialna
- Sprawdź się
- Dla nauczyciela

Źródło: Kelly Sikkema, domena publiczna.

Algorytmy, przetwarzając dane, korzystają z wielu różnych abstrakcyjnych typów danych. Są to abstrakcyjne zbiory wartości i operacji na tych wartościach. Struktury danych są natomiast rzeczywistymi implementacjami pewnych abstrakcyjnych typów danych. W tym e-materiale, używając pojęcia struktury danych, będziemy mieli na myśli abstrakcyjne typy danych.

Struktury danych służą do przechowywania, reprezentacji i modyfikacji danych. Jest wiele różnych rodzajów struktur, więc zawsze warto się zastanowić, którą z nich najlepiej wykorzystać do rozwiązania danego problemu. Odpowiedni wybór struktury danych może zmniejszyć złożoność obliczeniową.

Więcej informacji o poszczególnych rodzajach podstawowych struktur danych znajdziesz w pozostałych e-materiałach z serii:

- Podstawowe struktury danych: tablica,
- Podstawowe struktury danych: rekord,
- Podstawowe struktury danych: lista,
- Podstawowe struktury danych: stos i kolejka.

> **Twoje cele**
>
> - Przeanalizujesz rodzaje podstawowych struktur danych.

- Scharakteryzujesz możliwości i przykładowe zastosowania podstawowych struktur danych.
- Poznasz terminologię związaną z podstawowymi strukturami danych.

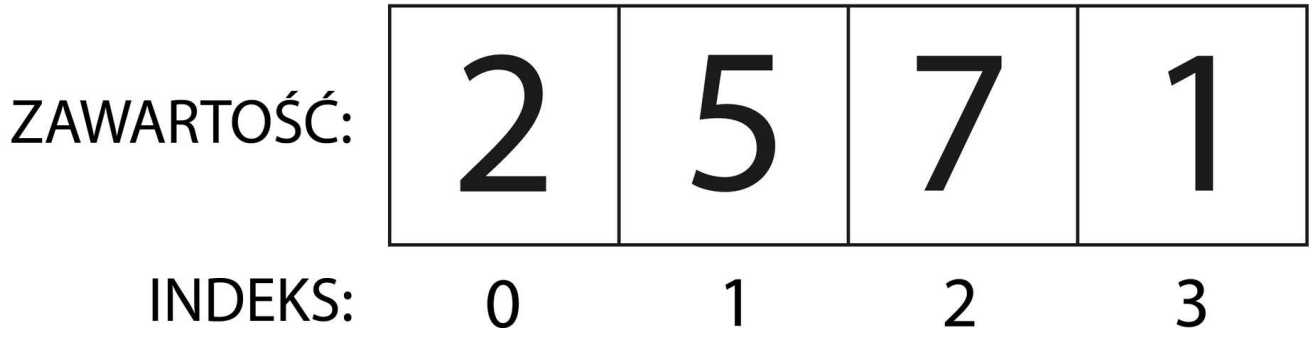
Przeczytaj

Rodzaje podstawowych struktur danych

W tym e-materiale omówimy koncepcyjnie różne rodzaje podstawowych struktur danych. Nie będziemy zagłębiać się w szczegóły implementacyjne, ponieważ mogą się one różnić w zależności od wykorzystywanego języka programowania.

Tablica

Tablica jest strukturą danych, która przechowuje wiele elementów **tego samego typu**. Powiedzmy, że chcemy w naszym programie pracować na milionie zmiennych całkowitych. Deklarowanie każdej z nich osobno zajęłoby ogromną ilość czasu. Na szczęście możemy przy pomocy jednej instrukcji utworzyć strukturę danych, która będzie się składać z miliona komórek, a w każdej z nich będzie przechowywana jedna liczba całkowita.



Źródło: Contentplus.pl Sp. z o.o., licencja: CC BY-SA 3.0.

Wartości w tablicy są przechowywane w pewnym porządku, każda komórka przechowująca wartość ma swój indeks. W zależności od języka programowania pierwszy indeks ma wartość 0 bądź 1. Aby odwołać się do konkretnej komórki, zazwyczaj korzystamy z instrukcji: `nazwa_tablicy[indeks]`.

Przykład użycia tablicy w pseudokodzie (zakładamy, że indeksujemy od 0):

```
1 tab[] ← {1, 2, 5}   // 3 elementowa tablica liczb
2
3 tab[0] ← 3
4 a ← tab[0] + tab[2]
5 wypisz(a)           // wypisze liczbę 8
```

Struktura

Załóżmy, że w naszym programie chcielibyśmy zaimplementować algorytm pracujący na punktach na płaszczyźnie. Będą one reprezentowane w postaci współrzędnych. Używanie zwykłej tablicy jest możliwe, ale niezbyt wygodne. Możemy za to skorzystać z typu danych zwanego strukturą. W jednym obszarze pamięci może przechowywać logicznie powiązane ze sobą dane tych samych lub **różnych** typów.

Struktura umożliwia więc grupowanie logicznie powiązanych ze sobą informacji różnego typu i tym samym opisywanie złożonych obiektów.

Poszczególne składowe struktury (pola) są **etykietowane**, czyli mają swoje unikatowe nazwy, za których pomocą możemy się do nich odwoływać. Dla lepszego zrozumienia przeanalizujmy poniższy pseudokod:

```
Struktura Punkt {
    wsp_x           // piersza składowa
    wsp_y           // druga składowa
}

Punkt a
a.wsp_x ← 1
a.wsp_y ← (a.wsp_x + 3)

wypisz(a.wsp_y)     // wypisze liczbę 4
```

Oczywiście, jeżeli potrzebujemy przechowywać dużą ilość punktów, możemy skorzystać z **tablicy struktur**. Działa ona tak samo jak zwykła tablica, tyle że przechowuje obiekty o typie naszej struktury.

Plik

Kolejną podstawową strukturą do przechowywania danych jest plik. Plik jest typem ciągłej, sekwencyjnej struktury danych o dowolnie dużym rozmiarze. Na plikach są dozwolone następujące podstawowe operacje:

- utworzenie nowego pliku,
- otworzenie istniejącego pliku,
- odczyt danych z otwartego pliku,
- zapis danych do otwartego pliku,
- zamknięcie otwartego pliku,
- usunięcie istniejącego pliku.

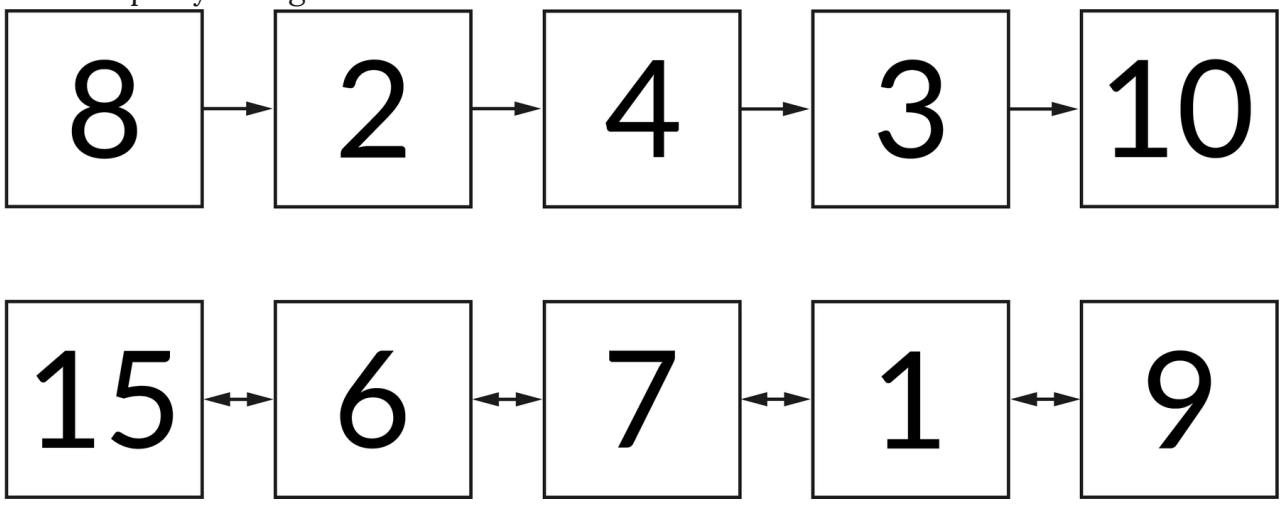
Pliki pozwalają zatem programowi na zapis i odczyt danych z zewnętrznych źródeł. Załóżmy, że napisaliśmy program, który tworzy grafikę, korzystając ze znaków ASCII i chcemy wysłać przykładową ilustrację naszemu znajomemu. Zamiast przesyłać cały program, możemy po prostu zapisać jego wynik do zewnętrznego pliku tekstowego, a następnie go przesłać.

W formie plików możemy także przechowywać dane wejściowe do naszych programów, dzięki temu nie musimy ich ręcznie wprowadzać przy każdym uruchomieniu. Pliki są dla naszego programu pewnego rodzaju zewnętrznym nośnikiem danych.

Lista

Lista jest nieco bardziej zaawansowaną strukturą danych w porównaniu do zwykłych tablic czy struktur. Przede wszystkim jest ona **dynamiczna**, to znaczy, że jej rozmiar może się zmieniać w trakcie działania programu. Jej główną cechą jest to, że przechowywane elementy są ułożone w **liniowym porządku** oraz są ze sobą **wzajemnie powiązane**. Poszczególne elementy listy to struktury przechowujące dane i zawierające pola wskazujące na sąsiednie elementy. Lista zawiera elementy tego samego typu.

Do konkretnych elementów listy nie możemy się odwołać wykorzystując odpowiedni indeks, tak jak mogliśmy to robić w przypadku tablic. Jako, że elementy są ułożone w określonym porządku, jedyną opcją jest przechodzenie do **następnika** lub **poprzednika** obecnie rozpatrywanego elementu.



Źródło: Contentplus.pl Sp. z o.o., licencja: CC BY-SA 3.0.

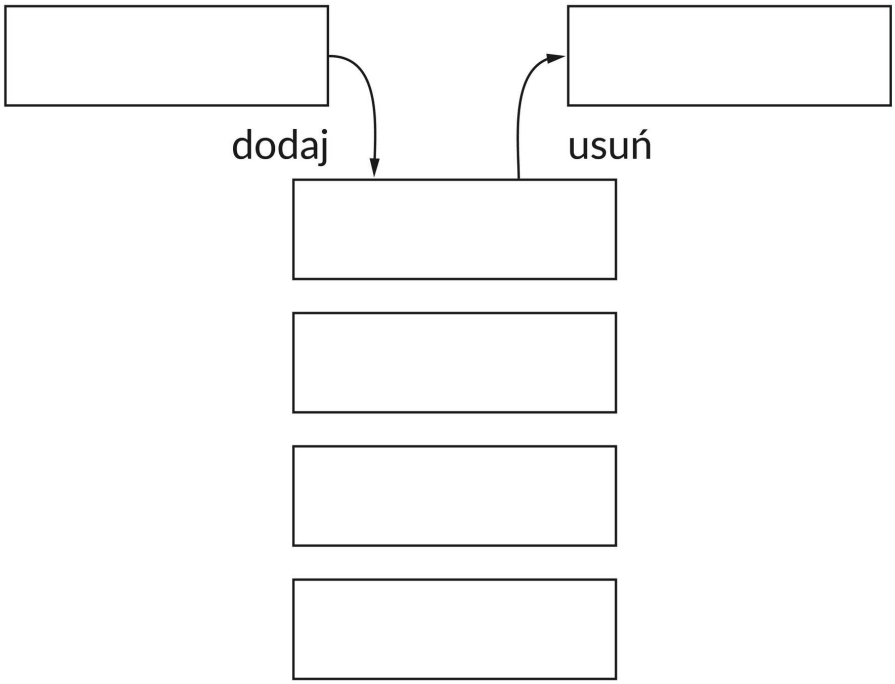
Wyróżnia się dwa rodzaje list: **jednokierunkową** oraz **dwukierunkową**. Różnią się one możliwością przemieszczania się po elementach listy. W liście jednokierunkowej możemy przechodzić jedynie do następnika, bez możliwości powrotu. Natomiast w dwukierunkowej możemy swobodnie przechodzić również do poprzednika.

Na liście możemy wykonywać operację dodawania oraz usuwania elementów. Dodatkowo możemy wygodnie łączyć ze sobą różne listy, oraz je rozdzielać.

Stos

Stos jest strukturą danych składającą się z elementów tego samego typu o dostępie **LIFO** (z ang. *Last In, First Out*). Tak samo jak lista, jest on **dynamiczny**. Jego nazwa jest analogiczna do działania. Mamy dostęp tylko i wyłącznie do elementu na samym **szczycie** stosu. Dodając element, dokładamy go na górę. Analogicznie operacja usunięcia elementu polega na zabraniu go ze szczytu stosu.

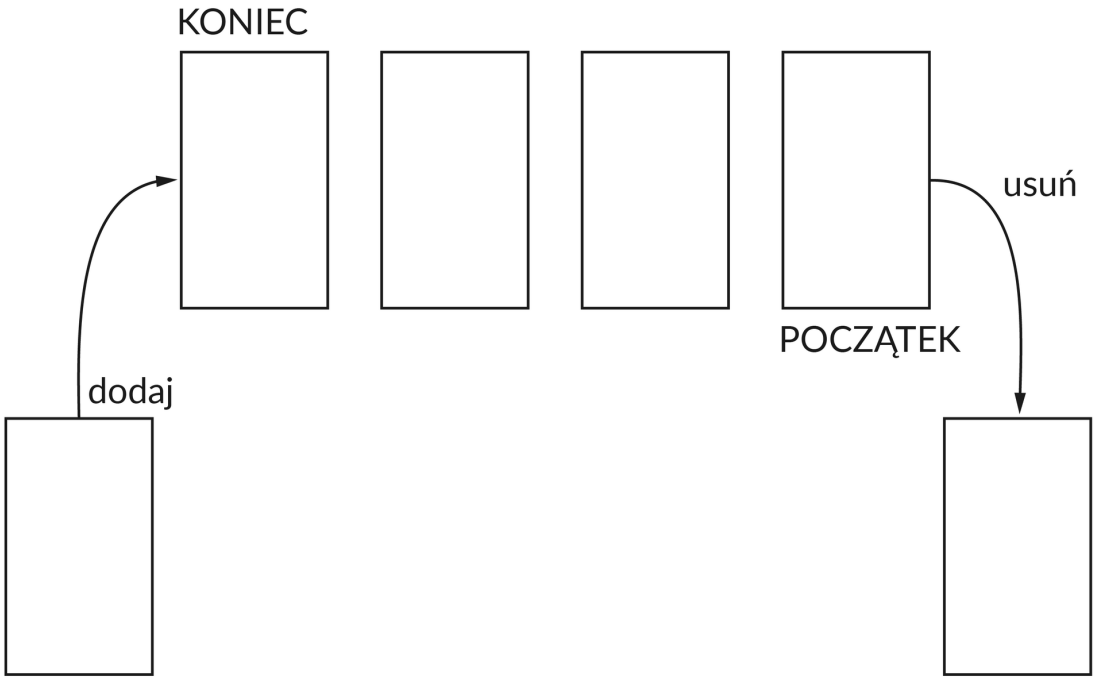
Nawiązuje do sposobu zapisywania i odczytywania elementów.



Kolejka

Kolejka jest za to strukturą danych składającą się z elementów tego samego typu o dostępie **FIFO** (z ang. *First In, First Out*) i również jest **dynamiczna**. Jej działanie jest analogiczne do zwykłej kolejki, z którą możemy się spotkać w sklepie. Tylko pierwsza osoba jest obsługiwana, reszta czeka na swoją kolej. Każda nowa osoba ustawia się na końcu kolejki.

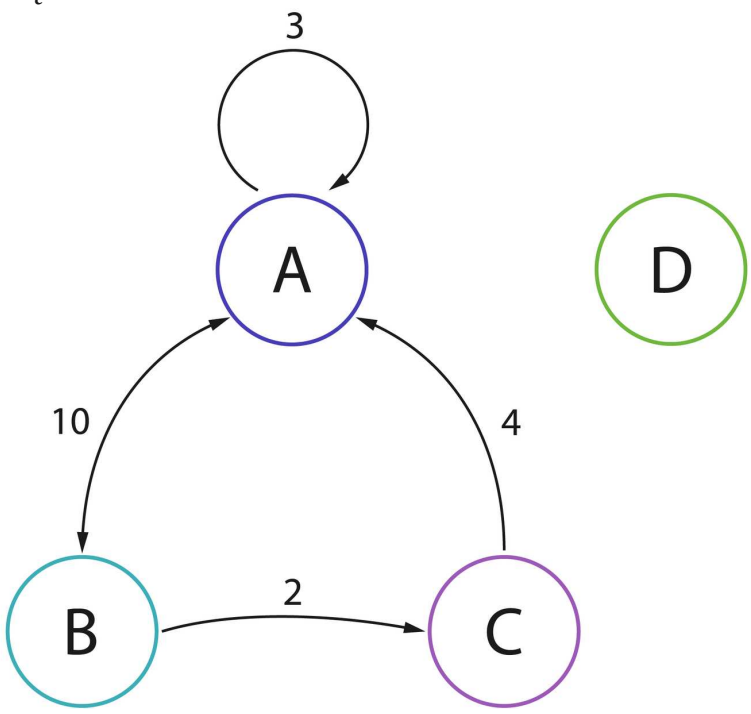
Dokładnie w taki sam sposób działa ta struktura. Operacja usunięcia elementu zabiera go z początku kolejki, natomiast dodawanie umieszcza go na samym końcu.



Źródło: Contentplus.pl Sp. z o.o., licencja: CC BY-SA 3.0.

Graf

Grafy są abstrakcyjną formą reprezentacji danych. Pozwalają one tworzyć, modelować i opisywać wszelkiego rodzaju sieci, w których występują skomplikowane zależności pomiędzy składnikami. Więcej na ich temat znajdziesz w e-materiałach dotyczących teorii grafów. Podstawowe informacje znajdują się w e-materiale Wprowadzenie do teorii grafów. Graf składa się z dwóch podstawowych rodzajów elementów, a mianowicie **wierzchołków** oraz **krawędzi**. Krawędzie reprezentują połączenie pomiędzy dwoma wierzchołkami, mogą one mieć swoją **wagę**, czy **kierunek**. Między wierzchołkami możemy się poruszać zgodnie z kierunkami łączących krawędzi.

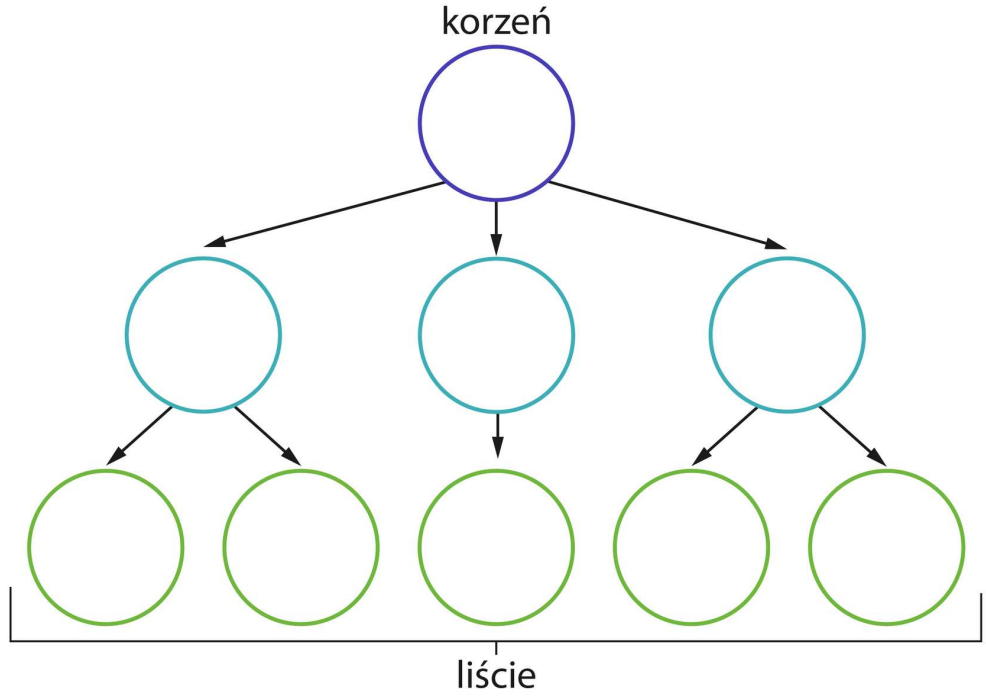


Źródło: Contentplus.pl Sp. z o.o., licencja: CC BY-SA 3.0.

Rozróżniamy różne rodzaje grafów m.in. **skierowane**, **nieskierowane** czy **ważone**. Przykładem grafu mogą być miasta wraz z łączącymi je drogami. Wagami krawędzi mogą być wtedy długości poszczególnych dróg.

Drzewo

Drzewo jest specjalnym rodzajem grafu. Jego najważniejszą cechą jest to, że istnieje dokładnie jedna droga między jego dowolnymi dwoma wierzchołkami. Dodatkowo wierzchołki drzewa, które nazywamy **węzłami**, są uporządkowane w pewnej hierarchii. Każdy węzeł z wyjątkiem korzenia i liści ma swojego **ojca**, czyli element znajdujący się nad nim oraz swoje **dzieci**, czyli elementy znajdujące się pod nim.



Źródło: Contentplus.pl Sp. z o.o., licencja: CC BY-SA 3.0.

W strukturze drzewa wyróżnia się jeden, charakterystyczny element nazywany **korzeniem**. Jego cechą jest to, że nie posiada on ojca. Dodatkowo węzły, które nie posiadają dzieci noszą nazwę **liści**.

Słownik

abstrakcyjny typ danych

abstrakcyjny zbiór wartości i operacji na tych wartościach; stanowi szablon implementacyjny dla struktur danych

ASCII

pierwotnie siedmiobitowy system kodowania znaków (współcześnie rozszerzony do ośmiu bitów); w oryginalnej wersji kodom z zakresu 0–127 przyporządkowano litery alfabetu łacińskiego, cyfry, znaki przestankowe oraz polecenia sterujące

paradygmat programowania

> wzorzec klasyfikujący języki programowania na podstawie ich własności; języki programowania mogą być sklasyfikowane do wielu różnych paradygmatów programowania
> **struktura danych**
>
> rzeczywista implementacja abstrakcyjnego typu danych

Prezentacja multimedialna

Polecenie 1

Zapoznaj się z prezentacją dotyczącą programowania strukturalnego i wykonaj polecenie.

1

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/PMCj8uedF

Idea programowania strukturalnego opiera się na trzech głównych elementach kontrolnych przepływu programu. Pierwszym z nich jest sekwencja, czyli wykonywanie ciągu kolejnych instrukcji. Jest to zaprezentowane w poniższym pseudokodzie. Instrukcje są wykonywane jedna po drugiej.

```
a ← 5
b ← a + 3
wypisz(a + b)
```

2

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/PMCj8uedF

Kolejnym głównym elementem programowania strukturalnego jest tzw. wybór. Polega on na tym, że w zależności od spełnienia określonych warunków wykonywana jest odpowiednia instrukcja.

```
n ← losuj()

Jeżeli n < 10 wykonuj:
    wypisz(n)
```

```
Albo jeżeli n < 20
wykonuj:
    wypisz(2 * n)
W przeciwnym razie:
    wypisz(n / 2)
```

3

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/PMCj8uedF

Ostatnim elementem jest iteracja, czyli wykonywanie określonego bloku instrukcji, dopóki spełniony jest pewien warunek.

```
i ← 0

Dopóki i < 10 wykonuj:
    wypisz(i)
    i ← i + 1
```

4

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/PMCj8uedF

Dodatkowo programowanie strukturalne jest paradygmatem programowania, który zakłada rozdzielenie funkcjonalności programu na odpowiednie moduły (podprogramy). Przykładowo jest to realizowane poprzez dzielenie kodu programu na funkcje i bloki.

```
Funkcja wypisz_ciag(n):
    i ← 0

    Dopóki i < n wykonuj:
        wypisz(i, " ")
        i ← i + 1
```

```
7
8       wypisz("\n")
```

```
1 wypisz_ciag(5) // program
  wypisze: 0 1 2 3 4
2 wypisz_ciag(3) // program
  wypisze: 0 1 2
```

5

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/PMCj8uedF

Zatem w programowaniu strukturalnym skupiamy się na wyglądzie kodu. W końcu dla komputera nie ma to większego znaczenia, ponieważ zawsze będzie działał w ten sam sposób.

Różnica jest za to widoczna po stronie programisty. Zależnie od wybranej metody programowania inaczej będzie się czytać kod i rozbudowywać program.

6

Materiał audio dostępny pod adresem:

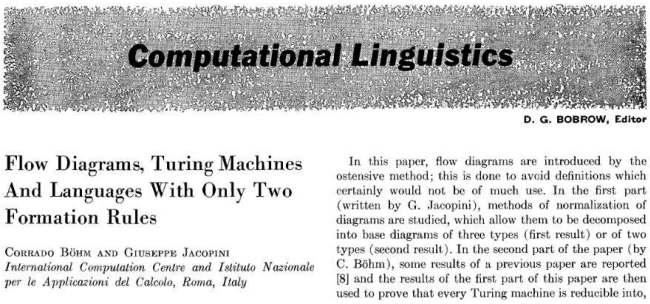
https://zpe.gov.pl/b/PMCj8uedF

Na samym początku, zanim pojawiła się idea programowania strukturalnego, kontrola przepływu w programie odbywała się głównie przy pomocy warunkowych i bezwarunkowych instrukcji skoku.

```
1             n ← 14
2 ODEJMIJ:    n ← n - 1
3             Jeżeli n < 10
  skocz do KONIEC
4             wypisz(n, " ")
```

```
5               skocz do
ODEJMIJ
6 KONIEC:
7
8 // program wypisze: 13 12
11 10
```

7



Computational Linguistics

D. G. BOBROW, Editor

Flow Diagrams, Turing Machines And Languages With Only Two Formation Rules

CORRADO BÖHM AND GIUSEPPE JACOPINI
International Computation Centre and Istituto Nazionale per le Applicazioni del Calcolo, Roma, Italy

In this paper, flow diagrams are introduced by the ostensive method; this is done to avoid definitions which certainly would not be of much use. In the first part (written by G. Jacopini), methods of normalization of diagrams are studied, which allow them to be decomposed into base diagrams of three types (first result) or of two types (second result). In the second part of the paper (by C. Böhm), some results of a previous paper are reported [8] and the results of the first part of this paper are then used to prove that every Turing machine is reducible into,

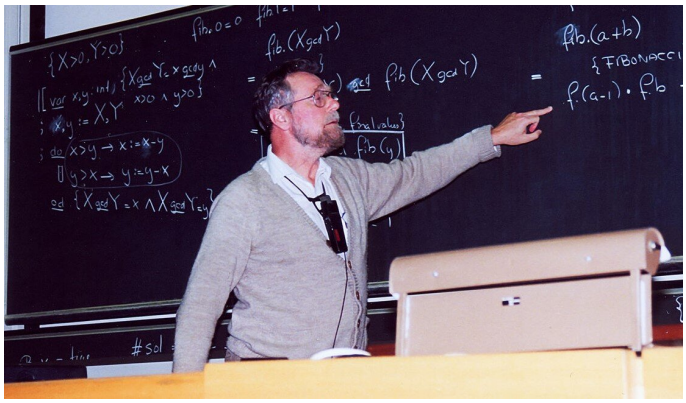
Źródło: ACM DL, dl.acm.org, dostęp 08.05.2024, licencja: CC 0.

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/PMCj8uedF

Koncepcję programowania strukturalnego oficjalnie sformalizowali dopiero w 1966 roku Corrado Böhm i Giuseppe Jacopini. Pokazali oni, że dowolny program zawierający instrukcje skoku, może zostać zapisany przy użyciu sekwencji, wyborów oraz iteracji.

8



Źródło: Andreas F. Borchert, licencja: CC BY-SA 4.0 DEED.

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/PMCj8uedF

Na przełomie lat $60.$ i $70.$ XX wieku powstawało wiele prac dotyczących tego jak dobrze tworzyć i analizować programy strukturalne. Jednym z autorów był holenderski naukowiec i pionier informatyki Edsger Wybe Dijkstra.

9

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/PMCj8uedF

Dijkstra w ogromnym stopniu przyczynił się do ograniczenia programowania opartego na instrukcjach skoku. Otwarcie głosił, że instrukcje te powinny zostać usunięte ze wszystkich wysokopoziomowych języków programowania, ponieważ są one zbędne. Uważał, że programy, które je wykorzystują trudno analizować oraz utrzymywać.

10

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/PMCj8uedF

Dziś możemy zaobserwować zanik użycia instrukcji skokowych w różnych językach programowania. Dodatkowo wykracza się poza założenia programowania strukturalnego stosując w programach obsługę wyjątków, albo instrukcje wczesnego wyjścia. Zostało to zaprezentowane w poniższym pseudokodzie programu, który losuje i wypisuje sto liczb, ale jeżeli wylosuje liczbę większą od stu, to kończy swoje działanie.

```
1 i ← 0
2
```

```
Dopóki i < 100 wykonuj:
    n ← losuj()

    Jeżeli n > 100 wykonuj:
        Zakończ program

    wypisz(n)
    i ← i + 1
```

Polecenie 2

Przekształć pseudokod programu znajdującego się na ostatnim slajdzie prezentacji tak, aby korzystał z instrukcji skoku.

```
1
```

Polecenie 3

Zapisz notatkę, w której podsumujesz najważniejsze informacje przedstawione w prezentacji multimedialnej.

Sprawdź się

Pokaż ćwiczenia:

Ćwiczenie 1

Dokończ zdanie.

Tablica przechowuje wiele wartości o...

- ◯ różnym typie.
- ◯ tym samym typie.

Ćwiczenie 2

Dokończ zdanie.

Elementy listy są...

- ◯ różnych typów.
- ◯ wzajemnie powiązane.
- ◯ uporządkowane względem ich wartości.
- ◯ rozproszone względem siebie.

Ćwiczenie 3

Zaznacz wszystkie rodzaje znaków wchodzące w zestaw znaków ASCII.

- [ ] Znaki przestankowe.
- [ ] Litery alfabetu polskiego.
- [ ] Emotikony.
- [ ] Polecenia sterujące.
- [ ] Litery alfabetu łacińskiego języka angielskiego.
- [ ] Cyfry.

Ćwiczenie 4

Dokończ zdanie.

Poszczególne składowe struktury są...

- ( ) tagowane.
- ( ) uporządkowane liniowo.
- ( ) uporządkowane hierarchicznie.
- ( ) etykietowane.

Ćwiczenie 5

Dokończ zdanie.

Jeżeli struktura danych jest dynamiczna to...

- ◯ jej elementy są uporządkowane hierarchicznie.
- ◯ jej elementy są uporządkowane liniowo.
- ◯ jej rozmiar może ulec zmianie w trakcie działania programu.
- ◯ jej rozmiar nie może ulec zmianie w trakcie działania programu.

Ćwiczenie 6

Wstaw elementy do odpowiednich grup.

lista

graf

tablica

nieskierowany | skierowany | krawędź | poprzednik | indeks | liniowy porządek | następnik | rozmiar ustalony z góry (statyczny) | dwukierunkowa | ważony

Ćwiczenie 7

Uzupełnij tekst.

Stos i kolejka to struktury danych składające się z elementów [ ] typu. Kolejka jest strukturą o dostępie [ ], natomiast stos jest strukturą o dostępie [ ]. Obie struktury charakteryzują się tym, że są [ ]. Nowy element jest umieszczany na [ ] stosu. Z kolejki usuwany jest pierwszy (początkowy) element, natomiast nowy element jest dodawany na [ ].

końca | koniec | szczycie | FIFO | dnie | statyczne | LIFO | tego samego | dynamiczne | początek | różnego

Ćwiczenie 8

Wyjaśnij różnicę między listą jednokierunkową a dwukierunkową.

Dla nauczyciela

Autor: Zespół autorski Contentplus.pl sp. z o.o.
Przedmiot: Informatyka

Temat: Podstawowe struktury danych

Grupa docelowa:

Liceum ogólnokształcące i technikum, liceum ogólnokształcące, technikum, zakres rozszerzony

Podstawa programowa:

Cele kształcenia – wymagania ogólne

I. Rozumienie, analizowanie i rozwiązywanie problemów na bazie logicznego i abstrakcyjnego myślenia, myślenia algorytmicznego i sposobów reprezentowania informacji.

Treści nauczania – wymagania szczegółowe

I. Rozumienie, analizowanie i rozwiązywanie problemów.

Zakres rozszerzony. Uczeń spełnia wymagania określone dla zakresu podstawowego, a ponadto:

2) do realizacji rozwiązania problemu dobiera odpowiednią metodę lub technikę algorytmiczną i struktury danych;

I + II. Zakres rozszerzony. Uczeń spełnia wymagania określone dla zakresu podstawowego, a ponadto:

3) objaśnia, a także porównuje podstawowe metody i techniki algorytmiczne oraz struktury danych, wykorzystując przy tym przykłady problemów i algorytmów, w szczególności:

i) struktury dynamiczne: stos, kolejka, lista (do realizacji algorytmu: ONP, symulacji problemu Flawiusza, sortowania leksykograficznego),

j) grafy (do przedstawiania abstrakcyjnego modelu sytuacji problemowych).

Kształtowane kompetencje kluczowe:

- kompetencje cyfrowe;

- kompetencje osobiste, społeczne i w zakresie umiejętności uczenia się;
- kompetencje matematyczne oraz kompetencje w zakresie nauk przyrodniczych, technologii i inżynierii.

Cele operacyjne (językiem ucznia):

- Przeanalizujesz rodzaje podstawowych struktur danych.
- Scharakteryzujesz możliwości i przykładowe zastosowania podstawowych struktur danych.
- Poznasz terminologię związaną z podstawowymi strukturami danych.

Strategie nauczania:

- konstruktywizm;
- konektywizm.

Metody i techniki nauczania:

- dyskusja;
- rozmowa nauczająca z wykorzystaniem multimedium i ćwiczeń interaktywnych;
- metody aktywizujące.

Formy pracy:

- praca indywidualna;
- praca w parach;
- praca w grupach;
- praca całego zespołu klasowego.

Środki dydaktyczne:

- komputery z głośnikami, słuchawkami i dostępem do internetu;
- zasoby multimedialne zawarte w e-materiale;
- tablica interaktywna/tablica, pisak/kreda.

Przebieg lekcji

Przed lekcją:

1. **Przygotowanie do zajęć.** Nauczyciel loguje się na platformie i udostępnia e-materiał: „Podstawowe struktury danych". Uczniowie mają zapoznać się z treściami w sekcji „Przeczytaj".

Faza wstępna:

1. Nauczyciel wyświetla i odczytuje temat lekcji oraz cele zajęć. Prosi uczniów o sformułowanie kryteriów sukcesu.

2. **Rozpoznanie wiedzy uczniów.** Nauczyciel wyświetla na tablicy pytania zawarte w sekcji „Wprowadzenie":
    - jak nazywamy obiekt, który przechowuje wiele zmiennych o tym samym typie?
    - czym różnią się dwa rodzaje list – jednokierunkowa oraz dwukierunkowa?
    - czy stos jest strukturą typu LIFO czy FIFO?
    - wymień podstawowe elementy grafu.

    Chętni uczniowie udzielają na nie odpowiedzi.

Faza realizacyjna:

1. **Praca z tekstem.** Uczniowie analizują treści z sekcji „Przeczytaj" wyświetlone na tablicy.
2. **Praca z multimedium.** Nauczyciel wyświetla zawartość sekcji „Prezentacja multimedialna". Uczniowie zapoznają się z treścią prezentacji, a następnie wykonują polecenie 2. Nauczyciel sprawdza poprawność wykonania zadania.
3. **Ćwiczenie umiejętności.** Uczniowie wykonują indywidualnie ćwiczenia nr 1–5. Po wykonaniu każdego z nich następuje omówienie rozwiązania przez nauczyciela.

Faza podsumowująca:

1. Nauczyciel ponownie wyświetla na tablicy temat lekcji zawarty w sekcji „Wprowadzenie" i inicjuje krótką rozmowę na temat zrealizowanych celów (czego uczniowie się nauczyli).
2. Na koniec zajęć nauczyciel prosi uczniów o rozwinięcie zdania: „Na dzisiejszych zajęciach nauczyłam/łem się jak…".

Praca domowa:

1. Jakiej struktury użyjesz, by przygotować katalog produktów oferowanych w kawiarni? Uzasadnij swój wybór i przygotuj przykładową strukturę.
2. Uczniowie wykonują ćwiczenie 6–8 z sekcji „Sprawdź się".

Wskazówki metodyczne:

- Treści w sekcji „Prezentacja multimedialna" można wykorzystać na lekcji jako podsumowanie i utrwalenie wiedzy uczniów.