Odwrotna notacja polska – zadania maturalne

Źródło: Guillaume Techer, domena publiczna.

Część języków programowania, a także kalkulatorów, używa do swoich obliczeń odwrotnej notacji polskiej. Wiesz już, na czym ona polega i czym różni się od tradycyjnego zapisu wyrażeń.

W tym e-materiale rozwiążesz zadania typu maturalnego, w których wykorzystasz umiejętność konwersji zapisu notacji.

Implementacje tego zagadnienia w różnych językach programowania przedstawiamy w e-materiałach:

- Odwrotna notacja polska w języku C++,
- Odwrotna notacja polska w języku Java,
- Odwrotna notacja polska w języku Python.

Twoje cele

- Przeanalizujesz przykładowe rozwiązania zadań maturalnych.
- Rozwiążesz kilka zadań maturalnych, wymagających konwersji zapisu na odwrotną notację polską.
- Prześledzisz schemat oceniania zadań maturalnych.

Przeczytaj

Zadanie 1

Odwrotna notacja polska (ONP) jest zapisem wyrażeń arytmetycznych łatwo przetwarzanym przez komputer. Zapis w ONP wyrażenia `W` nazywamy postacią ONP i oznaczamy ją `ONP(W)`. Operator (dodawania, odejmowania, mnożenia, dzielenia) umieszczamy za jego argumentami, np. `2 + 5` zapisujemy jako `2 5 +`. Postać ONP dla wyrażenia definiujemy rekurencyjnie w następujący sposób:

1. Jeżeli `W` jest liczbą, to jego postać ONP jest równa `W`.
2. Jeżeli `W` ma postać W_1 `op` W_2, gdzie `op` jest operatorem, a W_1 oraz W_2 wyrażeniami, to `ONP(W)` jest równe `ONP(`W_1`) ONP(`W_2`) op`.

Przykład:

W = W_1 op W_2	**W_1**	**W_2**	**op**	**ONP(W)**
1 + 2	1	2	+	1 2 +
5 - 7	5	7	-	5 7 -
3 * (5 - 7)	3	5 - 7	*	3 5 7 - *
(1 + 2) + (3 * (5 - 7))	1 + 2	3 * (5 - 7)	+	1 2 + 3 5 7 - * +

Zauważmy, że dla `W` = (1 + 2) + (3 * (5 − 7)) wartość `ONP(W)` uzyskujemy z połączenia ONP(1 + 2) = 1 2 +, ONP(3 * (5 − 7)) = 357 − * oraz znaku dodawania +.

Przedstawiony algorytm sprawdza, czy podany na wejściu ciąg liczb i operatorów jest poprawnym wyrażeniem w ONP.

Specyfikacja problemu:

Dane:

- `n` – liczba całkowita dodatnia
- `X[1..n]` – ciąg elementów o długości `n`, z których każdy jest liczbą lub znakiem ze zbioru {+, −, *}

Wynik:

- `Tak` – jeśli `X` jest poprawnym wyrażeniem w ONP
- `Nie` – w przeciwnym przypadku

Algorytm:

```
licznik ← 0
dla i = 1, 2, ..., n wykonuj:
    jeżeli X[i] jest cyfrą wykonaj:
        licznik ← licznik + 1

    jeżeli X[i] należy do zbioru {+, –, *} wykonaj:
        jeżeli licznik < 2 wykonaj:
            zwróć "Nie" i zakończ działanie
        w przeciwnym razie wykonaj:
            licznik ← licznik – 1

jeżeli licznik != 1 wykonaj:
    zwróć "Nie"
w przeciwnym razie wykonaj:
    zwróć "Tak"
```

Oceń, które z podanych napisów są wyrażeniami zapisanymi poprawnie w ONP, wpisując słowa TAK lub NIE w trzeciej kolumnie przedstawionej tabeli. W drugiej kolumnie podaj wartości zmiennej `licznik` po zakończeniu działania algorytmu dla poszczególnych napisów.

Napis	Wartość zmiennej licznik po zakończeniu algorytmu	Czy poprawne wyrażenie w ONP?
1 2 + *	1	NIE
1 2 + 3 4 - 5 * 7 8 + 9		
1 2 3 4 5 + + + +		
1 2 3 4 5 + + + + + +		
1 2 3 4 5 + + + + +		
1 2 + 2 3 - 3 4 * 4 5 + - - - -		
1 2 + 2 3 - 3 4 * 4 5 + - - - - -		
1 2 + 3 4 - 5 * 7 8 + 9 + + +		

> Zadanie zostało stworzone przez Centralną Komisję Egzaminacyjną i zostało opublikowane w zbiorze zadań z informatyki. Cały zbiór można znaleźć na stronie internetowej CKE.

Rozwiązanie

Jak wiemy, każde wyrażenie poprawnie zapisane w ONP, po zakończeniu obliczeń pozostawia na stosie tylko jedną wartość. Jest nią wynik działania.

Z każdą kolejną analizowaną liczbą na stosie wzrasta o jeden liczba danych, ponieważ napotkaną liczbę wrzucamy na stos. Jeżeli napotykamy operator, to zdejmujemy dwie liczby ze stosu i wykonujemy na nich operację wskazaną przez operator. Następnie wynik odkładamy z powrotem na stos. W rezultacie liczba elementów na stosie maleje o 1.

Zanim uznamy zapis za poprawny, upewniamy się, że zawsze gdy w wyrażeniu napotykamy operator, na stosie znajdują się co najmniej dwa elementy.

Przeanalizujmy podany w zadaniu algorytm dla ciągów znaków z tabeli.

Pierwszy napis to 1 2 + *. Pierwsze dwa jego znaki są cyframi, a zatem po pierwszych dwóch iteracjach głównej pętli algorytmu zmienna `licznik` będzie równa 2. Kolejny znak to +. Nie jest on cyfrą, więc zmniejszamy zmienną `licznik` o 1. Zmienna `licznik` zawiera obecnie wartość 1.

Gdyby napis kończył się w tym miejscu, byłby on poprawnym wyrażeniem w ONP. Jednakże mamy jeszcze jeden znak, który nie jest cyfrą, więc ponownie zmniejszamy wartość zmiennej `licznik` o 1. Jednak w 7. linii algorytmu znajduje się instrukcja warunkowa, która w tym przypadku kończy algorytm wcześniej i zwraca wartość `Nie`.

Napis	**Wartość zmiennej `licznik` po zakończeniu algorytmu**	**Czy poprawne wyrażenie w ONP?**
1 2 + *	1	NIE
1 2 + 3 4 - 5 * 7 8 + 9		
1 2 3 4 5 + + + +		
1 2 3 4 5 + + + + + +		
1 2 3 4 5 + + + + +		
1 2 + 2 3 - 3 4 * 4 5 + - - - -		
1 2 + 2 3 - 3 4 * 4 5 + - - - - -		
1 2 + 3 4 - 5 * 7 8 + 9 + + +		

Przejdźmy do analizy drugiego napisu, czyli 1 2 + 3 4 - 5 * 7 8 + 9. Po pierwszych dwóch iteracjach wartość licznika będzie równa 2. Następnie mamy znak +, więc zmienna `licznik` będzie wynosić 1. Następnie kolejne dwie cyfry zwiększą ją o 2 – w tym momencie zmienna `licznik` ma wartość 3. Podążając dalej zgodnie z algorytmem, na końcu otrzymamy zmienną `licznik` wynoszącą 4. Ponownie po zakończeniu działania algorytmu na stosie zostaje nam inna liczba elementów niż 1, więc to wyrażenie również nie jest poprawne w ONP.

Napis	**Wartość zmiennej `licznik` po zakończeniu algorytmu**	**Czy poprawne wyrażenie w ONP?**
1 2 + *	1	NIE
1 2 + 3 4 - 5 * 7 8 + 9	4	NIE
1 2 3 4 5 + + + +		
1 2 3 4 5 + + + + + +		
1 2 3 4 5 + + + + +		
1 2 + 2 3 - 3 4 * 4 5 + - - - -		
1 2 + 2 3 - 3 4 * 4 5 + - - - - -		
1 2 + 3 4 - 5 * 7 8 + 9 + + +		

Kolejny napis to 1 2 3 4 5 + + + +. Pięć pierwszych znaków to cyfry, zatem po pięciu pierwszych iteracjach zmienna `licznik` wynosi 5. Kolejne cztery znaki nie są cyframi, więc po kolejnych czterech iteracjach wartość zmiennej `licznik` wynosi 1. Jest to już koniec napisu, więc algorytm zakończył swoje działanie i wartość zmiennej `licznik` wynosi 1. Analizowany napis jest zatem poprawnym wyrażeniem w ONP.

Napis	**Wartość zmiennej `licznik` po zakończeniu algorytmu**	**Czy poprawne wyrażenie w ONP?**
1 2 + *	1	NIE
1 2 + 3 4 - 5 * 7 8 + 9	4	NIE
1 2 3 4 5 + + + +	1	TAK
1 2 3 4 5 + + + + + +		
1 2 3 4 5 + + + + +		

Napis	Wartość zmiennej `licznik` po zakończeniu algorytmu	Czy poprawne wyrażenie w ONP?
1 2 + 2 3 - 3 4 * 4 5 + - - - -		
1 2 + 2 3 - 3 4 * 4 5 + - - - - -		
1 2 + 3 4 - 5 * 7 8 + 9 + + +		

Przejdźmy do kolejnego napisu. Tym razem wygląda następująco: 1 2 3 4 5 + + + + + +. Jest bardzo podobny do poprzedniego, stąd wiemy już, że po pierwszych dziewięciu iteracjach głównej pętli zmienna `licznik` wynosi 1. W tym przypadku jednak nie jest to jeszcze koniec napisu. Dziesiąty analizowany znak nie jest cyfrą, więc zmienna `licznik` powinna wynosić 0. Nie dojdzie jednak do tego z powodu instrukcji warunkowej z 7. linii algorytmu. Sprawia ona, że działanie algorytmu kończy się wcześniej. Wiemy też, że wyrażenie nie jest poprawnym wyrażeniem w ONP.

Napis	Wartość zmiennej `licznik` po zakończeniu algorytmu	Czy poprawne wyrażenie w ONP?
1 2 + *	1	NIE
1 2 + 3 4 - 5 * 7 8 + 9	4	NIE
1 2 3 4 5 + + + +	1	TAK
1 2 3 4 5 + + + + + +	1	NIE
1 2 3 4 5 + + + + +		
1 2 + 2 3 - 3 4 * 4 5 + - - - -		
1 2 + 2 3 - 3 4 * 4 5 + - - - - -		
1 2 + 3 4 - 5 * 7 8 + 9 + + +		

Zostały cztery napisy, w których nie czekają nas już żadne niespodzianki, możemy więc uzupełnić tabelę do końca. Poprawnie wypełniona tabela wygląda następująco:

Napis	Wartość zmiennej `licznik` po zakończeniu algorytmu	Czy poprawne wyrażenie w ONP?

Napis	Wartość zmiennej `licznik` po zakończeniu algorytmu	Czy poprawne wyrażenie w ONP?
1 2 + *	1	NIE
1 2 + 3 4 - 5 * 7 8 + 9	4	NIE
1 2 3 4 5 + + + +	1	TAK
1 2 3 4 5 + + + + + +	1	NIE
1 2 3 4 5 + + + + +	1	NIE
1 2 + 2 3 - 3 4 * 4 5 + - - - -	1	TAK
1 2 + 2 3 - 3 4 * 4 5 + - - - - -	1	NIE
1 2 + 3 4 - 5 * 7 8 + 9 + + +	1	TAK

Schemat oceniania

- **2 pkt** – za poprawną odpowiedź we wszystkich wierszach,
- **1 pkt** – za poprawną odpowiedź w co najmniej czterech wierszach,
- **0 pkt** – za podanie poprawnej odpowiedzi w mniej niż czterech wierszach lub brak odpowiedzi.

Klucz oceniania pochodzi ze zbioru zadań maturalnych z informatyki, opracowanego przez Centralną Komisję Egzaminacyjną. Schemat został ułożony na podstawie podobnych zadań maturalnych. Cały zbiór można znaleźć na oficjalnej stronie internetowej CKE.

Słownik

> **odwrotna notacja polska**
>
> sposób zapisu działań, w których operator zostaje zapisany po argumentach działania
> **zapis infiksowy**
>
> klasyczny zapis arytmetyczny, w którym operator znajduje się między argumentami

Prezentacja multimedialna

Zadanie 2

Używając pseudokodu lub dowolnego języka programowania, napisz algorytm zamieniający wyrażenie na jego postać w ONP.

W wyrażeniach przyjmujemy, że operatory to znaki ze zbioru {+, –, *} oraz że każda operacja wraz z dwoma operandami objęta jest nawiasami.

> **Przykład 1**
>
> A: ((5*(((3-7)*2)-(3*(5+1))))-3)
>
> ONP(A): 5 3 7 - 2 * 3 5 1 + * - * 3 -
>
> B: ((2+4)*(4-6))
>
> ONP(B): 2 4 + 4 6 - *

Specyfikacja problemu:

Dane:

- `wyrazenie` – wyrażenie do konwersji na ONP; ciąg znaków

Wynik:

Program na standardowym wyjściu wypisuje wyrażenie w ONP.

> **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 wyżej wymienione operacje.

1

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/Ph4UJjypF

Rozwiązanie zadania przedstawimy w postaci pseudokodu, ponieważ na egzaminie

maturalnym można korzystać z pseudokodu lub z wybranego języka programowania: C/C++, Java lub Python.

2

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/Ph4UJjypF

Zaczniemy od deklaracji funkcji. Będzie ona przyjmować dwa argumenty: `wyrazenie`, oznaczające wyrażenie do konwersji na ONP oraz `i`, oznaczające obecny indeks w wyrażeniu.

```
1 funkcja ONP(wyrazenie, i):
```

3

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/Ph4UJjypF

Następnie zapiszemy instrukcję warunkową, która będzie odpowiadać za wykonanie części kodu w zależności od tego, czy obecnie badany znak to "(".

```
1 funkcja ONP(wyrazenie, i):
2     jeżeli wyrazenie[i] != '(':
```

4

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/Ph4UJjypF

Następnie sprawdzamy, czy znak jest cyfrą – cyfrą, a nie liczbą, ponieważ liczba może składać

się z więcej niż jednej cyfry. Używamy pętli `while` do wypisania całej liczby.

```
funkcja ONP(wyrazenie, i):
    jeżeli wyrazenie[i] != '(':
        dopóki wyrazenie[i] jest cyfrą wykonuj:
```

5

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/Ph4UJjypF

Dopóki kolejny znak jest cyfrą, wypisujemy go oraz zwiększamy wartość zmiennej `i` o 1.

```
funkcja ONP(wyrazenie, i):
    jeżeli wyrazenie[i] != '(':
        dopóki wyrazenie[i] jest cyfrą wykonuj:
            wypisz wyrazenie[i]
            i ← i + 1
```

6

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/Ph4UJjypF

Po zakończeniu działania pętli, zmienna `i` będzie wskazywała na pierwszy znak za liczbą. Odejmujemy od niej 1, aby wskazywała na ostatni element operandu.

```
funkcja ONP(wyrażenie, i):
```

```
    jeżeli wyrazenie[i] != '(':
        dopóki wyrazenie[i] jest cyfrą wykonuj:
            wypisz wyrazenie[i]
            i ← i + 1

        wypisz spację
        zwróć i - 1
```

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/Ph4UJjypF

7

Teraz przejdziemy do przypadku, gdy rozpatrywany obecnie znak jest nawiasem otwierającym. W tym wypadku wywołujemy ONP z indeksem powiększonym o 1. Jej wartość przypisujemy do zmiennej `i`

```
funkcja ONP(wyrazenie, i):
    jeżeli wyrazenie[i] != '(':
        dopóki wyrazenie[i] jest cyfrą wykonuj:
            wypisz wyrazenie[i]
            i ← i + 1

        wypisz spację
        zwróć i - 1

    w przeciwnym wypadku:
        i ← ONP(wyrazenie, i + 1)
```

8

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/Ph4UJjypF

Zwiększamy zmienną `i` o 1. Wskazuje ona teraz na operator arytmetyczny. Zapisujemy jego wartość do zmiennej `operator`.

```
funkcja ONP(wyrazenie, i):
    jeżeli wyrazenie[i] != '(':
        dopóki wyrazenie[i] jest cyfrą wykonuj:
            wypisz wyrazenie[i]
            i ← i + 1

        wypisz spację
        zwróć i - 1

    w przeciwnym wypadku:
        i ← ONP(wyrazenie, i + 1)
        i ← i + 1
        operator ← wyrazenie[i]
```

9

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/Ph4UJjypF

Wywołujemy ONP, by przeprowadzić analizę drugiego operandu – stojącego po prawej stronie po operatorze.

```
funkcja ONP(wyrazenie, i):
    jeżeli wyrazenie[i] != '(':
```

```
        dopóki
wyrazenie[i] jest cyfrą
wykonuj:
            wypisz
wyrazenie[i]
            i ← i + 1

        wypisz spację
        zwróć i - 1

    w przeciwnym wypadku:
        i ← ONP(wyrazenie,
i + 1)
        i ← i + 1
        operator ←
wyrazenie[i]
        i ← ONP(wyrazenie,
i + 1)
```

10

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/Ph4UJjypF

Teraz wystarczy ponownie zwiększyć zmienną `i` o 1. Wypisujemy operator oraz spację a także zwracamy indeks `i`.

```
funkcja ONP(wyrazenie, i):
    jeżeli wyrazenie[i] !=
'(':
        dopóki
wyrazenie[i] jest cyfrą
wykonuj:
            wypisz
wyrazenie[i]
            i ← i + 1

        wypisz spację
        zwróć i - 1

    w przeciwnym wypadku:
```

```
        i ← ONP(wyrazenie, i + 1)
        i ← i + 1
        operator ← wyrazenie[i]
        i ← ONP(wyrazenie, i + 1)
        i ← i + 1
        wypisz operator
        wypisz spację
        zwróć i
```

Tym samym zakończyliśmy pisanie algorytmu.
Za taki algorytm dostaniemy maksymalną liczbę punktów.

Schemat oceniania:

- **3 pkt** – za poprawny algorytm,
- **2 pkt** – za poprawny zapis, lecz błędną kolejność działań,
- **0 pkt** – za niepoprawny algorytm lub jego brak.

Sprawdź się

Pokaż ćwiczenia:

Zadanie 3

Jednym z obowiązków pracownika pewnej firmy jest wykonywanie podstawowych obliczeń. Ma on do dyspozycji prosty kalkulator, za którego pomocą może wprowadzać dowolne nieujemne liczby całkowite oraz przeprowadzać na nich operacje: `+`, `-`, `*`, `/`. Urządzenie działa w następujący sposób:

- zapamiętuje pośrednie wyniki;
- może wykonywać tylko jedną operację jednocześnie; najpierw wprowadzane są jeden lub dwa operandy, a następnie symbol operacji, która ma zostać na nich wykonana; jeśli został wprowadzony tylko jeden operand, kalkulator pobiera z pamięci wynik ostatniej wykonanej operacji;
- jeżeli spróbuje pobrać wynik z pamięci, kiedy jest ona pusta, zwróci ERROR;
- jeżeli operacje przestaną być wprowadzane do kalkulatora, a w jego pamięci pozostanie więcej niż jedna liczba, urządzenie zwróci ERROR.

Plik `operacje.txt` zawiera 50 wierszy z operacjami wykonanymi przez pracownika na kalkulatorze. Każdy wiersz należy traktować jako nową akcję, przeprowadzoną po restarcie kalkulatora. W każdej linii podawane są kolejno liczby i symbole operacji wprowadzane przez pracownika.

Napisz program, który dla każdego wiersza z operacją stwierdzi, czy kalkulator po wprowadzaniu do niego danej operacji zwrócił poprawny wynik, czy ERROR.

Plik o rozmiarze 1 021.00 B w języku polskim

> Przedstaw rozwiązanie zadania, pisząc program w języku C++, Java lub Python. Zadbaj o prawidłowe wczytanie danych z pliku tekstowego. Odpowiedź do zadania znajduje się pod sekcją ćwiczeń.

> Język JAVA

Ćwiczenie 1

Twoje zadania

1. Napisz program, który sprawdza, czy kalkulator zwróci poprawny wynik dla obu ciągów operacji.

Ćwiczenie 2

Twoje zadania

1. Napisz program, który sprawdza, czy kalkulator zwróci poprawny wynik dla obu ciągów operacji.

Ćwiczenie 3

Twoje zadania

1. Napisz program, który sprawdza, czy kalkulator zwróci poprawny wynik dla obu ciągów operacji.

Schemat oceniania

- **3 pkt** – za poprawną odpowiedź dla wszystkich wyrażeń,
- **2 pkt** – za poprawną odpowiedź dla wyrażeń w przynajmniej 25 wierszach,
- **1 pkt** – za poprawną odpowiedź dla wyrażeń w przynajmniej 10 wierszach,
- **0 pkt** – za udzielenie poprawnej odpowiedzi w mniej niż 10 wierszach lub brak odpowiedzi.

Odpowiedź

Odpowiedź do zadania dla danych zawartych w pliku tekstowym:

Plik o rozmiarze 250.00 B w języku polskim

Dla nauczyciela

Autor: Maurycy Gast
Przedmiot: Informatyka

Temat: Odwrotna notacja polska – zadania maturalne

Grupa docelowa:

Szkoła ponadpodstawowa, 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

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

2) wykorzystuje znane sobie algorytmy przy rozwiązywaniu i programowaniu rozwiązań następujących problemów:

d) zamiany wyrażenia na postać w odwrotnej notacji polskiej i obliczanie jego wartości na podstawie tej postaci,

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),

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 przykładowe rozwiązania zadań maturalnych.
- Rozwiążesz kilka zadań maturalnych, wymagających konwersji zapisu na odwrotną notację polską.
- Prześledzisz schemat oceniania zadań maturalnych.

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. **Przygotowanie do zajęć.** Nauczyciel loguje się na platformie i udostępnia e-materiał: „Odwrotna notacja polska – 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. **Rozpoznanie wiedzy uczniów.** Nauczyciel zadaje uczniom pytania dotyczące ich aktualnego stanu wiedzy w obszarze poruszanego tematu i programowania, np.
   – na czym polega odwrotna notacja polska?
   – co to jest zapis infiksowy?
   Chętni uczniowie udzielają na nie odpowiedzi.

Faza realizacyjna:

1. **Praca z tekstem.** Uczniowie analizują rozwiązanie zadania 1 z sekcji „Przeczytaj”, nauczyciel wyjaśnia ewentualne niezrozumiałe kwestie.
2. **Praca z multimedium.** Nauczyciel wyświetla zawartość sekcji „Prezentacja multimedialna”. Uczniowie wspólnie zapoznają się z przedstawionym w niej rozwiązaniem zadania nr 2 w postaci pseudokodu. Następnie w parach implementują przedstawiony program w wybranym języku programowania. Nauczyciel sprawdza poprawność wykonania zadania.
3. **Ćwiczenia umiejętności.** Uczniowie indywidualnie wykonują zadanie 3 z sekcji „Sprawdź się”. Piszą program w jednym z dostępnych języków programowania. Następnie porównują swoje rozwiązanie z inną osobą programującą w tym samym języku.

Faza podsumowująca:

1. Wybrany uczeń podsumowuje zajęcia, zwracając uwagę na nabyte umiejętności, omawia ewentualne problemy podczas rozwiązania ćwiczeń.

Praca domowa:

1. Uczniowie zapisują algorytm przedstawiony w sekcji „Przeczytaj”, wykorzystując wybrany język programowania.

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 treści w sekcjach: „Przeczytaj”, „Prezentacja multimedialna”, „Sprawdź się” jako materiał do lekcji powtórkowej.