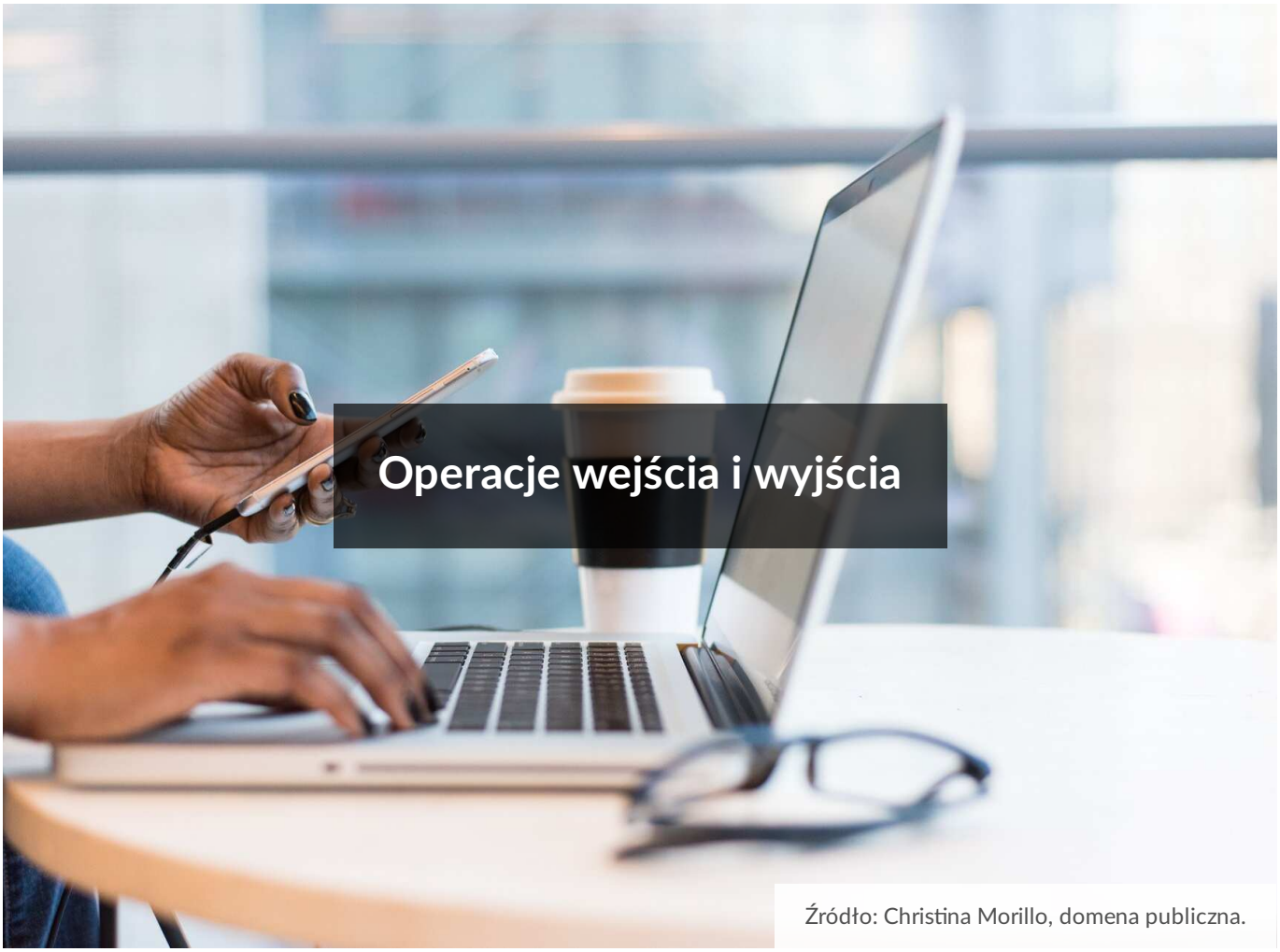
Operacje wejścia i wyjścia

- Wprowadzenie
- Przeczytaj
- Audiobook
- Sprawdź się
- Dla nauczyciela



Źródło: Christina Morillo, domena publiczna.

> W tym e-materiale powtarzamy wiadomości ze szkoły podstawowej.

Wszystkie programy, których używamy na co dzień, komunikują się z nami – bez tego byłyby bezużyteczne. Użytkownik musi mieć możliwość przekazania programowi danych do przetworzenia albo poleceń do wykonania. Z kolei program powinien zaprezentować użytkownikowi efekty działania. Instrukcje umożliwiające komunikację z użytkownikiem nazywamy operacjami wejścia/wyjścia.

Charakterystykę tych operacji w poszczególnych językach programowania znajdziesz w e-materiałach:

- Operacje wejścia i wyjścia w języku C++,
- Operacje wejścia i wyjścia w języku Java,
- Operacje wejścia i wyjścia w języku Python.

> **Twoje cele**
>
> - Przeanalizujesz przykładowe operacje wejścia i wyjścia.
> - Wyjaśnisz pojęcie interfejsu użytkownika i scharakteryzujesz jego typy.
> - Wymienisz najczęstsze problemy pojawiające się przy operacjach wejścia i wyjścia oraz przedstawisz sposoby zapobiegania im.

Przeczytaj

Operacje wejścia i wyjścia

Technicznie rzecz biorąc, operacjami wejścia i wyjścia (z ang. *input-output*, I/O) możemy nazwać każdą interakcję programu ze światem zewnętrznym. Przykładowe operacje wejścia to:

- odczytanie wciśnięcia klawiszy na klawiaturze czy ruchu myszy, ale również danych z czujników,
- odczytanie danych z pliku (np. znajdującego się na dysku),
- odbieranie pakietów sieciowych.

Przykładowe operacje wyjścia to:

- regulacja ustawień na czujniku,
- wyświetlenie obrazów lub tekstu,
- zapisanie danych do pliku,
- przesłanie danych przez sieć.

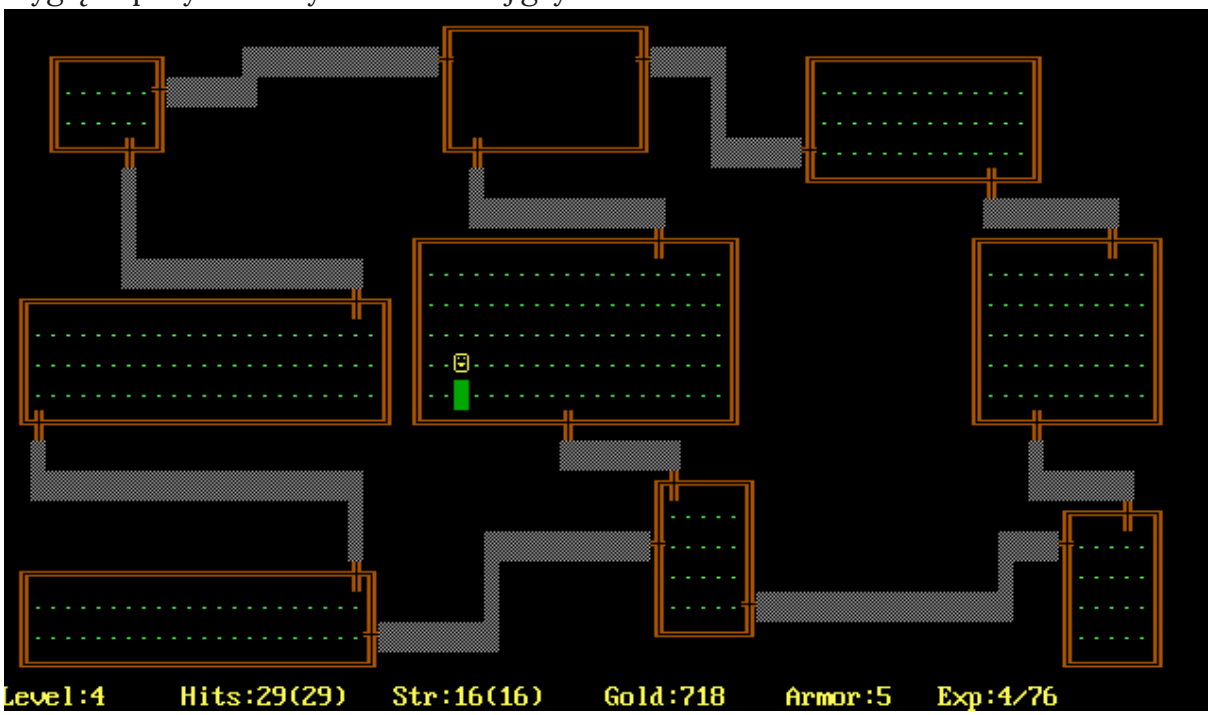
> **Ciekawostka**
>
> Wraz z nadejściem bardziej zaawansowanych komputerów pojawiła się możliwość uruchamiania wielu procesów jednocześnie. Te programy również muszą się ze sobą komunikować, a odpowiada za to tzw. komunikacja międzyprocesowa.
>
> IPC (ang. *inter-process communication*) towarzyszy nam właściwie nieustannie – przykładem metody komunikacji międzyprocesowej są pliki czy pamięć współdzielona.

Interfejsy użytkownika

Jako użytkownicy komputerów porozumiewamy się z programami. W tym celu używamy intefejsów użytkownika, które obejmują zarówno sposoby przekazywania przez nas danych do programu, jak i przekazywania nam danych przez program. Wśród najpopularniejszych typów intefejsów możemy wyróżnić:

- wiersz poleceń – wpisuje się w nim polecenia na klawiaturze, a informacja zwrotna to zazwyczaj monochromatyczny tekst (chociaż bywają także wersje obsługujące kolory);
- interfejs tekstowy, w którym, jak wskazuje nazwa, wyświetlany jest tylko tekst, jednak oferuje on szersze możliwości niż tylko wiersz poleceń – np. swobodne poruszanie się kursora po ekranie;
- interfejs graficzny, czyli to, co najczęściej wyobrażamy sobie, kiedy myślimy o interfejsach; w tym interfejsie – na wyjściu – rysowana jest grafika.

Wśród innych interfejsów możemy wymienić interfejsy gestowe (sterowane ruchem), głosowe, a także bardzo dynamicznie rozwijający się interfejs rzeczywistości wirtualnej. Interfejsy można ze sobą łączyć. Przykładem może być chociażby umożliwienie sterowania aplikacją na urządzeniu mobilnym zarówno za pomocą dotyku, jak i głosu.

> **Ciekawostka**
>
> Interfejsy można wykorzystywać w sposób kreatywny. Przykładem mogą być gry komputerowe, których sposoby sterowania często utożsamiane są z interfejsem graficznym. Istnieją gry wykorzystujące do komunikacji zarówno interfejs wiersza poleceń (na przykład gatunki MUD, SUD), jak i interfejs tekstowy. Jednym z prekursorów dzisiejszych gier jest *Rogue*, której cały interfejs, ze względu na ograniczenia technologiczne, opierał się na kolorowych znakach w trybie tekstowym.
>
> Tak wygląda przykładowy ekran dla tej gry:
>
> 
>
>
> Źródło: Artoftransformation, dostępny w internecie: commons.wikimedia.org [dostęp 31.08.2021], licencja: CC BY-SA 3.0.

Problemy związane z wejściem i wyjściem

Projektując systemy wejścia i wyjścia, należy zwrócić uwagę na kilka ważnych elementów. Pierwszym z nich jest zapewnienie poprawnego działania programów w przypadku podania błędnych danych przez użytkownika (tzw. obsługa wyjątków). Jeżeli piszemy kalkulator, który nie sprawdza, czy dzielimy liczbę przez `0`, szybko możemy doprowadzić do błędu w programie. Możemy też stworzyć luki w zabezpieczeniach naszego programu. Ważne jest zatem sprawdzanie poprawności danych (z ang. *sanity check*) przed ich przetworzeniem.

Drugim problemem jest asynchroniczność (lub jej brak). Jeżeli pobieramy dane asynchronicznie, oznacza to, że nasz program kontynuuje działanie, kiedy czeka na dane.

Przykładem takiego działania jest wysyłanie maila. Z kolei synchroniczne pobieranie danych możemy porównać do rozmowy telefonicznej: nie możemy jej rozpocząć (a tym samym nie możemy rozpocząć przekazywania danych), dopóki druga osoba nie podniesie słuchawki, tj. dopóki nie odbierze połączenia. Chociaż zdarzają się sytuacje, gdy takie zachowanie jest niepożądane, w znacznej większości przypadków asynchroniczność jest wymagana. Inny przykład dotyczy operacji bankowych: jeżeli serwer w banku czekałby, aż jeden użytkownik wykona czynność na swoim koncie, do tego czasu cały system stałby w miejscu, co jest niedopuszczalne.

> **Ciekawostka**
>
> Szczególnym przykładem asynchroniczności jest pobieranie danych z dysku. Pamięć zewnętrzna jest znacznie wolniejsza od pamięci operacyjnej, a nasz program czekałby na załadowanie wszystkich danych z dysku. Doprowadziłoby to do znacznego spowolnienia jego działania. Warto wtedy rozważyć czy nie powinniśmy ładować danych równolegle do wykonywania naszego programu, czy nawet pracować na częściowo załadowanych danych, jeśli jest to możliwe.

Trzecim problemem jest kwestia przejrzystości interfejsu. Jest to nie tyle problem programistyczny, co psychologiczny. Zadaniem twórcy interfejsu jest zaprojektowanie go w taki sposób, aby był on intuicyjny. Często widzimy interfejsy stworzone w myśl zasady „więcej znaczy lepiej" – w efekcie są one nieczytelne i zrozumiałe tylko dla ich twórców. Rozwiązaniem tego problemu jest przeanalizowanie, czy na pewno interfejs jest czytelny dla kogoś, kto nie ma z nim doświadczenia – przydatni do tego celu mogą być testerzy oprogramowania.

Ważnym zagadnieniem związanym z wejściem i wyjściem jest kwestia dostępności. Na temat jej oraz standardu WCAG przeczytasz w e-materiale WCAG, dostępność, treści alternatywne.

Słownik

> **asynchroniczność operacji wejścia**
>
> pobieranie danych w taki sposób, że program nie jest zablokowany, kiedy czeka na pełen zestaw danych
> **interfejs użytkownika**
>
> sposób obustronnej komunikacji między programem a użytkownikiem
> **operacje wyjścia i wejścia**
>
> są to wszystkie sposoby interakcji programu ze światem zewnętrznym
> **sprawdzanie poprawności danych**

(ang. *sanity check);* upewnienie się, że dane są poprawne i nie spowodują niepożądanych efektów; w razie potrzeby dane zostają naprawione lub odrzucone

Audiobook

> **Polecenie 1**
>
> Zapoznaj się z audiobookiem, następnie zastanów się, o jakich kwestiach trzeba pamiętać, projektując interfejs użytkownika.

Audiobook można wysłuchać pod adresem: https://zpe.gov.pl/b/PYNiyRmr0

Prawidłowo działający interfejs użytkownika jest fundamentalnym elementem każdego programu. Spójrzmy na przykład bardzo prostej aplikacji, jaką będzie klasyczna gra w węża. Nawet na tak małej skali będziemy mogli zauważyć poszczególne aspekty interfejsu.

Wyjście naszej gry, czyli sposób przekazywania informacji od programu do użytkownika, będzie opierało się wyłącznie na siatce kwadratów wyświetlanych na ekranie. Nie będzie w niej żadnego dźwięku. Jedynymi elementami widocznymi dla gracza będzie wąż, którym steruje oraz owoce, które należy zbierać. W każdej klatce, jaką wyświetlimy na ekranie, będziemy najpierw czyścili ekran, rysowali owoce, a następnie rysowali naszego złożonego z wielu segmentów węża.

Wejście, czyli przekazywanie informacji od użytkownika do programu, sprowadzi się do czterech przycisków – strzałek w górę, w dół, w lewo oraz w prawo.

Docieramy tutaj do pierwszego pytania: w jaki sposób pobierać informacje o wciśniętych przyciskach? Możemy aktywnie wysyłać zapytania do systemu operacyjnego o stan wszystkich klawiszy, a następnie sprawdzić tylko te cztery, które nas interesują.

Innym sposobem jest użycie tak zwanych obserwatorów, po angielsku nazywających się Listener. Każdy przycisk będzie miał swojego obserwatora. Jeżeli przycisk zostanie wciśnięty, obserwator zostanie o tym powiadomiony. Ze względu na wydajność i łatwość implementacji, jest to najczęściej używany sposób. Różne jego wersje możemy znaleźć niemal w każdej większej aplikacji, od gier komputerowych po aplikacje do obsługi arkuszy kalkulacyjnych.

Co zrobić jednak z informacją, że przycisk został wciśnięty? Tutaj pojawia się kolejna warstwa naszej aplikacji, a mianowicie odpowiednia interpretacja wejścia. Program, oprócz kierowania węża w kierunku wyznaczonym przez strzałkę, musi być

przygotowany na radzenie sobie z kilkoma przypadkami brzegowymi. Co, jeżeli wciśnięte jest kilka klawiszy naraz? Co w przypadku, gdy użytkownik wciska strzałkę w prawo, kiedy wąż już porusza się w prawo? Czy umożliwić wężowi zawracanie o sto osiemdziesiąt stopni?

Na takie pytania musimy odpowiedzieć tworząc nasz interfejs. Ważne jest, aby był on intuicyjny – nie chcemy, aby użytkownik był zmuszony do studiowania kilku stron instrukcji w celu wykonania najprostszych zadań. Jeżeli nie widzimy innego wyjścia niż stworzenie nieintuicyjnego interfejsu, powinniśmy rozważyć dodanie odpowiednich porad w łatwo dostępnych miejscach. Dobrym przykładem takiej funkcjonalności jest umieszczanie krótkich opisów czynności po najechaniu kursorem myszy na konkretne elementy naszego interfejsu.

Chociaż omawiana gra jest bardzo prosta, na wszystkie bardziej zaawansowane aplikacje można patrzeć w podobny sposób. Jeżeli chcemy umożliwić graczom ruszanie się w trójwymiarowej przestrzeni, musimy odpowiednio zaimplementować poruszanie kamery – wykorzystamy do tego mysz. Obsługa myszy to nic innego jak zwykła operacja wejścia. Tak samo działają ekrany dotykowe, sterowanie głosem czy nawet tak pozornie skomplikowane systemy jak kontrolowanie programu za pomocą gestów.

Polecenie 2

Opracuj notatkę podsumowującą najważniejsze informacje przedstawione w audiobooku.

Sprawdź się

Pokaż ćwiczenia:

Ćwiczenie 1

Zaznacz cechy dobrego interfejsu użytkownika.

- [ ] przejrzystość i czytelność
- [ ] intuicyjność obsługi
- [ ] jak najwięcej elementów na jak najmniejszej powierzchni w celu zapewnienia kompaktowości

Ćwiczenie 2

Podaj przykład sytuacji, w której asynchroniczne wejście będzie niepożądane.

Ćwiczenie 3

Dopasuj operacje do kategorii.

operacje wyjścia

operacje wejścia

pobranie danych z sieci

odtworzenie muzyki

odczytanie danych z pliku

odtworzenie filmu

odczytanie stanu ekranu dotykowego

wyświetlenie okna dialogowego

Ćwiczenie 4

Wyjaśnij, jakie zastosowania może mieć interfejs tekstowy w dobie potężnych komputerów o ogromnej mocy obliczeniowej, a jakie interfejs wiersza poleceń.

Ćwiczenie 5

Zaznacz zadanie, które nie należy do tych wykonywanych przez interfejs.

- ◯ przetwarzanie danych
- ◯ podawanie użytkownikowi informacji zwrotnych
- ◯ wczytywanie danych od użytkownika

Ćwiczenie 6

Zaznacz, pod jakim kątem powinna zostać sprawdzona poprawność danych przy wprowadzaniu adresu email.

- ◯ występowania cyfr w adresie
- ◯ struktura złożona z adresu, znaku @ oraz domeny
- ◯ występowania słów z adresu w słowniku

Ćwiczenie 7

Zapoznaj się z fragmentem kodu i wykonaj ćwiczenie.

Kod C++:

```cpp
#include <iostream>
#include <cstdlib>

using namespace std;

int main()
{
    float liczba1, liczba2;

    cout << "Podaj pierwsza liczbe: ";
    cin >> liczba1;
    cout << "Podaj druga liczbe: ";
    cin >> liczba2;

    cout << "\nWynik dzielenia pierwszej liczby przez druga to:

    system("pause");
    return 0;
}
```

Kod Java:

```java
import java.util.Scanner;

class Dzielenie{
    public static void main(String[] args){
        Scanner scanner = new Scanner(System.in);

        System.out.println("Podaj pierwszą liczbę: ");
        double liczba1 = scan.nextDouble();
        System.out.println("Podaj drugą liczbę: ");
        double liczba2 = scan.nextDouble();

        System.out.println("Wynik dzielenia pierwszej liczby prz
```

```
    }
}
```

Kod Python:

```
liczba1 = (float)(input("Podaj pierwsza liczbe: "))
liczba2 = (float)(input("Podaj druga liczbe: "))
print ('Wynik dzielenia pierwszej liczby przez druga to:  ', lic
```

Wskaż instrukcje, za pomocą których program pobiera oraz wyświetla dane.

Ćwiczenie 8

Zaproponuj, jakie wyjątki dotyczące wprowadzanych danych należy przewidzieć

i oprogramować ich obsługę w przypadku programów przedstawionych w ćwiczeniu 7.

Dla nauczyciela

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

Temat: Operacje wejścia i wyjścia

Grupa docelowa:

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

Podstawa programowa:

Cele kształcenia – wymagania ogólne

> 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 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);

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 operacje wejścia i wyjścia.
- Wyjaśnisz pojęcie interfejsu użytkownika i scharakteryzujesz jego typy.

- Wymienisz najczęstsze problemy pojawiające się przy operacjach wejścia i wyjścia oraz przedstawisz sposoby zapobiegania im.

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: „Operacje wejścia i wyjścia". Nauczyciel prosi uczniów o zapoznanie się z treściami w sekcji „Przeczytaj".

Faza wstępna:

1. Przedstawienie tematu zajęć oraz wspólne z uczniami ustalenie kryteriów sukcesu.
2. **Rozpoznanie wiedzy uczniów.** Nauczyciel prosi wybranego ucznia lub uczniów o przedstawienie sytuacji problemowej związanej z tematem lekcji.

Faza realizacyjna:

1. **Praca z tekstem.** Chętna lub wybrana osoba referuje najważniejsze kwestie poruszane w sekcji „Przeczytaj" (nauczyciel może nagrodzić tę osobę oceną). Pozostali uczniowie komentują omawiane zagadnienia i zadają pytania dotyczące niezrozumiałych elementów – nauczyciel wyjaśnia ewentualne wątpliwości.

2. **Praca z multimedium.** Nauczyciel wyświetla zawartość sekcji „Audiobook". Uczniowie wspólnie zapoznają się z poleceniem 1 i treścią audiobooka, w którym na przykładzie prostej aplikacji klasycznej gry w węża omówione zostały poszczególne aspekty interfejsu. W kolejnym kroku uczniowie pracując w parach, wypisują kwestie, o których trzeba pamiętać, projektując interfejs użytkownika. Następnie uczniowie omawiają swoje spostrzeżenia na forum klasy.
3. **Ćwiczenie umiejętności.** Uczniowie omawiają interfejs wybranej gry/aplikacji.

Faza podsumowująca:

1. Na koniec zajęć nauczyciel raz jeszcze wyświetla na tablicy temat lekcji i cele zawarte w sekcji „Wprowadzenie". W odniesieniu do ich realizacji dokonuje szczegółowej oceny rozwiązania zastosowanego przez wybranego ucznia.
2. Nauczyciel prosi uczniów o podsumowanie zgromadzonej wiedzy.

Praca domowa:

1. Uczniowie wykonują ćwiczenia 1-8 z sekcji „Sprawdź się".
2. Uczniowie wykonują polecenie 2 z sekcji „Audiobook".

Wskazówki metodyczne:

- Treści w sekcji „Audiobook" można wykorzystać jako materiał stanowiący wprowadzenie do analizy interfejsów innych aplikacji.