

SCENARIUSZ ZAAWANSOWANEGO E-MATERIAŁU

1. Metryczka materiału

Tytuł materiału	Generator programów
Numer materiału	VI.13
Autorzy scenariusza	Marzena Krzysztoń, Monika Skucińska, Michał Szymczak
Weryfikacja WCAG	Zespół ekspertów ds. WCAG (Dominika Gaponiuk, Agnieszka Brodowska, Urszula Grygier, Łukasz Mroziński)
Weryfikacja założeń techniczno-informatycznych	Zespół informatyków ds. integrowania e-materiałów pod względem technologicznym (Paweł, Tomaszek, Katarzyna Gagan, Anna Magdziarz-Tomaszek, Grzegorz Kusztełak)
Weryfikacja językowa	Alicja Berbeka
Rodzaj multimedium	wirtualna symulacja
Wykorzystanie AR lub VR AR - rozszerzona rzeczywistość VR - wirtualna rzeczywistość	standardowa 2D lub 3D <input type="checkbox"/> AR <input type="checkbox"/> VR
Etap(y) edukacyjny(e), dla których przeznaczony jest materiał	II etap: SP IV-VIII III etap: Liceum / technikum zakres podstawowy
Przedmiot(y), do nauki których przeznaczony jest materiał	fizyka informatyka



Fundusze Europejskie
dla Rozwoju Społecznego



Rzeczpospolita
Polska

Dofinansowane przez
Unię Europejską



2. Opis materiału

Skrócony opis materiału (abstrakt)

Symulacja składa się z następujących elementów:

- Analiza algorytmu zapisanego w **postaci listy kroków**. Użytkownik ma możliwość analizy i przetestowania tego algorytmu dla różnych wybranych samodzielnie zestawów danych (inspiracja magiczne bloczki). Następnie uczeń zapisuje ten algorytm w postaci programu w Pythonie, składając program z bloków instrukcji języka. Na każdym etapie tworzenia programu będzie mógł przetestować działanie programu.
- Analiza algorytmu zapisanego w **postaci pseudokodu**. Użytkownik ma możliwość analizy i przetestowania tego algorytmu dla różnych wybranych samodzielnie zestawów danych (inspiracja magiczne bloczki). Następnie uczeń zapisze ten algorytm w postaci programu w Pythonie, składając program z bloków instrukcji języka. Na każdym etapie tworzenia programu będzie mógł przetestować działanie programu.
- Analiza programu. Użytkownik otrzymuje specyfikację programu oraz przykładowy kod programu, który zawiera błąd logiczny. Zadaniem ucznia jest poprawienie błędu tak, by program działał zgodnie ze specyfikacją.
- Wskazanie różnych sposobów rozwiązania tego samego problemu. Analiza złożoności obliczeniowej algorytmów na konkretnych przykładach oraz ogólnie.
- Możliwość zapisania i przetestowania własnego algorytmu w postaci listy kroków/pseudokodu, a następnie zapisania go w postaci programu w Pythonie i testowanie dla dowolnych danych.

Cel ogólny materiału

Rozwijanie umiejętności rozwiązywania problemów na bazie logicznego i abstrakcyjnego myślenia, myślenia algorytmicznego oraz wykorzystanie pojęć i wielkości fizycznych do opisu zjawisk oraz wskazywanie ich przykładów w otaczającej rzeczywistości.

Cele z podstawy programowej kształcenia ogólnego możliwe do realizacji za pomocą materiału

Szkoła podstawowa

Informatyka

Uczeń:

- formułuje problem w postaci specyfikacji (czyli opisuje dane i wyniki) i wyróżnia kroki w algorytmicznym rozwiązywaniu problemów. Przedstawia algorytm za pomocą listy kroków;
- stosuje przy rozwiązywaniu problemów podstawowe algorytmy:
 - a) na liczbach naturalnych: bada podzielność liczb, wyodrębnia cyfry danej liczby, przedstawia działanie algorytmu Euklidesa,
 - b) wyszukiwania i porządkowania: wyszukuje element w zbiorze nieuporządkowanym oraz porządkuje elementy w zbiorze metodą przez proste wybieranie;
- przedstawia sposoby reprezentowania w komputerze, liczb naturalnych (system binarny);
- projektuje, tworzy i testuje programy w procesie rozwiązywania problemów. W programach stosuje: instrukcje wejścia/wyjścia, wyrażenia arytmetyczne i logiczne, instrukcje warunkowe, instrukcje iteracyjne, funkcje oraz zmienne i tablice. W szczególności programuje algorytmy z pkt. 2);

Fizyka

Uczeń:

- przelicza wielokrotności i podwielokrotności (mikro-, mili-, centy-, hekto-, kilo-, mega);
- przelicza jednostki czasu (sekunda, minuta, godzina);



Fundusze Europejskie
dla Rozwoju Społecznego



Rzeczpospolita
Polska

Dofinansowane przez
Unię Europejską



- posługuje się skalami temperatur (Celsjusza, Kelvina); przelicza temperaturę w skali Celsjusza na temperaturę w skali Kelvina i odwrotnie.

Szkoła ponadpodstawowa

Informatyka(zakres podstawowy)

Uczeń:

- planuje kolejne kroki rozwiązywania problemu, z uwzględnieniem podstawowych etapów myślenia komputacyjnego (określenie problemu, definicja modeli i pojęć, znalezienie rozwiązania, zaprogramowanie i testowanie rozwiązania);
- stosuje przy rozwiązywaniu problemów z różnych dziedzin algorytmy poznane w szkole podstawowej oraz algorytmy:
 - a) na liczbach: badania pierwszości liczby, zamiany reprezentacji liczb między pozycyjnymi systemami liczbowymi, działań na ułamkach z wykorzystaniem NWD i NWW,
 - b) na tekstach: porównywania tekstów, wyszukiwania wzorca w tekście metodą naiwną, szyfrowania tekstu metodą Cezara,
 - c) porządkowania ciągu liczb przez wstawianie i metodą bąbelkową,
 - d) obliczania wartości elementów ciągu metodą iteracyjną w tym wartości elementów ciągu Fibonacciego;
- sprawdza poprawność działania algorytmów dla przykładowych danych.

3. Charakterystyka materiału

Opis zawartości merytorycznej materiału

Język programowania Python

Pozycje w menu (do wyboru przez ucznia, niezależnie od siebie) - mogą być inaczej nazwane (bardziej atrakcyjnie):

- Wstęp do programowania.
- Algorytmy z podstawy programowej (tu rozróżnienie na szkołę podstawową i ponadpodstawową) zapisane za pomocą listy kroków.
- Algorytmy z podstawy programowej (tu rozróżnienie na szkołę podstawową i ponadpodstawową) zapisane za pomocą pseudokodu (pseudokod zgodny z zapisem w zadaniach w arkuszach maturalnych).
- Wykrywanie błędów w programach.
- Złożoność obliczeniowa.
- Własne algorytmy.

Dodatkowo multimedia powinno zawierać **w menu pomoc**, w której znajdują się **opisy instrukcji listy kroków, pseudokodu oraz Pythona, wraz z przykładami użycia**. Pomoc powinna być dostępna z **każdego** poziomu multimedia. Uczeń analizując pseudokod, może sprawdzić w pomocy, co realizuje wskazane polecenie.

Wstęp do programowania - zawiera proste algorytmy, których zadaniem jest pokazanie, jak działają podstawowe konstrukcje programistyczne:

- Algorytmy sekwencyjne np. obliczanie sumy dwóch (trzech) liczb, wyznaczenie średniej arytmetycznej, obliczanie pola powierzchni prostokąta, zamiana walut, np. złote na euro, przeliczanie jednostek np. st Celsjusza na Fahrenheita, wykorzystanie wzorów z fizyki (np. obliczanie prędkości, przebytej drogi itp), wyodrębnianie cyfry jedności (dziesiątek) liczby.
- Algorytmy z rozgałęzieniem np. badanie parzystości liczby, badanie podzielności, znajdowanie większej z dwóch (trzech liczb), realizacja funkcji sign, badanie czy osoba jest pełnoletnia.
- Algorytmy z pętlą np. wypisanie liczb parzystych z zadanego przedziału, obliczanie sumy



Fundusze Europejskie
dla Rozwoju Społecznego



Rzeczpospolita
Polska

Dofinansowane przez
Unię Europejską



ciągu zakończonego zerem, obliczanie sumy (średniej arytmetycznej) n liczb.

Algorytmy z podstawy programowej dla szkoły podstawowej:

- wyodrębnianie cyfr liczby, obliczanie ich sumy,
- algorytm Euklidesa (dwie wersje),
- znajdowanie min/max w liście elementów,
- porządkowanie elementów metodą wyboru.

Algorytmy z podstawy programowej dla szkoły ponadpodstawowej:

- porządkowanie elementów metodą bąbelkową, wstawiania,
- badanie pierwszości liczby,
- zamiany reprezentacji liczb między pozycyjnymi systemami liczbowymi,
- obliczanie wyrazów ciągu Fibonacciego,
- obliczanie elementów innych ciągów metodą iteracyjną,
- szyfr Cezara.

Kluczowe wymagania merytoryczne i dydaktyczne dla Wykonawcy materiału, które muszą zostać uwzględnione

Materiał musi zawierać ćwiczenia umożliwiające zrozumienie działania algorytmów zapisanych w postaci listy kroków, pseudokodu, języka programowania, przepływu sterowania w programie. Pozwolić na tworzenie własnego kodu.

W materiale ma być język prosty, precyzyjny dopasowany do odbiorcy, szczególnie w części realizacji algorytmów dla szkoły podstawowej i wstępu do programowania.

Wykonawca ma posiadać wiedzę z zakresu algorytmiki, testowania oprogramowania, szacowania złożoności obliczeniowej.

Opis struktury materiału

Pozycje menu 1., 2., 3.

Wszystkie ekrany są opracowane według tego samego schematu, np. podział ekranu na część lewą i prawą. W lewym oknie: podana specyfikacja problemu, a poniżej opis algorytmu za pomocą listy kroków/pseudokodu. Uczeń ma możliwość podania własnych danych. Jest możliwość uruchomienia całego algorytmu jednym przyciskiem, wtedy widać w jakiej kolejności, które bloki są wykonywane, aktywne bloki podświetlone, linie łączące bloki, po których "podróżują dane" również wyróżnione. Uczeń może uruchamiać algorytm krokowo, wówczas wyświetlają się po każdym kroku wartości zmiennych.

W prawym oknie umieszczone są instrukcje Pythona (w postaci blozków), z których uczeń może składać program (metodą ciągnij-upuść) i testować jego działanie, podobnie jak to jest w przypadku listy kroków. Jeżeli wynik programu jest błędny uczeń może zmodyfikować swój kod i powtórnie testować lub poprosić o odpowiedź (kliknąć w przycisk z odpowiedzią - tak jak w Khan Academy). Odpowiedzi powinny być dostosowane do stopnia zaawansowania kodu ucznia.

Uczeń może wybrać do analizy dowolny algorytm z listy. Nie musi realizować ich w kolejności.

Pozycja menu 4. (wykrywanie błędów)

W podmenu jest minimum 10 programów do analizy i poprawy. Uczeń może wybierać je niezależnie (nie jest wymagana kolejność rozwiązania).

W oknie z lewej strony jest zapisana specyfikacja programu.

Poniżej specyfikacji jest zapisany program w języku Python z błędem logicznym (wykonania).

Uczeń ma możliwość przetestowania tego programu dla wybranych przez siebie danych.



Fundusze Europejskie
dla Rozwoju Społecznego



Rzeczpospolita
Polska

Dofinansowane przez
Unię Europejską



W oknie z prawej strony uczeń ma miejsce na wpisanie poprawnego kodu programu i przetestowanie go. Jeśli uzna, że jego kod jest poprawny, klika w przycisk SPRAWDŹ. Otrzymuje wówczas informację zwrotną (dobrze/źle). Jest możliwość uzyskania odpowiedzi. Uczeń może dostać odpowiedź do zadania w przypadku, gdy wykona minimum 3 nieudane sprawdzenia i o nią poprosi np. klikając w przycisk ODPOWIEDŹ.

Pozycja menu 5. (złożoność obliczeniowa)

Zaprezentowanie wszystkich podanych algorytmów (lista poniżej) + 2 dodatkowo wybrane przez realizatora multimediu i ich analiza pod względem złożoności czasowej lub pamięciowej.

Prezentacja w postaci animacji. Dane do algorytmów powinny być tak dobrane, aby było widać korzyści ze stosowania rozwiązań o mniejszej złożoności.

Uczeń ma możliwość przetestowania algorytmów na własnych danych.

Okno podzielone na **cztery** części:

pasek górny: opis problemu wraz ze specyfikacją;

okno z lewej strony: program o większej złożoności;

okno z prawej strony: program o mniejszej złożoności;

pasek dolny: wnioski.

Lista algorytmów

LP	Nazwa / przeznaczenie algorytmu	Wersje
1.	Wyszukiwanie elementu w uporządkowanym zbiorze danych	1. Wyszukiwanie liniowe
		2. Wyszukiwanie binarne
2.	Testowanie czy liczba naturalna n jest liczbą pierwszą	1. Wyszukiwanie podzielników w przedziale $<2;n-1>$
		2. Wyszukiwanie podzielników do pierwiastka z liczby n
		3. Sito Eratostenesa - kiedy warto tablicować liczby pierwsze
3.	NWD	1. Wersja z odejmowaniem
		2. Wersja z dzieleniem
4.	Wyznaczanie spójnego podciągu o zadanej właściwości (np. najdłuższy rosnący)	1. Złożoność liniowa
		2. Złożoność kwadratowa
5.	Wyznaczanie liczb Fibonacciego	1. Wersja iteracyjna
		2. Wersja rekurencyjna
		3. Tablicowanie liczb. Przykłady problemów, dla rozwiązania których warto zastosować tablicowanie.
6.	Sortowanie	Przez wybór
		Szybkie / scalanie



Fundusze Europejskie
dla Rozwoju Społecznego



Rzeczpospolita
Polska

Dofinansowane przez
Unię Europejską



7.	Zamiana liczb binarnych na dziesiętne	Z definicji liczby w danym systemie pozycyjnym
		Schemat Hornera
8.	Znajdowanie min/max	1. Posortowanie niemalejąco i wybranie pierwszego/ostatniego elementu
		2. Algorytm liniowy

Pozycja menu 6. (własny algorytm)

Mechanika materiału

Interakcje w materiale odbywają się za pomocą menu, pól tekstowych oraz za pomocą przycisków. Uczeń może wybrać do analizy dowolny algorytm z menu. Ćwiczenia mogą być realizowane w dowolnej kolejności. Podział ekranu na okna. Każde okno zawiera środowisko typu sandbox. Uczeń ma możliwość zapisania własnego algorytmu w postaci pseudokodu i jego testowanie oraz zapis programu w Pythonie. Wygląd okna podobny jak w przypadku pkt 3. menu. Sposób zapisu pseudokodu, albo ręczne wprowadzenie, albo z gotowych bloczków, w których można uzupełniać warunki, zakres pętli itp.

Wybierają osobne przyciski, powiązane z zawartością znajdującą się w tej części ekranu.

Dla pozycji menu 1. , 2. , 3.

- W lewym oknie są dwa przyciski, służące testowaniu algorytmu: Uruchom i Uruchom krokowo.
- W lewym oknie należy również przewidzieć miejsce (np. pola tekstowe) na wprowadzanie danych przez użytkownika.
- W prawym oknie powinny znaleźć się bloczki (instrukcje języka Python), z których budowany jest program opisany w lewym oknie (metodą ciągnij-upuść), po najechaniu myszką na bloczek pojawia się jego opis.
- W prawym oknie są również dwa przyciski Sprawdź i Odpowiedź.

Dla pozycji menu 4., 6. - podobnie jak powyżej tylko dostosować inne nazwy przycisków.

Dla pozycji menu 5 - podział ekranu na 4 okna:

- pasek górny: opis problemu wraz ze specyfikacją, możliwość wpisania przez ucznia własnych danych (np. za pomocą pól tekstowych),
- okno z lewej strony: program o większej złożoności,
- okno z prawej strony: program o mniejszej złożoności
- pasek dolny: wnioski.

Grafika

Układ ekranu:

- Podział ekranu na dwie części: lewa i prawa, z wyraźnym, ale subtelnym wizualnym rozdzieleniem.
- Część lewa skupia się na prezentacji problemu i algorytmu.



Fundusze Europejskie
dla Rozwoju Społecznego



Rzeczpospolita
Polska

Dofinansowane przez
Unię Europejską



- Część prawa przeznaczona dla ucznia – zawiera polecenia i elementy do budowania kodu.
- Część lewa – specyfikacja problemu i algorytm:
- Górna sekcja: Opis problemu przedstawiony w sposób jasny i zwięzły.
 - Dolna sekcja: Algorytm przedstawiony za pomocą listy kroków lub pseudokodu, ułożony w czytelnej hierarchii.
 - Przyciski do uruchomienia pseudokodu umieszczone poniżej opisu algorytmu.
- Część prawa – działania ucznia:
- Górna sekcja: Polecenia dla ucznia sformułowane w sposób prosty i konkretny.
 - Dolna sekcja: Bloczki kodu do wyboru, odpowiednio ułożone i kategoryzowane.
 - Przyciski do uruchomienia programu widoczne i intuicyjnie rozmieszczone.
- Rozróżnienie kolorystyczne:**
- Część lewa i prawa mają różne kolory tła, które subtelnie rozróżniają sekcje, bez wprowadzania wizualnego chaosu.
 - Kolor czerwony jest zarezerwowany wyłącznie do wskazywania błędów, takich jak nieprawidłowo dobrane bloczki lub błędy w pseudokodzie.
- Minimalizm i przejrzystość:**
- Grafika bez zbędnych detali czy rozpraszających elementów.
 - Wszystkie teksty i ikony są wyraźne i odpowiednio skalowane, aby ułatwić czytanie i nawigację.
- Przyciski uruchamiania:**
- Przyciski „Uruchom pseudokod” i „Uruchom program” są w widocznych miejscach w każdej sekcji, aby uczniowie mogli testować swoje rozwiązania w czasie rzeczywistym.
- Interaktywność:**
- Elementy takie jak bloczki czy kroki algorytmu są możliwe do kliknięcia, przesuwania lub edytowania.

Przykładowe inspiracje

Magiczne bloczki (<https://erissoftware.pl/dla-szkoly/schematy-blokowe-magiczne-bloczki>)

Co czerpać?

Mechanizm śledzenia wartości zmiennych – użytkownik widzi, jak zmieniają się wartości w trakcie wykonywania algorytmu.

Wykonywanie krok po kroku – symulacja działania kodu w podziale na etapy.

Zastąpienie schematów blokowych alternatywnymi formami reprezentacji – zamiast klasycznego schematu blokowego, aplikacja stosuje listę kroków (lewe okno) lub pseudokod/bloki instrukcji w Pythonie (prawe okno).

Intuicyjny system nauki algorytmów – uczniowie mogą porównywać różne sposoby zapisu algorytmu w jednym widoku.

Scratch

Co czerpać?

Blokowa reprezentacja kodu – intuicyjne tworzenie programów przez przeciąganie i łączenie bloczków.

Podział na sekcję algorytmiczną i kodową – podobny układ do opisanego w scenariuszu.

Python Tutor

Co czerpać?

Animowana wizualizacja wykonywania kodu w Pythonie – użytkownik może krok po kroku zobaczyć, jak działa jego program.

Śledzenie wartości zmiennych – użytkownik widzi, jak zmieniają się wartości w kolejnych liniach



Fundusze Europejskie
dla Rozwoju Społecznego



Rzeczpospolita
Polska

Dofinansowane przez
Unię Europejską



kodu.

Khan Academy (programowanie)

Co czerpać?

System interaktywnych podpowiedzi – użytkownik dostaje wskazówki po kilku błędnych próbach rozwiązania zadania.

Stopniowe odkrywanie kolejnych podpowiedzi – zamiast od razu dawać rozwiązanie, system prowadzi użytkownika do właściwego wyniku.

4. Wymagania WCAG

Opis dostosowania materiału celem spełnienia standardu WCAG

Zaawansowany e-materiał musi uwzględniać założenia uniwersalnego projektowania w edukacji (UDL) oraz być zgodny ze standardami dostępności cyfrowej WCAG obowiązującymi na dzień ogłoszenia naboru, standardem ATAG 2.0 oraz zapisami ustawy z dnia 19 lipca 2019 r. o zapewnianiu dostępności osobom ze szczególnymi potrzebami (Dz. U. z 2019 r. poz. 1696) i ustawy z dnia 4 kwietnia 2019 r. o dostępności cyfrowej stron internetowych i aplikacji mobilnych podmiotów publicznych (Dz.U. z 2019 r. poz. 848). Powinien też uwzględniać dobre praktyki, stosowane w celu zapewnienia wysokiej jakości dostępnych cyfrowo materiałów edukacyjnych.

Użytkownik ze szczególnymi potrzebami, korzystający z przygotowanego zaawansowanego e-materiału, powinien korzystać z mechaniki materiału (menu nawigacyjnego) w taki sam sposób, jak wszyscy użytkownicy. Należy przygotować menu, w którym wybiera on dostosowania materiału do swoich potrzeb. W ramach wybranych dostosowań zaawansowanego e-materiału użytkownik powinien korzystać ze wszystkich zaprojektowanych funkcjonalności. Zaawansowany e-materiał powinien spełniać kryteria dostępu dla technologii dotykowych (np. ekranów dotykowych), dostępności z poziomu klawiatury czy za pomocą zewnętrznych urządzeń wejściowych (np. mysz powiększona), technologii asystujących (np. czytniki ekranu). Poszczególne ułatwienia dostępu oraz ich konfiguracja powinny być dostępne w menu przed uruchomieniem aplikacji. Powinna istnieć również możliwość zapamiętania wybranych przez użytkownika ustawień, tak aby mogła być stosowana przy kolejnych uruchomieniach aplikacji przez użytkownika.

Zaawansowany e-materiał powinien spełniać następujące kryteria:

1. umożliwiać użytkownikowi z różnymi potrzebami korzystać z ułatwień dostępu, na wszystkich poziomach i etapach e-materiału;
2. posiadać instrukcję dla użytkowników z różnymi potrzebami, zawierającą informacje o sposobie korzystania z ułatwień dostępu i mechanizmach poruszania się po menu, przygotowaną za pomocą tzw. prostego języka;
3. posiadać rozwiązania z zakresu dostępności, które pozwalają uniknąć QTE lub działań związanych z łączeniem przycisków (uwzględnia ustawienie pozwalające je uprościć lub pominąć/wyłączyć);
4. umożliwiać korzystanie z wirtualnej klawiatury ekranowej (jeśli materiał tego wymaga), którą można sterować za pomocą myszy lub technologii wspomagających, takich jak wzrok lub przełącznik;
5. umożliwiać skorzystanie z pomocy w sytuacjach potencjalnie trudnych, związanych z poruszaniem się po materiale;
6. użytkownik przed skorzystaniem z zaawansowanego e-materiału powinien mieć możliwość zapoznania się tutorialiem objaśniającym, jak korzystać z ułatwień dostępu;
7. mechanika zaawansowanego e-materiału powinna pozwalać na dostęp do wszystkich obszarów interfejsu użytkownika;
8. zaawansowany e-materiał powinien być dostępny za pomocą technologii asystujących,



Fundusze Europejskie
dla Rozwoju Społecznego



Rzeczpospolita
Polska

Dofinansowane przez
Unię Europejską



m.in. czynników ekranu, oprogramowania asystującego w technologiach mobilnych.

Jeżeli w materiale będą występowały treści nieinterpretowalne przez technologie asystujące, wykonawca zobowiązany jest zapewnić alternatywę wchodzącą w e-materiał i stanowiącą integralną całość zaawansowanego e-materiału. Bez konsultacji z ekspertami ORE nie dopuszcza się tworzenia alternatywnego (równoległego rozwiązania) dedykowanego osobom z różnymi potrzebami.

Zaawansowany e-materiał musi uwzględniać między innymi potrzeby osób:

- z ograniczeniami wzroku,
- z ograniczeniami słuchu,
- z ograniczeniami ruchu rąk i mobilności,
- z ograniczeniami możliwości poznawczych (związanymi z np. pamięcią, przetwarzaniem informacji, dysleksją),
- z zaburzeniami neurorozwojowymi i psychicznymi (np. spektrum autyzmu, ADHD, stanami lękowymi, epilepsją),
- z zaburzeniami mowy,
- korzystających z czynników ekranu.

Podczas projektowania e-materiału należy uwzględniać różne potrzeby i możliwości użytkowników ze względu na:

Ograniczenia wzroku:

- stosowanie dobrze kontrastujących kolorów, czytelnych rozmiarów i typów fontów, możliwość zmiany i indywidualnego dopasowania przez użytkownika tych elementów;
- stosowanie zawsze widocznego fokusa (przynajmniej częściowo);
- używanie kombinacji koloru, kształtów i tekstu, niestosowanie znaczenia tylko kolorem;
- umieszczanie przycisków i powiadomień w kontekście;
- stosowanie odpowiedniej wielkości, kolorów i rozmieszczenia elementów interfejsu;
- umożliwienie zmiany kolorów dla osób będących daltonistami;
- umożliwienie zmiany wielkości elementów interfejsu;
- używanie dźwięku przestrzennego i rozróżnialnych dźwięków, różnych w zależności od zdarzeń;
- umożliwienie wyboru wyglądu kursora/celownika, zmiany kształtu, wielkości, koloru, jeśli projektowana mapa interaktywna zakłada bardzo dużo obiektów;
- wyświetlanie istotnych informacji w centrum, na linii wzroku lub możliwość powiększania całości, poszczególnych elementów mapy interaktywnej;
- nawigacja i sterowanie za pomocą klawiatury;
- stosowanie tekstów alternatywnych lub audiodeskrypcji do grafik;
- elementy materiału powinny być duże i łatwe do odróżnienia oraz oddalone od siebie;
- dodanie opisów alternatywnych do obrazów i innych elementów wizualnych, które opisują treści lub funkcje;
- stosowanie dużego kontrastu między istotnymi elementami w materiale;
- użytkownicy niewidomi powinni móc skorzystać z każdej funkcjonalności materiału z poziomu klawiatury.

Ograniczenia słuchu:

- stosowanie prostego języka, niestosowanie figur stylistycznych i idiomów;
- zapewnienie alternatywy tekstowej każdej kluczowej informacji dźwiękowej;
- dodanie napisów i transkrypcji do treści audio i wideo;
- możliwość modyfikacji napisów, zmiana rozmiaru/koloru oraz ich włączania i wyłączania zanim pojawi się dźwięk;
- stosowanie napisów rozszerzonych informujących o dodatkowych dźwiękach i nastroju oraz postaci mówiących;
- stosowanie prostych logicznych i spójnych układów treści;



Fundusze Europejskie
dla Rozwoju Społecznego



Rzeczpospolita
Polska

Dofinansowane przez
Unię Europejską



- zapewnienie możliwości osobnej regulacji dźwięku dla różnych elementów multimedialnych w mapie interaktywnej;
- zastosowanie przełącznika dźwięku mono/stereo w materiałach filmowych i audio (jeśli takie się pojawią w zaawansowanym materiale).

Ograniczenia ruchu rąk i mobilności:

- umożliwienie w menu materiału ustawienia dużych obszarów klikalnych;
- projektowanie obsługi za pomocą klawiatury i mowy;
- unikanie tworzenia dynamicznych treści, wymagających dużego ruchu myszy;
- nieograniczanie czasu otwarcia okien, wykonania zadań;
- zapewnienie alternatywy dla akcji, wymagających równoczesnych czynności (np. klik zamiast przeciągnij i upuść);
- zapewnienie sterowania przy użyciu prostych kontrolerów.
- unikanie stosowania bardzo precyzyjnych ruchów.

Ograniczenia poznawcze oraz zaburzenia neurorozwojowe i psychiczne:

- używanie prostych, stonowanych barw;
- używanie prostego języka, bez stosowania figur stylistycznych i idiomów;
- używanie krótkich zdań i punktowania;
- używanie wyjaśnienia skrótów;
- tworzenie opisowych przycisków;
- budowanie prostych i spójnych układów treści;
- wyrównanie tekstów do lewej i zachowanie spójnego układu;
- niestosowanie dużych bloków ciężkiego tekstu;
- niestosowanie podkreślania słów, niepochylania tekstu i pisanie wielkimi literami;
- umożliwienie zmiany kontrastu pomiędzy tłem a tekstem;
- niestosowanie ograniczenia czasowego na wykonanie zadania;
- niestosowanie presji czasowej lub związanej z możliwością wykonania tylko jednej próby wykonania zadania.

Ograniczenia związane z korzystaniem z czytników ekranów:

- opisywanie obrazów, stosownie transkrypcji, audiodeskrypcji;
- nieumieszczanie informacji tylko na obrazie lub wideo;
- nadawanie struktury treści i nieoznaczanie jej tylko rozmiarem i rozmieszczeniem tekstu;
- stosowanie liniowego logicznego układu;
- umożliwienie sterowania za pomocą klawiatury;
- tworzenie opisowych łączy.

Powyższe wytyczne są jedynie przykładami potrzeb, jakie powinny zostać spełnione przy projektowaniu zaawansowanego e-materiału. Beneficjent konkursowy powinien zapewnić możliwie największą dostępność dla osób z różnymi potrzebami. Rozwiązania związane z zapewnieniem dostępności osobom z różnymi potrzebami Beneficjent konkursowy powinien konsultować z ekspertami ORE na poszczególnych etapach realizacji projektu konkursowego.



Fundusze Europejskie
dla Rozwoju Społecznego



Rzeczpospolita
Polska

Dofinansowane przez
Unię Europejską



5. Wymagania funkcjonalne i techniczne

Kluczowe warunki funkcjonalne dla Wykonawców

Aplikacja musi spełniać wymagania określone w dokumencie „Ogólne wymagania funkcjonalne i techniczne dla e-materiałów”.

Ekran startowy:

Z możliwością niezależnego wyboru modułów.:

- o Wstęp do programowania:
- o Podmenu z prostymi algorytmami, pokazującymi podstawowe konstrukcje programistyczne (np. sekwencyjne, z pętlą, rozgałęzieniem).
- Algorytmy z podstawy programowej z rozróżnieniem na:
 - o szkołę podstawową.
 - o szkołę ponadpodstawową.

Algorytmy zapisane w formie listy kroków

- Algorytmy z podstawy programowej z rozróżnieniem na:
 - o Szkołę podstawową.
 - o Szkołę ponadpodstawową.
 - Algorytmy zapisane w formie pseudokodu zgodnym z zapisem w zadaniach w arkuszach maturalnych.

Realistyczna symulacja i interaktywność

- Dokładne odwzorowanie procesów:
 - o Symulacja algorytmów oparta na przykładach z podstawy programowej:
 - o Algorytmy sekwencyjne (np. obliczanie sumy dwóch liczb, zamiana walut, obliczanie prędkości). Algorytmy z pętlą (np. wypisywanie liczb parzystych, obliczanie sumy n liczb).
 - o Algorytmy z rozgałęzieniem (np. badanie podzielności liczby, badanie czy osoba jest pełnoletnia).
 - o Wykrywanie błędów w kodzie: symulacja programu w Pythonie zawierającego błędy logiczne, które uczeń musi poprawić zgodnie ze specyfikacją.
 - o Interaktywne elementy:
 - o Edytor do budowania programów w Pythonie z blozków: Przeciąganie i łączenie instrukcji takich jak "if", "for", "while" w celu zapisania algorytmu w postaci programu.
 - o Możliwość podania własnych danych do testowania algorytmu:
 - o Wprowadzenie danych wejściowych do algorytmów (np. liczby do posortowania w metodzie bąbelkowej).
 - o Wybór sposobu uruchamiania algorytmu:
 - Tryb krokowy z podświetlaniem aktywnych bloków i przepływu danych.

Nawigacja i opcje wyświetlania

- Swobodne przemieszczanie się po symulacji:
 - o Menu główne z pozycjami umożliwiającymi wybór:
 - „Wstęp do programowania”.
 - „Algorytmy z podstawy programowej” (osobne dla szkół podstawowych i ponadpodstawowych).
 - „Wykrywanie błędów w programach”.
 - „Złożoność obliczeniowa”.
 - „Własne algorytmy”.



Fundusze Europejskie
dla Rozwoju Społecznego



Rzeczpospolita
Polska

Dofinansowane przez
Unię Europejską



- Użytkownik może analizować algorytmy w dowolnej kolejności, bez narzuconego porządku.
- Tryby wyświetlania i perspektywy:
- Dwa widoki pracy:
 - Widok pseudokodu i listy kroków.
 - Widok kodu w Pythonie.
- Opcja porównania różnych algorytmów w jednym oknie (np. pod względem złożoności czasowej i pamięciowej).

Scenariusze i poziomy trudności

- Scenariusze edukacyjne:
 - Predefiniowane scenariusze edukacyjne, takie jak:
 - "Obliczanie sumy liczb".
 - "Znajdowanie największej liczby w liście".
 - "Porównanie metod sortowania".
 - Każdy scenariusz zawiera opis problemu, pseudokod oraz przykładowy zestaw danych wejściowych.
- Dostosowywane poziomy trudności:
 - Tryby: "Podstawowy", "Zaawansowany" oraz "Ekspert", dostosowujące stopień skomplikowania zadania.
 - Zróżnicowanie zadań:
 - Proste algorytmy dla szkoły podstawowej (np. algorytm Euklidesa).
 - Zaawansowane algorytmy dla szkoły ponadpodstawowej (np. sito Eratostenesa).

System testowania wiedzy i zadania interaktywne

- Analiza algorytmów: zapisanych w formie listy kroków; zapisanych w formie pseudokodu. Budowanie programu w Pythonie na podstawie listy kroków lub pseudokodu, z użyciem blochków. Wykrywanie i poprawa błędów w programach. Analiza i porównanie złożoności obliczeniowej algorytmów. Tworzenie własnych algorytmów. Ćwiczenia z algorytmów z podstawy programowej.
- Zadania interaktywne: testowanie algorytmów na własnych danych.
- Funkcja podpowiedzi z opisem rozwiązania dostępna po trzech nieudanych próbach.

Śledzenie postępów i zapisywanie wyników

- Historia działań użytkownika:
 - Rejestrowanie wykonanych symulacji, w tym:
 - Uruchomionych algorytmów (np. sortowanie bąbelkowe).
 - Wyników testów i quizów.
 - Możliwość zapisania postępów i kontynuacji pracy od ostatniego etapu.
- Profilowanie wyników i osiągnięć:
 - System oceniania na podstawie:
 - Poprawności rozwiązania zadań (np. poprawne sortowanie danych).
 - Wyników quizów i ćwiczeń praktycznych.
 - Generowanie raportów z podsumowaniem wyników.

Personalizacja przez nauczyciela

- Dostosowanie parametrów symulacji:
 - Możliwość definiowania warunków początkowych:
 - Nauczyciel ustawia zestaw danych dla algorytmu sortowania.



Fundusze Europejskie
dla Rozwoju Społecznego



Rzeczpospolita
Polska

Dofinansowane przez
Unię Europejską



- Ograniczenie dostępnych funkcji:
 - Skupienie na wybranym zagadnieniu, np. jedynie algorytmy iteracyjne.
- Tworzenie scenariuszy i zadań edukacyjnych:
 - Edytor dla nauczyciela:
 - Dodawanie własnych algorytmów (np. nowych metod sortowania).
 - Tworzenie pytań quizowych związanych z konkretnym algorytmem.
- System powinien umożliwiać użytkownikowi zapisanie i wczytanie własnych ustawień algorytmów i programów, co pozwala na powrót do wcześniejszych decyzji i analizowanie wyników długoterminowych.

Mechanika aplikacji

- Interakcje użytkownika:
 - Przykłady przycisków i funkcji:
 - „Uruchom” algorytm sortowania metodą wyboru.
 - „Sprawdź” poprawność zapisu pseudokodu dla algorytmu Euklidesa.
 - Tryby sandbox:
 - Budowanie programu w Pythonie z bloczków.
 - Możliwość testowania działania algorytmu na własnych danych.

Grafika i projekt interfejsu

- Przejrzystość:
 - Układ z wyraźnym podziałem:
 - Lewe okno: specyfikacja i pseudokod algorytmu.
 - Prawe okno: bloczki języka Python do budowy programu.
 - Podświetlanie aktywnych elementów w trakcie działania algorytmu.
- Pomoc kontekstowa pomoc

Kluczowe warunki techniczne dla Wykonawców

Aplikacja musi spełniać wymagania określone w dokumencie „Ogólne wymagania funkcjonalne i techniczne dla e-materiałów”.

Raportowanie i statystyki:

- System raportowania wyników dla nauczycieli: Funkcja umożliwiająca nauczycielom monitorowanie wyników i postępów uczniów w ćwiczeniach i zadaniach związanych z symulacją.
- Podsumowanie wyników dla użytkownika: Po zakończeniu sesji użytkownik powinien mieć możliwość przeglądania swoich wyników, co wspiera proces nauki i identyfikacji obszarów wymagających powtórzenia.



Fundusze Europejskie
dla Rozwoju Społecznego



Rzeczpospolita
Polska

Dofinansowane przez
Unię Europejską

