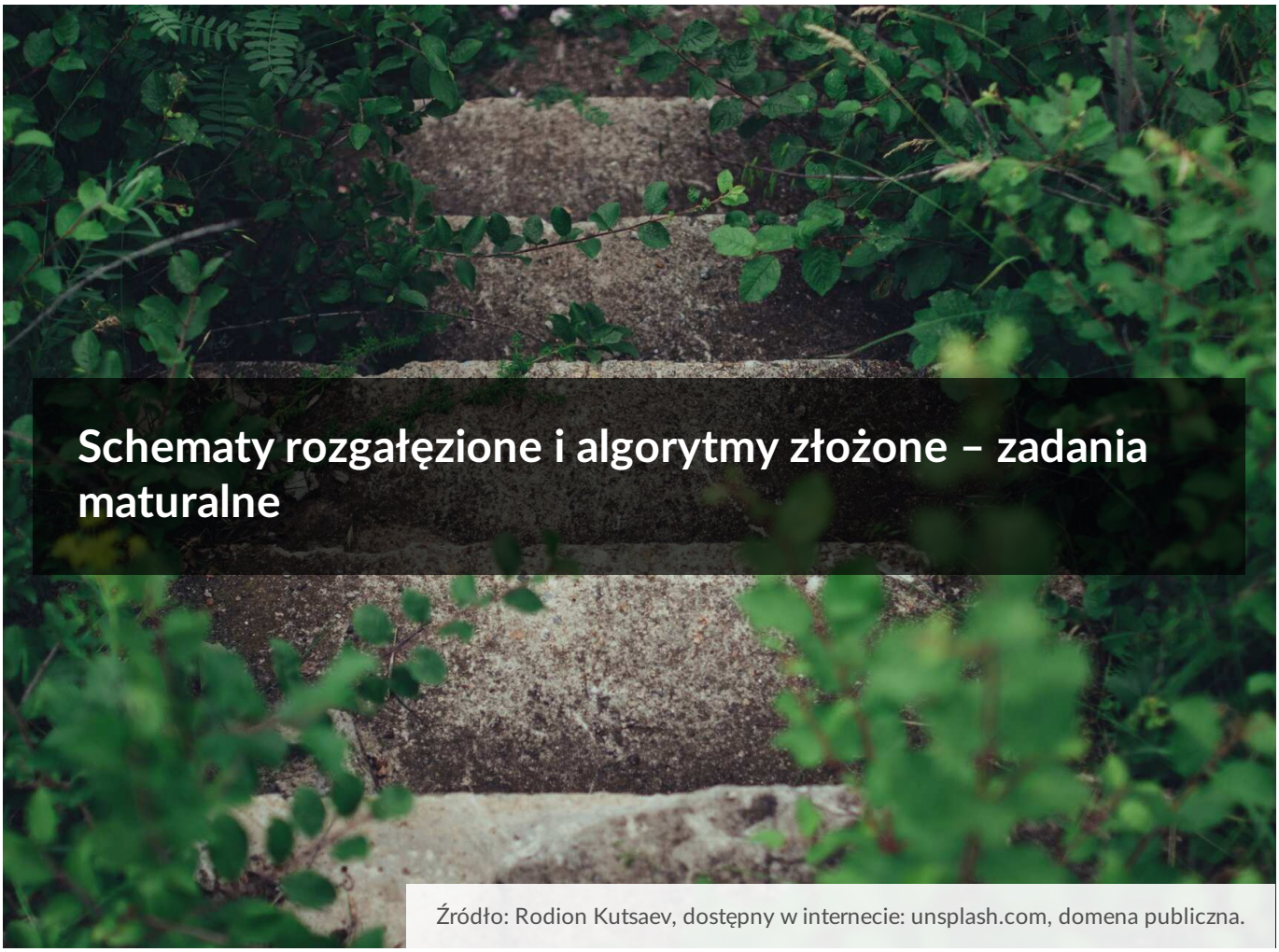
Schematy rozgałęzione i algorytmy złożone – zadania maturalne



Schematy rozgałęzione i algorytmy złożone – zadania maturalne

Źródło: Rodion Kutsaev, dostępny w internecie: unsplash.com, domena publiczna.

W tym e-materiale przyjrzymy się zadaniom poświęconym algorytmom złożonym.

W zadaniach sprawdzimy znajomość dwóch sposobów zapisu algorytmów – pseudokodu oraz implementacji w wybranym języku programowania.

Podobne zagadnienia znajdziesz w e-materiałach:

- Schematy rozgałęzione i algorytmy złożone,
- Proste algorytmy rozgałęzione,
- Algorytmy geometryczne,
- Czy algorytm jest poprawny?.

Twoje cele

- Utrwalisz wiedzę dotyczącą schematów rozgałęzionych i algorytmów złożonych.
- Rozwiążesz kilka zadań typu maturalnego, dotyczących algorytmów rozgałęzionych, wykorzystując różne typy notacji algorytmów.
- Zapiszesz algorytmy rozgałęzione za pomocą pseudokodu i wybranego języka programowania.

Przeczytaj

Zadanie 1

Zapisz w wybranej przez siebie notacji (w postaci pseudokodu lub w wybranym języku programowania) algorytm, który dla danej liczby `x` obliczy sumę jej parzystych czynników pierwszych. Maksymalna liczba punktów za rozwiązanie zadania zostanie przyznana, jeśli algorytm wykona $O(\log\ x)$ operacji arytmetycznych, wymienionych w uwadze.

> **Uwaga!**
>
> W zapisie algorytmu możesz wykorzystać tylko operacje arytmetyczne: dodawanie, odejmowanie, mnożenie, dzielenie, dzielenie całkowite, resztę z dzielenia oraz porównywanie liczb, instrukcje sterujące i przypisania do zmiennych lub samodzielnie napisane funkcje, zawierające wcześniej wymienione operacje.

Specyfikacja problemu:

Dane:

- `x` – liczba naturalna, której sumę parzystych czynników pierwszych należy obliczyć

Wynik:

- `wynik` – suma parzystych czynników pierwszych liczby `x`

Rozwiązanie

> Rozwiązanie zadania przedstawimy w postaci pseudokodu, ponieważ na egzaminie maturalnym można korzystać z pseudokodu lub z wybranego języka programowania: C++, Java lub Python.

Jedynym parzystym czynnikiem pierwszym danej liczby może być 2, ponieważ 2 to jedyna parzysta liczba pierwsza.

Jeżeli liczba `x` jest nieparzysta, to nie ma żadnych parzystych czynników pierwszych, zwracamy więc 0. Jeżeli liczba `x` jest parzysta, to dzielimy liczbę przez 2, aż stanie się niepodzielna przez 2. Na koniec zwracamy wynik, czyli liczbę 2 pomnożoną przez liczbę jej wystąpień jako czynnika pierwszego danej liczby.

```
1 funkcja sumaPierwszych(x):
2     jeżeli x mod 2 = 1:
3         zwróć 0
4     suma = 0
```

```
    dopóki x mod 2 = 0:
        suma ← suma + 2
        x ← x div 2
    zwróć suma
```

Zauważenie, że jedynym parzystym czynnikiem pierwszym danej liczby może być 2, ułatwia rozwiązanie zadania.

Za przedstawione rozwiązanie dostaniemy maksymalną liczbę punktów, ponieważ jego złożoność wynosi $O(\log\ x)$.

> **Praca domowa**
>
> Przedstaw rozwiązanie zadania w postaci programu napisanego w wybranym języku (C++, Java lub Python).

Słownik

> **algorytm liniowy**
>
> algorytm zwany również sekwencyjnym; kroki algorytmu następują po sobie; wykonanie jednego kroku powoduje przejście bezpośrednio do kolejnego; nie określa się w nim żadnych warunków
>
> **algorytm rozgałęziony**
>
> algorytm warunkowy; algorytm, w którym może występować kilka ciągów działań; po spełnieniu lub niespełnieniu danego warunku następuje wybór jednego z ciągów działań
>
> **pseudokod**
>
> jedna z metod zapisywania algorytmów; pseudokod jest zapisem uniwersalnym, w przypadku którego nie używa się słów kluczowych języków programowania; nie istnieją sformalizowane reguły zapisu pseudokodu – ma być on przejrzysty i zrozumiały, a do jego odczytania nie może być wymagana znajomość języków programowania

Schemat interaktywny

Zadanie 2.

W pliku `liczby.txt` znajduje się 100 wierszy, każdy z nich zawiera parę liczb naturalnych.

Plik o rozmiarze 942.00 B w języku polskim

Napisz program, który dla pliku `liczby.txt` obliczy, ile silni liczb naturalnych znajduje się pomiędzy liczbami podanymi w danym wierszu. Odpowiedzi zapisz do pliku `wynik.txt`.

> **Przykład 1**
>
> Dla liczb 12 i 122 poprawnym wynikiem będzie 2. Między tymi liczbami znajdują się liczby $4!$ i $5!$.
>
> Dla 2 i 720 poprawnym wynikiem jest 5. Między tymi liczbami znajduje się $2!$, $3!$, $4!$, $5!$ i $6!$.

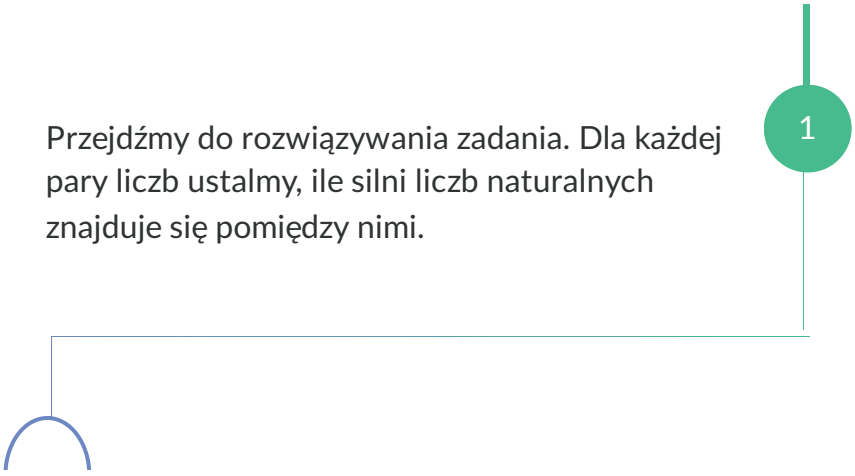
Do oceny oddajesz:

- plik `wynik.txt` zawierający odpowiedź do zadania (100 wierszy, każdy z nich zawiera liczbę naturalną, która oznacza, ile silni liczb naturalnych znajduje się pomiędzy liczbami podanymi w odpowiednim wierszu w pliku `liczby.txt`),
- plik(i) z komputerową realizacją zadania (kodem programu).

> **Polecenie 1**
>
> Przeanalizuj prezentację przedstawiającą rozwiązanie zadania, a następnie wykonaj polecenia.

Rozwiązanie

> Rozwiązanie zadania przedstawimy w postaci pseudokodu, ponieważ na egzaminie maturalnym można korzystać z wybranego języka programowania: C++, Java lub Python.



2

Zaczynamy od wczytania danych z pliku.

```
dla i = 0, 1, ..., 99
wykonuj:
    liczby[i][0] ← wczytaj pierwszą liczbę z wiersza z pliku "liczby.txt"
    liczby[i][1] ← wczytaj drugą liczbę z wiersza z pliku "liczby.txt"
    przejdź do następnego wiersza w pliku "liczby.txt"
```

3

Następnie tworzymy główną pętlę programu, która wykona się 100 razy, ponieważ tyle mamy par liczb.

```
dla i = 0, 1, ..., 99
wykonuj:
    liczby[i][0] ← wczytaj pierwszą liczbę z wiersza z pliku "liczby.txt"
    liczby[i][1] ← wczytaj drugą liczbę z wiersza z pliku "liczby.txt"
    przejdź do następnego wiersza w pliku "liczby.txt"

dla i = 0, 1, ..., 99
wykonuj:
```

4

Inicjujemy w niej zmienne: `silnia` zawierającą obliczaną silnię oraz `x` oznaczającą liczbę, której silnię w danym momencie bierzemy pod uwagę.

```
dla i = 0, 1, ..., 99
wykonuj:
    liczby[i][0] ← wczytaj pierwszą liczbę z wiersza z pliku "liczby.txt"
    liczby[i][1] ← wczytaj drugą liczbę z wiersza z pliku "liczby.txt"
    przejdź do następnego wiersza w pliku "liczby.txt"

dla i = 0, 1, ..., 99
wykonuj:
    silnia ← 1
    x ← 1
```

5

Następnie obliczamy najmniejszą silnię, która mieści się w badanym przedziale. W tym celu tworzymy pętlę `dopóki`.

```
dla i = 0, 1, ..., 99
wykonuj:
    liczby[i][0] ← wczytaj pierwszą liczbę z wiersza z pliku "liczby.txt"
    liczby[i][1] ← wczytaj drugą liczbę z wiersza z pliku "liczby.txt"
    przejdź do następnego wiersza w pliku "liczby.txt"

dla i = 0, 1, ..., 99
wykonuj:
    silnia ← 1
    x ← 1
    dopóki silnia < liczby[i][0] wykonuj:
```

6

W niej na bieżąco aktualizujemy wartość zmiennych `silnia` oraz `x`.

```
dla i = 0, 1, ..., 99
wykonuj:
    liczby[i][0] ← wczytaj pierwszą liczbę z wiersza z pliku "liczby.txt"
    liczby[i][1] ← wczytaj drugą liczbę z wiersza z pliku "liczby.txt"
    przejdź do następnego wiersza w pliku "liczby.txt"

dla i = 0, 1, ..., 99
wykonuj:
    silnia ← 1
    x ← 1
    dopóki silnia < liczby[i][0] wykonuj:
        x ← x + 1
        silnia ← silnia * x
```

7

Następnie inicjujemy zmienną `wynik`, którą użyjemy do zliczania, ile silni mieści się w danym przedziale.

```
dla i = 0, 1, ..., 99
wykonuj:
    liczby[i][0] ← wczytaj pierwszą liczbę z wiersza z pliku "liczby.txt"
    liczby[i][1] ← wczytaj drugą liczbę z wiersza z pliku "liczby.txt"
```

```
    przejdź do następnego wiersza w pliku "liczby.txt"

dla i = 0, 1, ..., 99 wykonuj:
    silnia ← 1
    x ← 1
    dopóki silnia < liczby[i][0] wykonuj:
        x ← x + 1
        silnia ← silnia * x

    wynik ← 0
```

8

Tworzymy kolejną pętlę `dopóki`, która będzie się wykonywać dopóty, dopóki wyjdziemy poza oznaczony przedział.

```
dla i = 0, 1, ..., 99 wykonuj:
    liczby[i][0] ← wczytaj pierwszą liczbę z wiersza z pliku "liczby.txt"
    liczby[i][1] ← wczytaj drugą liczbę z wiersza z pliku "liczby.txt"
    przejdź do następnego wiersza w pliku "liczby.txt"

dla i = 0, 1, ..., 99 wykonuj:
    silnia ← 1
    x ← 1
    dopóki silnia < liczby[i][0] wykonuj:
        x ← x + 1
```

```
        silnia ← silnia *
x

    wynik ← 0
    dopóki silnia <=
liczby[i][1] wykonuj:
```

9

W tej pętli po kolei aktualizujemy zmienne, obliczając silnię i inkrementując `wynik`.

```
dla i = 0, 1, ..., 99
wykonuj:
    liczby[i][0] ← wczytaj
pierwszą liczbę z wiersza
z pliku "liczby.txt"
    liczby[i][1] ← wczytaj
drugą liczbę z wiersza z
pliku "liczby.txt"
    przejdź do następnego
wiersza w pliku
"liczby.txt"

dla i = 0, 1, ..., 99
wykonuj:
    silnia ← 1
    x ← 1
    dopóki silnia <
liczby[i][0] wykonuj:
        x ← x + 1
        silnia ← silnia *
x

    wynik ← 0
    dopóki silnia <=
liczby[i][1] wykonuj:
        wynik ← wynik + 1
        x ← x + 1
        silnia ← silnia *
x
```

10

Na koniec wpisujemy wynik w nowej linii w pliku wynikowym i przechodzimy do kolejnej iteracji głównej pętli.

```
dla i = 0, 1, ..., 99 wykonuj:
    liczby[i][0] ← wczytaj pierwszą liczbę z wiersza z pliku "liczby.txt"
    liczby[i][1] ← wczytaj drugą liczbę z wiersza z pliku "liczby.txt"
    przejdź do następnego wiersza w pliku "liczby.txt"

dla i = 0, 1, ..., 99 wykonuj:
    silnia ← 1
    x ← 1
    dopóki silnia < liczby[i][0] wykonuj:
        x ← x + 1
        silnia ← silnia * x

    wynik ← 0
    dopóki silnia <= liczby[i][1] wykonuj:
        wynik ← wynik + 1
        x ← x + 1
        silnia ← silnia * x

    wpisz wynik do nowej linii w pliku "wynik.txt"
```

Odpowiedź do zadania

Odpowiedź do zadania dla danych z pliku `liczby.txt`:

Plik o rozmiarze 298.00 B w języku polskim

> **Polecenie 2**
>
> Zapisz rozwiązanie, wykorzystując schemat interaktywny. Przetestuj działanie programu dla liczb 2 oraz 720.

> **Polecenie 3**
>
> Przedstaw rozwiązanie zadania w postaci programu napisanego w wybranym języku (C++, Java lub Python). Zadbaj o prawidłowe wczytanie danych z pliku tekstowego do programu.

```
1
```

```
1
```

Sprawdź się

Zadanie 3

W pliku `liczby.txt` znajduje się 100 liczb naturalnych z przedziału [1, 1000].

Plik o rozmiarze 482.00 B w języku polskim

Napisz program, który dla każdej liczby `n` z pliku `liczby.txt` obliczy sumę dzielników naturalnych liczb z przedziału [1, `n`]. Odpowiedzi zapisz w pliku `wynik.txt`.

> **Przykład 1**
>
> Weźmy pod uwagę liczbę 4.
>
> Liczby z przedziału [1,4] to 1, 2, 3 i 4.
>
> Suma dzielników liczby 1 to 1.
>
> Suma dzielników liczby 2 to 1 + 2 = 3.
>
> Suma dzielników liczby 3 to 1 + 3 = 4.
>
> Suma dzielników liczby 4 to 1 + 2 + 4 = 7.
>
> Mamy więc 1 + 3 + 4 + 7 = 15. Poprawną odpowiedzią jest więc liczba 15.
>
> Dla liczby 5 poprawnym wynikiem jest 21, dla 6 natomiast 33 itd.

Do oceny oddajesz:

- plik `wynik.txt` zawierający odpowiedź do zadania (100 liczb naturalnych oznaczających sumy dzielników naturalnych liczb z przedziału [1, `n`]),
- plik(i) z komputerową realizacją zadania (kodem programu).

> **Praca domowa**
>
> Przedstaw rozwiązanie zadania w postaci programu napisanego w wybranym języku (C++, Java lub Python). Zadbaj o prawidłowe wczytanie danych z pliku tekstowego do programu.

Ćwiczenie 1

Język C++

Twoje zadania

1. Program wypisuje sumy dzielników naturalnych liczb z przedziału <1; `liczba`>.

Ćwiczenie 2

Język Java

Twoje zadania

1. Program wypisuje sumy dzielników naturalnych liczb z przedziału <1; `liczba`>.

Ćwiczenie 3

Język Python

Twoje zadania

1. Program wypisuje sumy dzielników naturalnych liczb z przedziału <1; `liczba`>.

Odpowiedź do zadania

Odpowiedź do zadania dla liczb z pliku `liczby.txt`:

Plik o rozmiarze 741.00 B w języku polskim

Dla nauczyciela

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

Temat: Schematy rozgałęzione i algorytmy złożone – zadania maturalne

Grupa docelowa:

Liceum ogólnokształcące i technikum, liceum ogólnokształcące, technikum, zakres podstawowy i 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.

II. Programowanie i rozwiązywanie problemów z wykorzystaniem komputera oraz innych urządzeń cyfrowych: układanie i programowanie algorytmów, organizowanie, wyszukiwanie i udostępnianie informacji, posługiwanie się aplikacjami komputerowymi.

Treści nauczania – wymagania szczegółowe

I. Rozumienie, analizowanie i rozwiązywanie problemów.

Zakres podstawowy. Uczeń:

1) planuje kolejne kroki rozwiązywania problemu, z uwzględnieniem podstawowych etapów myślenia komputacyjnego (określenie problemu, definicja modeli i pojęć, znalezienie rozwiązania, zaprogramowanie i testowanie rozwiązania).

4) porównuje działanie różnych algorytmów dla wybranego problemu, analizuje algorytmy na podstawie ich gotowych implementacji;

5) sprawdza poprawność działania algorytmów dla przykładowych danych.

II. Programowanie i rozwiązywanie problemów z wykorzystaniem komputera i innych urządzeń cyfrowych.

Zakres podstawowy. Uczeń:

1) projektuje i programuje rozwiązania problemów z różnych dziedzin, stosuje przy tym: instrukcje wejścia/wyjścia, wyrażenia arytmetyczne i logiczne, instrukcje warunkowe, instrukcje iteracyjne, funkcje z parametrami i bez parametrów, testuje poprawność programów dla różnych danych; w szczególności programuje algorytmy z punktu I.2);

2) do realizacji rozwiązań problemów prawidłowo dobiera środowiska informatyczne, aplikacje oraz zasoby, wykorzystuje również elementy robotyki;

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):

- Utrwalisz wiedzę dotyczącą schematów rozgałęzionych i algorytmów złożonych.
- Rozwiążesz kilka zadań typu maturalnego, dotyczących algorytmów rozgałęzionych, wykorzystując różne typy notacji algorytmów.
- Zapiszesz algorytmy rozgałęzione za pomocą pseudokodu i wybranego języka programowania.

Strategie nauczania:

- konstruktywizm;
- konektywizm.

Metody i techniki nauczania:

- dyskusja;
- rozmowa nauczająca z wykorzystaniem multimedium i ćwiczeń interaktywnych;
- ćwiczenia praktyczne;
- burza mózgów.

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: „Schematy rozgałęzione i algorytmy złożone – zadania maturalne". Nauczyciel prosi uczniów o zapoznanie się z treściami w sekcji „Przeczytaj".

Faza wstępna:

1. Nauczyciel wyświetla temat i cele zajęć zawarte w sekcji „Wprowadzenie". Następnie wspólnie z uczniami ustala kryteria sukcesu.
2. Nauczyciel prosi, by metodą burzy mózgów uczniowie przypomnieli najważniejsze informacje dotyczące schematów rozgałęzionych i algorytmów złożonych. Chętna lub wybrana osoba zapisuje je na tablicy w formie mapy myśli. W razie potrzeby nauczyciel uzupełnia propozycje uczniów.

Faza realizacyjna:

1. **Praca z tekstem.** Uczniowie analizują rozwiązanie zadania 1 z sekcji „Przeczytaj". Następnie w parach implementują je w wybranym języku programowania. Nauczyciel sprawdza poprawność wykonania zadania.
2. **Praca z multimedium.** Uczniowie zapoznają się z treścią zadania 2 z sekcji „Schemat interaktywny", a następnie analizują prezentację, w której omówiono jego rozwiązanie za pomocą pseudokodu. W razie wątpliwości nauczyciel wyjaśnia niezrozumiałe kroki procedury.
3. Uczniowie wykonują Polecenie 2 z sekcji „Schemat interaktywny".
4. **Ćwiczenie umiejętności.** Uczniowie pracując w grupach, rozwiązują zadanie 3 z sekcji „Sprawdź się". Piszą program w wybranym języku programowania, a następnie porównują swoje rozwiązanie z inna grupą programującą w tym samym języku.

Faza podsumowująca:

1. Nauczyciel wyświetla na tablicy temat lekcji i cele zawarte w sekcji „Wprowadzenie". W kontekście ich realizacji podsumowuje przebieg zajęć, a także wskazuje mocne i słabe strony pracy uczniów.

Praca domowa:

1. Uczniowie implementują, w wybranym języku programowania, rozwiązanie zadania 2 z sekcji „Schemat interaktywny".

Materiały pomocnicze:

- Oficjalna dokumentacja techniczna dla języka C++.
- Oficjalna dokumentacja techniczna dla języka Java SE 8 (lub nowszej wersji).
- Oficjalna dokumentacja techniczna dla języka Python 3 (lub nowszej wersji).
- Oficjalna dokumentacja techniczna dla kompilatora GCC/G++ 4.5 (lub nowszej wersji).
- Oficjalna dokumentacja techniczna dla oprogramowania Eclipse 4.4 (lub nowszej wersji).
- Oficjalna dokumentacja techniczna dla oprogramowania Code::Blocks 16.01 (lub nowszej wersji), Orwell Dev-C++ 5.11 (lub nowszej wersji) lub Microsoft Visual Studio.
- Oficjalna dokumentacja techniczna dla oprogramowania PyCharm lub IDLE.

Wskazówki metodyczne:

- Uczniowie mogą wykorzystać treści w sekcjach: „Przeczytaj”, „Schemat interaktywny”, „Sprawdź się” jako materiał do lekcji powtórkowej.