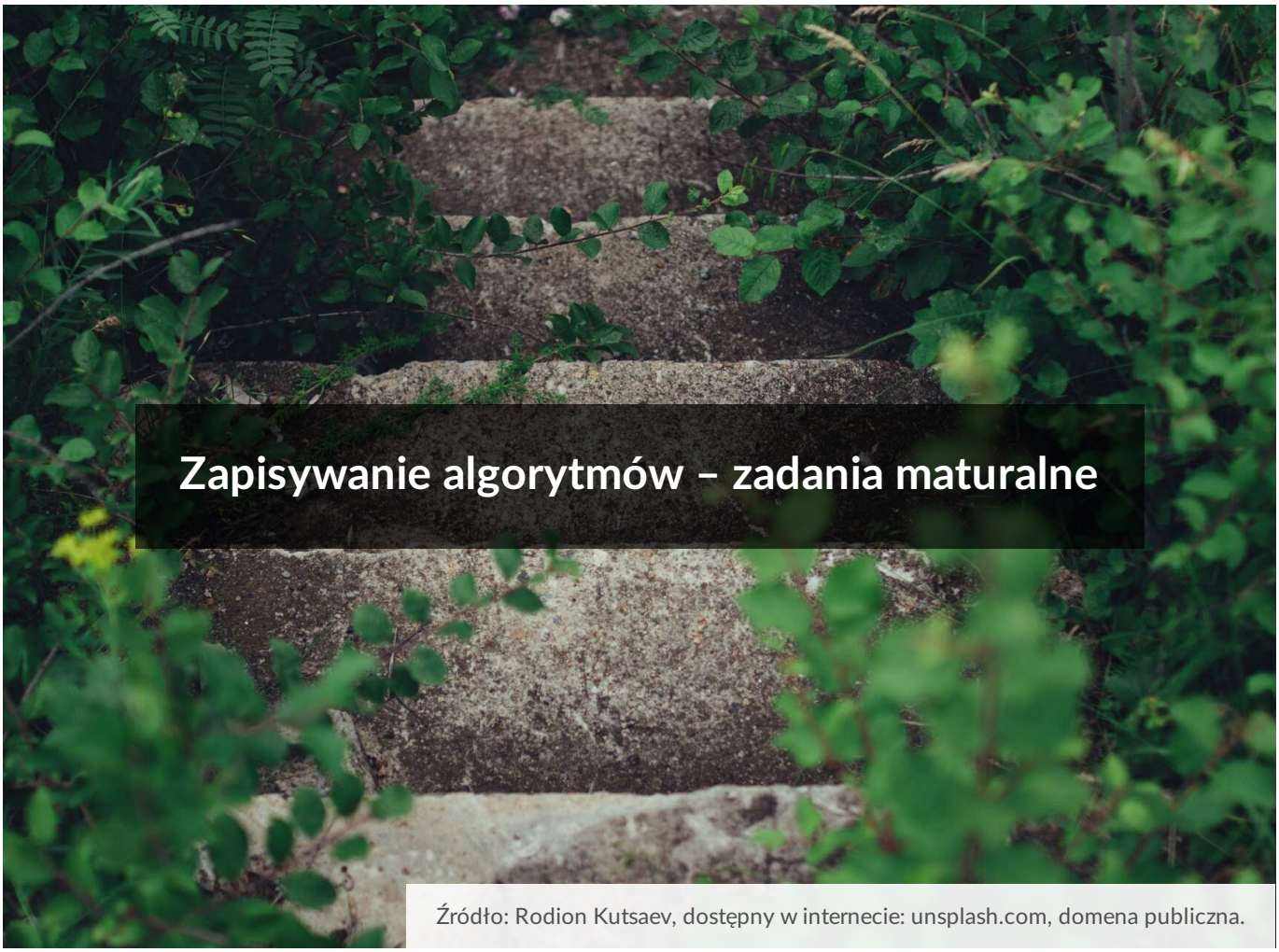
Zapisywanie algorytmów – zadania maturalne



Zapisywanie algorytmów – zadania maturalne

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

W tym e-materiale przyjrzymy się zadaniom dotyczącym trzech głównych sposobów zapisu algorytmów. Te sposoby to: lista kroków, schemat blokowy oraz pseudokod. Dzięki odpowiedniemu zapisowi jesteśmy w stanie lepiej zrozumieć algorytm i sprawniej zaimplementować go w dowolnym języku programowania.

Więcej informacji i ćwiczeń dotyczących zapisywania algorytmów znajdziesz w e-materiałach:

- Zapisywanie algorytmów za pomocą listy kroków,
- Zapisywanie algorytmów za pomocą schematu blokowego,
- Zapisywanie algorytmów za pomocą schematu blokowego – ćwiczenia,
- Zapisywanie algorytmów za pomocą pseudokodu.

Twoje cele

- Rozwiążesz kilka zadań typu maturalnego, wykorzystując różne sposoby zapisu algorytmów.
- Zapoznasz się z przykładowym zadaniem maturalnym.
- Przećwiczysz umiejętność korzystania z pseudokodu, ze schematów blokowych i z list kroków.

Przeczytaj

Zadanie 1. Potęgowanie modulo

Rozważmy operację potęgowania modularnego, stosowaną np. w algorytmie RSA. Liczbę a podnosimy do potęgi x, po czym bierzemy resztę z dzielenia otrzymanej liczby przez ustaloną liczbę M, dzięki czemu otrzymujemy wynik

$$b = a^x \bmod M,$$

gdzie a, M – dodatnie liczby całkowite, x – nieujemna liczba całkowita.

Mówimy wtedy, że $a^x \bmod M$ równa się b.

> **Przykład 1**
>
> Dla $a = 2, x = 5, M = 7$ liczymy resztę z dzielenia 2^5 (czyli 32) przez 7, zatem $b = 4$.
>
> Dla $a = 3, x = 3$ i $M = 11$ mamy $b = 3^3 mod\ 11 = 5$, natomiast dla $a = 10, x = 2$ i $M = 13$ wynikiem jest $b = 10^2\ mod\ 13 = 9$.

Zapisz w wybranej przez siebie notacji (w postaci pseudokodu, listy kroków lub w wybranym języku programowania) algorytm, który gdy są dane liczby a, x i M, obliczy $b = a^x\ mod\ M$. Aby otrzymać maksymalną liczbę punktów, twój algorytm powinien wykonywać $O(\log x)$ operacji arytmetycznych wymienionych w poniższej 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 wymienione operacje.

Specyfikacja problemu:

Dane:

- `a` – liczba całkowita dodatnia
- `x` – nieujemna liczba całkowita
- `M` – liczba całkowita dodatnia

Wynik:

- `b` – nieujemna liczba całkowita o wartości równej $a^x \bmod M$

> Zadanie zostało opracowane przez Centralną Komisję Egzaminacyjną i pojawiło się w przykładowym arkuszu maturalnym egzaminu z informatyki w formule 2023. Cały

> arkusz można znaleźć na stronie internetowej CKE.

Rozwiązanie

Pseudokod

> **Ważne!**
>
> W zapisanym za pomocą pseudokodu algorytmie używamy operatorów `div` i `mod`. Operator `div` oznacza dzielenie całkowitoliczbowe, a `mod` – resztę z dzielenia.

Zauważmy, że ograniczenie liczby operacji ($O(\log x)$) wyklucza podnoszenie do potęgi poprzez mnożenie podstawy tyle razy, ile wynosi wykładnik ($O(x)$). W związku z tym skorzystamy z algorytmu szybkiego potęgowania liczb.

Zapisujemy początkowe warunki funkcji:

```
1 funkcja potęga(a,x,M)
2     wynik ← 1
3     mnoznik ← a
```

W każdym przebiegu pętli będziemy podnosić mnożnik do kwadratu i wykonywać na nim działanie mod b (zgodnie z przechodzeniem na wyższe bity z postaci binarnej) oraz pomniejszać wartość x za pomocą dzielenia całkowitoliczbowego przez 2. Jeśli bit na danej pozycji jest niezerowy, przemnażamy wynik przez mnożnik i wykonujemy operację mod b:

```
1 funkcja potęga(a,x,M)
2     wynik ← 1
3     mnoznik ← a
4     dopóki x > 0 wykonuj:
5         jeżeli x mod 2 = 1 wykonaj:
6             wynik ← wynik * mnoznik mod M
7         mnoznik ← mnoznik * mnoznik mod M
8         x ← x div 2
```

Po zakończeniu pętli (czyli wymnożeniu dla wszystkich bitów z postaci binarnej liczby x) zwracamy wynik:

```
1 funkcja potęga(a,x,M)
2     wynik ← 1
3     mnoznik ← a
```

```
    dopóki x > 0 wykonuj:
        jeżeli x mod 2 = 1 wykonaj:
            wynik ← wynik * mnoznik mod M
        mnoznik ← mnoznik * mnoznik mod M
        x ← x div 2
    zwróc wynik
```

Oto rozwiązania w innych notacjach:

Lista kroków

1. Zainicjuj `wynik` jako 1.
2. Zainicjuj `mnożnik` jako `a`.
3. Dopóki `x > 0`, wykonuj kroki 4 - 7.
4. Jeżeli `x mod 2 = 1`, wykonaj krok 5.
5. Wynik pomnóż przez `mnoznik mod M`.
6. Mnożnik pomnóż przez `mnoznik mod M`.
7. Podziel całkowicie `wynik` przez 2.
8. Zwróć `wynik`.

C++

```cpp
int potega(int a, int x, int M) {
    int wynik = 1;
    int mnoznik = a;
    while (x > 0) {
        if (x % 2 == 1) {
            wynik = wynik * mnoznik % M;
        }
        mnoznik = mnoznik * mnoznik % M;
        x = x / 2;
    }
    return wynik;
}
```

Java

```java
int potega(int a, int x, int M) {
    int wynik = 1;
    int mnoznik = a;
```

```
    while (x > 0) {
            if (x % 2 == 1) {
            wynik = wynik * mnoznik % M;
    }
        mnoznik = mnoznik * mnoznik % M;
        x = x / 2;
    }
    return wynik;
}
```

Python

```
def potega(a, x, M):
    wynik = 1
    mnoznik = a
    while x > 0:
        if x % 2 == 1:
            wynik = wynik * mnoznik % M
        mnoznik = mnoznik * mnoznik % M
        x = x // 2
    return wynik
```

Schemat oceniania

- **4 pkt** – za poprawny algorytm, w tym:
  - **3 pkt** – za poprawne obliczenie potęgi a^x (w tym **1 pkt** za warunki początkowe oraz **2 pkt** za poprawną konstrukcję pętli z uwzględnieniem parzystości x),
  - **1 pkt** – za zwrócenie poprawnego wyniku,
- **2 pkt** – za poprawny algorytm o złożoności większej niż logarytmiczna,
- **0 pkt** – za odpowiedź niepoprawną lub niepełną albo za brak odpowiedzi.

> **Praca domowa**
>
> Na podstawie przedstawionych funkcji napisz na swoim komputerze program w języku, w którym programujesz. Przetestuj rozwiązanie dla wybranych przez siebie danych.

Zadanie 2

W tym zadaniu zajmujemy się algorytmami działającymi na n-elementowej tablicy liczb całkowitych `A[1..n]`, gdzie `n` jest dodatnią liczbą całkowitą.

Poniżej zapisano rekurencyjną procedurę `W`, której parametrem jest liczba całkowita `j` z przedziału `[1, n]`:

```
procedura W(j):
    jeśli j > 1 to
        jeśli A[j-1] > A[j] to
            v ← A[j]
            A[j] ← A[j-1]
            A[j-1] ← v
            W(j-1)
```

Oto przykładowa, 10-elementowa tablica `A[1..10]` o zawartości `[2,4,6,8,10,9,7,5,3,1]`.

W wybranej przez siebie notacji zapisz nierekurencyjną wersję procedury `W`.

> **Uwaga!**
>
> W zapisie algorytmu możesz wykorzystać tylko operacje arytmetyczne (dodawanie, odejmowanie, mnożenie, dzielenie, dzielenie całkowite, reszta z dzielenia), instrukcje porównania, instrukcje sterujące i przypisania do zmiennych lub samodzielnie napisane funkcje, wykorzystujące wyżej wymienione operacje.
> Zadanie zostało opracowane przez CKE i pojawiło się w informatorze o egzaminie maturalnym z informatyki w formule 2023. Cały arkusz można znaleźć na stronie internetowej Centralnej Komisji Egzaminacyjnej.

Rozwiązanie

Przedstawmy algorytm za pomocą dopuszczonych na egzaminie maturalnym notacji, czyli pseudokodu, listy kroków, a także języków programowania: C++, Java i Python.

Zauważmy, że procedura `W` ma pewne cechy wspólne z algorytmem sortowania bąbelkowego pojedynczego elementu. Jest on zamieniany miejscami z elementem o wcześniejszym indeksie dopóty, dopóki trafi na element mniejszy od siebie. Warunki początkowe procedury pozostają bez zmiany.

Weźmy pod uwagę przykładową tablicę podaną w treści zadania: `A[1..10]` o zawartości `[2,4,6,8,10,9,7,5,3,1]`.

Jeżeli wywołamy dla niej procedurę `W` z parametrem 7, otrzymamy:

```
2 4 6 7 8 10 9 5 3 1
```

Widzimy, że element będący na siódmym miejscu w tablicy, czyli 7, był zamieniany z poprzednim elementem, aż trafił na wartość mniejszą od siebie, czyli 6.

Następnie wywołajmy procedurę z parametrem 9. Otrzymujemy następujący wynik:

```
2 3 4 6 7 8 10 9 5 1
```

Na dziewiątym miejscu znajdował się element 3. Widzimy, że był on zamieniany z poprzednim elementem, aż trafił na mniejszy od siebie, czyli 2.

Zapiszmy nierekurencyjną wersję procedury `W`. Tworzymy zmienną pomocniczą `i`. Dopóki indeks `i` jest większy od 1, a elementy na poprzedzających indeksach większe od wartości `v`, dopóty przesuwamy je w prawo.

Następnie wstawiamy element `v` w miejsce o tym indeksie, na którym skończyliśmy pętlę.

```
procedura W(j)
    jeżeli j > 1:
        v ← A[j]
        i ← j
        dopóki i > 1 oraz A[i - 1] > v wykonuj:
            A[i] ← A[i - 1]
            i ← i - 1
        A[i] ← v
```

Rozwiązania zapisane za pomocą innych dopuszczonych na egzaminie maturalnym notacji wyglądają następująco:

Lista kroków

1. Jeżeli `j > 1`, wykonaj kroki 2–6.
2. Zainicjuj `v` jako `A[j]`.
3. Zainicjuj `i` jako `j`.
4. Dopóki `i > 1` oraz `A[i - 1] > v`, wykonuj kroki 4 i 5.
5. Do `A[i]` przypisz `A[i - 1]`.
6. Do `i` przypisz `i - 1`.
7. Do `A[i]` przypisz `v`.

C++

```
```

```
void W(int j) {
    if (j > 1) {
        int v = A[j];
        int i = j;
        while ((i > 1) && A[i - 1] > v) {
            A[i] = A[i - 1];
            i -= 1;
        }
        A[i] = v;
    }
}
```

Java

```
void W(int j) {
    if (j > 1) {
        int v = A[j];
        int i = j;
        while ((i > 1) && A[i - 1] > v) {
            A[i] = A[i - 1];
            i -= 1;
        }
        A[i] = v;
    }
}
```

Python

```
def W(j):
    if j > 1:
        v = A[j]
        i = j
        while i > 1 and A[i - 1] > v:
            A[i] = A[i - 1]
            i -= 1
        A[i] = v
```

Schemat oceniania

- **3 pkt** – poprawny algorytm w tym:

    - **1 pkt** – za poprawne inicjowanie zmiennych,
    - **1 pkt** – za poprawną konstrukcję pętli,
    - **1 pkt** – za poprawne instrukcje w pętli,
- **0 pkt** – za niepoprawną odpowiedź albo brak odpowiedzi.

> **Praca domowa**
>
> Na podstawie przedstawionych funkcji napisz na swoim komputerze program w języku, w którym programujesz. Przetestuj rozwiązanie dla wybranych przez siebie danych.

Słownik

> **lista kroków**
>
> jeden ze sposobów zapisu algorytmu; jest to numerowana instrukcja postępowania
> **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

Prezentacja multimedialna

Zadanie 3. Ulubione liczby

Małgosia i Jaś lubią liczby. Małgosia lubi liczby nieparzyste, a Jaś lubi liczby parzyste. Każde z dzieci zapisało po kilka spośród swoich ulubionych liczb na jednej wspólnej kartce. Najpierw Małgosia zapisała wszystkie swoje liczby, a potem Jaś dopisał swoje.

Napisz algorytm (w postaci listy kroków, za pomocą pseudokodu lub w wybranym języku programowania), który dla danego ciągu liczb zapisanych przez dzieci znajdzie pierwszą liczbę zapisaną przez Jasia. Zakładamy, że każde z dzieci zapisało co najmniej jedną liczbę.

Przy ocenie będzie brana pod uwagę złożoność czasowa twojego algorytmu. Maksymalną liczbę punktów uzyskasz za algorytm o złożoności lepszej niż liniowa.

> **Uwaga!**
>
> W zapisie algorytmu możesz wykorzystać tylko operacje arytmetyczne (dodawanie, odejmowanie, mnożenie, dzielenie, dzielenie całkowite, reszta z dzielenia), instrukcje porównania, instrukcje sterujące i przypisania do zmiennych lub samodzielnie napisane funkcje, wykorzystujące wyżej wymienione operacje.

Specyfikacja problemu:

Dane:

- `n` – liczba całkowita większa od `1`
- `A[1..n]` – tablica zawierająca ciąg `n` liczb zapisanych przez dzieci (najpierw wszystkie liczby nieparzyste, a potem wszystkie liczby parzyste)

Wynik:

- `w` – pierwsza od lewej parzysta liczba w tablicy `A`

Przykład:

Dane:

- `n = 10`
- `A[1..n] = {5, 99, 3, 7, 111, 13, 4, 24, 4, 8}`

Wynik:

- `w = 4`

> Zadanie zostało opracowane przez Centralną Komisję Egzaminacyjną i pojawiło się na egzaminie maturalnym z informatyki w maju 2019 roku (poziom rozszerzony). Cały arkusz

można znaleźć na stronie internetowej CKE.

Polecenie 1

Zapisz rozwiązanie zadania w postaci listy kroków, pseudokodu lub za pomocą wybranego języka programowania.

Lista kroków:

Pseudokod:

```
1
```

Wybrany język programowania:

```
1
```

```
1
```

Polecenie 2

Porównaj swoje rozwiązanie z przedstawionym w prezentacji.

1

W treści zadania zaznaczono, że przy ocenie rozwiązania będzie brana pod uwagę jego złożoność czasowa, czyli złożoność obliczeniowa opisująca ilość czasu potrzebnego komputerowi do wykonania algorytmu.

Źródło: Isabelle Grosjean, pl.wikipedia.org, CC BY-SA 3.0.

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/P1FAtXbfB

Zadanie możemy rozwiązać na dwa sposoby. Jeden z nich będzie korzystał z algorytmu wyszukiwania liniowego. Drugi sposób będzie polegał na użyciu zmodyfikowanego algorytmu wyszukiwania binarnego.

2

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/P1FAtXbfB

Wiemy, że wszystkie liczby nieparzyste zostały zapisane z lewej strony, zaś parzyste znajdują się po nich. Algorytm będzie zatem sprawdzał wszystkie kolejne liczby, zaczynając od lewej. Pierwsza napotkana liczba parzysta będzie tą szukaną.

3

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/P1FAtXbfB

Implementacja algorytmu wyszukiwania liniowego zapisana za pomocą listy kroków prezentuje się następująco:

1. Zainicjuj zmienną `i` z wartością 1.

2. Dopóki element tablicy `A[i]` jest nieparzysty, wykonuj krok 2.1:

    2.1. Zwiększ zmienną `i` o 1.

3. Zwróć element `A[i]` i zakończ.

Ponieważ pętla skończy pracę, kiedy `A[i]` będzie liczbą parzystą, możemy zwrócić jej wynik bez sprawdzania dodatkowych warunków.

4

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/P1FAtXbfB

W treści zadania widnieje wzmianka, że rozwiązanie o złożoności czasowej liniowej nie zostanie ocenione na maksymalną liczbę punktów. Za zaprezentowane dotychczas rozwiązanie otrzymalibyśmy 3 punkty na 5 możliwych. Zastosujmy zatem algorytm wyszukiwania binarnego o złożoności czasowej logarytmicznej, która jest lepsza niż liniowa.

5

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/P1FAtXbfB

Użycie tego algorytmu może wydawać się nieintuicyjne. Algorytm wymaga posortowanych danych, a liczby w tablicy przecież nie muszą być ułożone ani w kolejności niemalejącej, ani nierosnącej. Wybór tego algorytmu uzasadnia jednak fakt, że skoro lewy kraniec przedziału to liczby nieparzyste, a prawy koniec to liczby parzyste, gdzieś w środku następuje przejście z liczby nieparzystej na parzystą. Możemy zatem stwierdzić, że liczby te są „posortowane” ze względu na parzystość.

6

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/P1FAtXbfB

Zapiszmy początkowe kroki algorytmu:

1. Zainicjuj zmienną `lewy` z wartością 1.

2. Zainicjuj zmienną `prawy` z wartością n.

3. Zainicjuj zmienną `środek`.

4. Dopóki `lewy` jest mniejszy od `prawy`, wykonuj...

7

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/P1FAtXbfB

Następnie wyznaczamy środek przedziału. Jest on częścią całkowitą ze średniej arytmetycznej jego krańców. Środek przedziału będzie potrzebny w kolejnym kroku.

1. Zainicjuj zmienną `lewy` z wartością 1.

2. Zainicjuj zmienną `prawy` z wartością n.

3. Zainicjuj zmienną `środek`.

4. Dopóki `lewy` jest mniejszy od `prawy`, wykonuj krok 4.1:

    4.1 Do zmiennej `środek` przypisz wartość wyrażenia `(lewy + prawy) div 2`.

Operator `div` oznacza dzielenie całkowitoliczbowe, natomiast `mod` – resztę z dzielenia.

8

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/P1FAtXbfB

Jeżeli element na środku jest nieparzysty, możemy odrzucić lewą cześć przedziału włącznie z tym elementem – wszystkie elementy na lewo od niego, jak i on sam, są nieparzyste. Jeśli element jest parzysty, możemy odrzucić prawą stronę, ponieważ szukamy pierwszego parzystego elementu.

1. Zainicjuj zmienną `lewy` z wartością 1.

2. Zainicjuj zmienną `prawy` z wartością n.

3. Zainicjuj zmienną `środek`.

4. Dopóki `lewy` jest mniejszy od `prawy`, wykonuj krok 4.1.

    4.1 Do zmiennej `środek` przypisz wartość wyrażenia `(lewy + prawy) div 2`.

    4.2 Jeżeli `A[środek]` jest parzyste wykonaj krok 4.2.1, w przeciwnym razie wykonaj krok 4.2.2.

        4.2.1 Do zmiennej `prawy` przypisz wartość

zmiennej `środek`.

4.2.2 Do zmiennej `lewy` przypisz wartość zmiennej `środek + 1`.

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/P1FAtXbfB

Zwracamy otrzymany element parzysty.

1. Zainicjuj zmienną `lewy` z wartością 1.

2. Zainicjuj zmienną `prawy` z wartością n.

3. Zainicjuj zmienną `środek`.

4. Dopóki `lewy` jest mniejszy od `prawy`, wykonuj krok 4.1.

    4.1 Do zmiennej `środek` przypisz wartość wyrażenia `(lewy + prawy) div 2`.

    4.2 Jeżeli `A[środek]` jest parzyste, wykonaj krok 4.2.1, w przeciwnym razie wykonaj krok 4.2.2.

        4.2.1 Do zmiennej `prawy` przypisz wartość zmiennej `środek`.

        4.2.2 Do zmiennej `lewy` przypisz wartość zmiennej `środek + 1`.

5. Zwróć element `A[prawy]`.

10

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/P1FAtXbfB

Tak zapisany kod gwarantuje 5 punktów na 5 możliwych, ponieważ złożoność czasowa tego algorytmu jest logarytmiczna, zatem znacznie mniejsza od liniowej.

11

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/P1FAtXbfB

W kolejnych slajdach przedstawimy rozwiązania na 5 punktów, zapisane za pomocą pseudokodu oraz dostępnych na egzaminie maturalnym języków programowania. Zwróć uwagę na to, że w programach zastosowano indeksowanie od 0, mimo że ten sposób numeracji jest niezgodny ze specyfikacją. Rozwiązanie takie wynika z faktu, że w językach programowania zawsze pierwszy indeks przyjmuje wartość 0.

12

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/P1FAtXbfB

Rozwiązanie zapisane za pomocą pseudokodu:

```
1 funkcja algorytm(A, n)
2     lewy ← 0
3     prawy ← n - 1
4     dopóki lewy < prawy
wykonuj:
5         srodek ← (lewy +
prawy) div 2;
6         jeżeli A[srodek]
mod 2 = 0:
7             prawy ← srodek
```

```
        w przeciwnym
wypadku:
            lewy ← srodek
+ 1
    zwróć A[prawy]
```

13

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/P1FAtXbfB

Rozwiązanie w języku C++:

```
int algorytm(int A[], int
n) {
    int lewy = 0;
    int prawy = n - 1;
    while (lewy < prawy) {
        int srodek = (lewy
+ prawy) / 2;
        if (A[srodek] % 2
== 0) {
            prawy =
srodek;
        }
        else {
            lewy = srodek
+ 1;
        }
    }
    return A[prawy];
}
```

14

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/P1FAtXbfB

Rozwiązanie w języku Java:

```
int algorytm(int[] A, int n) {
    int lewy = 0;
    int prawy = n - 1;
    while (lewy < prawy) {
        int srodek = (lewy + prawy) / 2;
        if (A[srodek] % 2 == 0) {
            prawy = srodek;
        }
        else {
            lewy = srodek + 1;
        }
    }
    return A[prawy];
}
```


Więcej zadań maturalnych znajdziesz w zbiorze dostępnym na stronie internetowej Okręgowej Komisji Egzaminacyjnej w Warszawie.
Źródło: OKE, oke.waw.pl, tylko do użytku edukacyjnego.

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/P1FAtXbfB

Rozwiązanie w języku Python:

```
def algorytm(A, n):
    lewy = 0
    prawy = n - 1
    while lewy < prawy:
        srodek = (lewy + prawy) // 2
```

```
6           if A[srodek] % 2
== 0:
7               prawy = srodek
8           else:
9               lewy = srodek
+ 1
10      return A[prawy]
```


> **Polecenie 3**
>
> Oceń swoje rozwiązanie zadania zgodnie ze schematem przedstawianym poniżej.

Schemat oceniania

- **5 pkt** – za poprawny algorytm o złożoności czasowej mniejszej niż liniowa, w tym:
  - **1 pkt** – za prawidłowy warunek pętli,
  - **1 pkt** – za prawidłowe wyznaczenie podziału ciągu liczb,
  - **1 pkt** – za prawidłowe wyznaczenie początku podciągu liczb,
  - **1 pkt** – za prawidłowe wyznaczenie końca podciągu liczb,
  - **1 pkt** – za prawidłowe wyznaczenie pierwszego elementu parzystego w `A[1..n]` (lub jego indeksu),
- **3 pkt** – za poprawny algorytm o złożoności czasowej liniowej, w tym:
  - **1 pkt** – za prawidłowy przebieg pętli,
  - **1 pkt** – za sprawdzenie warunku (parzystości liczby),
  - **1 pkt** – za prawidłowe wyznaczenie pierwszego elementu parzystego w `A[1..n]` (lub jego indeksu),
- **0 pkt** – za błędną odpowiedź albo brak odpowiedzi.

> **Ważne!**
>
> W prezentacji wspomniane zostały algorytmy wyszukiwania binarnego oraz liniowego. Więcej informacji na ich temat znajdziesz w e-materiale Znajdowanie określonego elementu w zbiorze.
>
> **Ciekawostka**
>
> Zastosowany w prezentacji algorytm wyszukiwania binarnego jest bardzo podobny do algorytmu bisekcji, używanego do wyznaczenia przybliżenia miejsca zerowego funkcji.

> Warunkami, jakie muszą być spełnione, aby bisekcja mogła zostać użyta na danym przedziale, są różne znaki wartości funkcji dla jego krańców oraz monotoniczność funkcji w tym przedziale. Z tych warunków wynika, że funkcja osiągnie wartość zerową dla pewnego argumentu w tym przedziale. Właśnie dlatego mogliśmy użyć algorytmu wyszukiwania binarnego – w omawianym przypadku różnymi znakami są liczby parzyste i nieparzyste, zaś monotoniczność wynika z tego, że liczby są podzielone na parzyste i nieparzyste. Wiemy zatem, że „miejsce zerowe" (tutaj: pierwsza liczba parzysta) mogą zostać wyszukane za pomocą tego algorytmu.

Sprawdź się

Pokaż ćwiczenia:

Zadanie 4. Cyfrowe dopełnienie

Niech n będzie nieujemną liczbą całkowitą, której najbardziej znacząca cyfra w zapisie dziesiętnym jest większa od 0 i mniejsza od 9. Cyfrowym dopełnieniem liczby n nazywamy liczbę całkowitą d, której zapis dziesiętny otrzymujemy z zapisu dziesiętnego liczby n przez zamianę każdej cyfry tego zapisu na cyfrę, która jest jej uzupełnieniem do 9.

Przykład:

Cyfrowym dopełnieniem liczby 2021 jest liczba 7978.

W postaci pseudokodu lub w wybranym języku programowania napisz algorytm, który dla dodatniej liczby całkowitej n obliczy jej cyfrowe dopełnienie d. O liczbie n wiadomo, że jej najbardziej znacząca cyfra jest większa od 0 i mniejsza od 9.

> **Uwaga!**
>
> **Twój algorytm może używać wyłącznie zmiennych przechowujących liczby całkowite oraz może operować wyłącznie na liczbach całkowitych.** W zapisie algorytmu możesz korzystać tylko z instrukcji sterujących, operatorów arytmetycznych: dodawania, odejmowania, mnożenia, dzielenia, dzielenia całkowitego i reszty z dzielenia; operatorów logicznych, porównań i instrukcji przypisywania lub samodzielnie napisanych funkcji i procedur wykorzystujących powyższe operacje. **Zabronione** jest używanie funkcji wbudowanych dostępnych w językach programowania. Nie wolno w szczególności korzystać z żadnych funkcji zamiany z typu znakowego lub napisowego na liczbowy i odwrotnie.

Specyfikacja problemu:

Dane:

- n – dodatnia liczba całkowita taka, że jej najbardziej znacząca cyfra jest większa od 0 i mniejsza od 9

Wynik:

- d – dodatnia liczba całkowita, cyfrowe dopełnienie liczby n

> Zadanie zostało opracowane przez Centralną Komisję Egzaminacyjną i pojawiło się na egzaminie maturalnym z informatyki w maju 2021 roku (poziom rozszerzony). Cały arkusz można znaleźć na stronie internetowej CKE.

Ćwiczenie 1

Schemat oceniania

- **4 pkt** – za poprawny algorytm, w tym:
  - **1 pkt** – za poprawne odwoływanie się (w pętli) do cyfry najmniej znaczącej albo najbardziej znaczącej oraz jej modyfikację (obliczenie jej dopełnienia),
  - **1 pkt** – za poprawną konstrukcję pętli,
  - **1 pkt** – za poprawne instrukcje wyliczające kolejne potęgi liczby 10,
  - **1 pkt** – za otrzymanie poprawnej wartości d,
- **0 pkt** – za odpowiedź błędną lub brak odpowiedzi.

Zadanie 5. Liczby skojarzone

Dwie różne liczby całkowite a i b większe od 1 nazwiemy skojarzonymi, jeśli suma wszystkich różnych dodatnich dzielników a mniejszych od a jest równa $b + 1$, a suma wszystkich różnych dodatnich dzielników b mniejszych od b jest równa $a + 1$.

Skojarzone są np. liczby 140 i 195, ponieważ:

- dzielnikami 140 są: $1, 2, 4, 5, 7, 10, 14, 20, 28, 35, 70$, a ich suma wynosi $196 = 195 + 1$,
- dzielnikami 195 są: $1, 3, 5, 13, 15, 39, 65$, a suma tych liczb równa jest $141 = 140 + 1$.

Dana jest liczba całkowita a większa od 1. Ułóż i zapisz w wybranej przez siebie notacji algorytm, który znajdzie i wypisze liczbę b skojarzoną z a lub komunikat „NIE”, jeśli taka liczba nie istnieje.

W zapisie algorytmu możesz korzystać tylko z następujących operacji arytmetycznych: dodawania, odejmowania, mnożenia, dzielenia całkowitego i obliczania reszty z dzielenia.

> **Uwaga!**
>
> Przy ocenie algorytmu będzie brana pod uwagę liczba operacji arytmetycznych wykonywanych przez twój algorytm.

Specyfikacja problemu:

Dane:

- liczba całkowita `a > 1`

Wynik:

- liczba całkowita `b` skojarzona z `a` lub komunikat `NIE`, jeśli taka liczba nie istnieje

> Zadanie zostało opracowane przez Centralną Komisję Egzaminacyjną i pojawiło się na egzaminie maturalnym z informatyki w maju 2016 roku (poziom rozszerzony). Cały arkusz można znaleźć na stronie internetowej CKE.

Ćwiczenie 2

Schemat oceniania

- **4 pkt** – za poprawny algorytm, w tym:
    - **3 pkt** – za poprawne obliczanie sumy dzielników zadanej liczby (lub potencjalnej liczby skojarzonej):
        - **1 pkt** – za sumowanie kolejnych dzielników,
        - **1 pkt** – za poprawną konstrukcję pętli,
        - **1 pkt** – za algorytm o złożoności nie gorszej niż $\sqrt{}$,
    - **1 pkt** – za poprawne ustalenie liczby `b` oraz za sprawdzenie, czy liczby `a` i `b` są skojarzone,

- **0 pkt** – za odpowiedź błędną albo brak odpowiedzi.

Dla nauczyciela

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

Temat: Zapisywanie algorytmów – zadania maturalne

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.

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

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

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

2) stosuje zasady programowania strukturalnego i obiektowego w rozwiązywaniu problemów;

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

1) zapisuje za pomocą listy kroków, schematu blokowego lub pseudokodu, i implementuje w wybranym języku programowania, algorytmy poznane na wcześniejszych etapach oraz algorytmy:

b) znajdowania określonego elementu w zbiorze: lidera, idola, elementu w zbiorze uporządkowanym metodą binarnego wyszukiwania,

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):

- Rozwiążesz kilka zadań typu maturalnego, wykorzystując różne sposoby zapisu algorytmów.
- Zapoznasz się z przykładowym zadaniem maturalnym.
- Przećwiczysz umiejętność korzystania z pseudokodu, ze schematów blokowych i z list kroków.

Strategie nauczania:

- konstruktywizm;
- konektywizm.

Metody i techniki nauczania:

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

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;
- oprogramowanie dla języka C++, w tym kompilator GCC/G++ 4.5 (lub nowszej wersji) i Code::Blocks 16.01 (lub nowszej wersji), Orwell Dev-C++ 5.11 (lub nowszej wersji) lub Microsoft Visual Studio;
- oprogramowanie dla języka Java SE 8 (lub nowszej wersji), w tym Eclipse 4.4 (lub nowszej wersji);
- oprogramowanie dla języka Python 3 (lub nowszej wersji), w tym PyCharm lub IDLE.

Przebieg lekcji

Przed lekcją:

1. Nauczyciel prosi uczniów o przypomnienie sobie najważniejszych informacji związanych z zapisywaniem algorytmów.
2. **Przygotowanie do zajęć.** Nauczyciel loguje się na platformie i udostępnia e-materiał: „Zapisywanie algorytmów – zadania maturalne". Nauczyciel prosi uczniów o zapoznanie się z treściami w sekcji „Przeczytaj".

Faza wstępna:

1. Nauczyciel wprowadza uczniów szczegółowo w temat lekcji i jej cele. Może posłużyć się wyświetloną na tablicy zawartością sekcji „Wprowadzenie".
2. Prowadzący prosi uczniów, aby zgłaszali swoje propozycje pytań do tematu. Jedna osoba może zapisywać je na tablicy. Gdy uczniowie wyczerpią swoje pomysły, a pozostały jakieś ważne kwestie do poruszenia, nauczyciel je dopowiada.

Faza realizacyjna:

1. **Praca z tekstem.** Nauczyciel ocenia, na podstawie informacji na platformie, stan przygotowania uczniów do zajęć. Jeżeli jest ono niewystarczające prosi o ciche zapoznanie się z treścią w sekcji „Przeczytaj". Chętna lub wybrana osoba referuje najważniejsze informacje dotyczące zapisywania algorytmów.
2. W razie potrzeby nauczyciel wyjaśnia niezrozumiałe kwestie związane z problemami przedstawionymi w sekcji „Przeczytaj".
3. **Praca z multimedium.** Nauczyciel wyświetla zawartość sekcji „Prezentacja multimedialna". Uczniowie zapoznają się z zadaniem 3 i indywidualnie opracowują jego rozwiązanie. Następnie porównują swoje rozwiązania z przedstawionym w prezentacji.
4. Chętne lub wybrane osoby przedstawiają swoje rozwiązania.
5. **Ćwiczenie umiejętności.** Uczniowie pracując w parach rozwiązują zadanie 4 z sekcji „Sprawdź się". Następnie omawiają rozwiązanie na forum klasy.

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.
2. Wybrany uczeń podsumowuje zajęcia, zwracając uwagę na nabyte umiejętności.

Praca domowa:

1. Uczniowie rozwiązują zadanie 5 z sekcji „Sprawdź się".

Materiały pomocnicze:

- Oficjalna dokumentacja techniczna dla języka C++.

- Oficjalna dokumentacja techniczna dla kompilatora GCC/G++ 4.5 (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 języka Java SE 8 (lub nowszej wersji).
- Oficjalna dokumentacja techniczna dla oprogramowania Eclipse 4.4 (lub nowszej wersji).
- Oficjalna dokumentacja techniczna dla języka Python 3 (lub nowszej wersji).
- Oficjalna dokumentacja techniczna dla oprogramowania PyCharm lub IDLE.

Wskazówki metodyczne:

- Uczniowie mogą wykorzystać multimedium w sekcji „Prezentacja multimedialna” do przygotowania się do lekcji powtórkowej.