Informatyka w Edukacji, XXI
UMK Toruń, 2025

MATERIAŁY EDUKACYJNE Z INFORMATYKI – JAK UCZYĆ ZŁOŻONYCH ZAGADNIEŃ W JAK NAJPROSTSZY SPOSÓB

Damian Kurpiewski
Wydział Matematyki i Informatyki UMK,
Instytut Podstaw Informatyki PAN
d.kurpiewski@proton.me; blackbat13.github.io

Abstract. *This article presents an educational platform offering free, Polish-language computer science resources for teachers and students. It features lesson plans, interactive projects (e.g., Python game development), exam prep materials, and visualized algorithms with code examples in C++ and Python. Designed to bridge technical depth with pedagogical clarity, it supports diverse teaching formats (in-person, remote), enhancing classroom engagement and computational thinking.*

1. Wstęp

W epoce cyfrowej dostęp do wysokiej jakości materiałów edukacyjnych z zakresu informatyki staje się nie tylko potrzebą, ale obowiązkiem edukacyjnym. W sieci można znaleźć liczne zasoby skierowane zarówno do uczniów, jak i nauczycieli, jednak wiele z nich napotyka na poważne ograniczenia. Część jest dostępna wyłącznie w języku angielskim, inne są przeznaczone wyłącznie dla wąskiej grupy wiekowej, a kolejne wymagają zaawansowanej wstępnej wiedzy lub opłat za dostęp. Równie istotnym problemem jest brak materiałów, które łączą wiedzę informatyczną z praktycznymi metodami jej przekazywania. Tego typu wady często prowadzą do zniechęcenia zarówno uczniów, jak i nauczycieli do podejmowania próby nauki lub wykorzystania tych zasobów.

Szczególnie w języku polskim, mimo istnienia wartościowych opracowań, widoczna jest luka w zakresie bezpłatnych materiałów dydaktycznych, które byłyby jednocześnie:

- **uniwersalne** – dostosowane do różnych poziomów edukacji (szkoła podstawowa, średnia, studia),
- **przystępne** – nie wymagające zaawansowanej wiedzy wstępnej,
- **praktyczne** – zawierające zadania, schematy, ilustracje i wskazówki metodyczne,
- **profesjonalne** – opracowane z myślą o nowoczesnej dydaktyce informatyki.

Obserwuję, że większość dostępnych zasobów powstaje w dwóch głównych kierunkach:

1. **Autorzy z dziedziny edukacji** – nauczyciele, którzy mają głęboką wiedzę pedagogiczną, ale często ograniczoną wiedzę techniczną. Ich materiały są dobrze strukturalne, ale mogą być powierzchowne lub opierać się wyłącznie na standardowych podręcznikach.
2. **Specjaliści informatyczni** – programiści, inżynierowie i naukowcy, którzy mają bogatą wiedzę techniczną, ale często nie posiadają umiejętności tłumaczenia skomplikowanych zagadnień w sposób zrozumiały dla osób z różnych poziomów zaawansowania.

W efekcie, mimo rosnącej liczby publikacji i zasobów online, brakuje materiałów, które łączyłyby te dwa światy: głęboką wiedzę informatyczną z efektywną metodologią nauczania.

1.1. Motywacja i cel projektu

Jako nauczyciel informatyki z doświadczeniem na wszystkich etapach edukacji, a także absolwent studiów magisterskich i inżynierskich z informatyki, postanowiłem stworzyć zasób, który wypełni tę lukę. Pracując jednocześnie jako wykładowca akademicki i prowadząc własny projekt badawczy w zakresie informatyki, zauważyłem, że potrzebny jest nowoczesny, otwarty i praktyczny zestaw materiałów dydaktycznych. Mój projekt, nazwany **CS HTiEW (Computer Science: Hard Topics in Easiest Way)**, ma na celu dostarczenie zasobów, które są:

- **proste i intuicyjne,**
- **zwięzłe** (bez zbędnych dygresji),
- **wielopoziomowe** (dostosowane do różnych grup odbiorców),
- **kompleksowe** (obejmujące prezentacje, zadania, filmy i plany lekcji).

Nazwa projektu odzwierciedla jego podstawową filozofię: skupienie się na trudnych, ale kluczowych zagadnieniach informatyki, przedstawianych w najprostszy możliwy sposób.

1.2. Struktura artykułu

W niniejszym artykule skupię się na konkretnych materiałach dostępnych na mojej platformie edukacyjnej (https://edu.cs-htiew.pl), prezentując ich zakres i potencjalne zastosowanie w praktyce szkolnej. Omówię przykłady treści, które mogą być szczególnie przydatne dla nauczycieli informatyki oraz uczniów na różnych etapach edukacji. W kolejnych rozdziałach przedstawię gotowe scenariusze lekcji, prezentacje multimedialne i zadania do wykorzystania w klasie, dostosowane do podstawy programowej. Zaprezentuję materiały wspierające samodzielną naukę, w tym filmy instruktażowe,

interaktywne zadania programistyczne oraz zestawy ćwiczeń z rozwiązaniami. Podkreślę, jak te zasoby mogą pomóc w zrozumieniu zagadnień, które często sprawiają trudności

1.3. Uwaga o jakości i informacja zwrotna

Mimo że projekt jest rozwijany od kilku lat, nadal pozostaje w fazie dynamicznego rozwoju. Jako jego jedyny aktywny współtwórca, zdaję sobie sprawę z możliwości wystąpienia błędów, niedoskonałości lub luk w materiałach. Wszelkie uwagi, sugestie czy krytyka są mile widziane – można je kierować na mój adres mailowy podany w zakładce „Kontakt" na stronie projektu.

2. Materiały dla nauczycieli

Jak już wspomniałem we wstępie, większość zasobów dostępnych na mojej platformie (https://edu.cs-htiew.pl) jest skierowana do szerokiego grona odbiorców: uczniów, studentów, samouków i nauczycieli. Istnieją jednak sekcje, które zostały zaprojektowane specjalnie z myślą o nauczycielach informatyki, wspierając ich codzienną pracę w klasie i pozwalając oszczędzić czas na przygotowaniach. W tym rozdziale skupię się na dwóch najważniejszych z nich: **„Scenariusze zajęć"** oraz **„Dydaktyka Informatyki"**, omawiając ich zawartość i potencjalne zastosowanie w praktyce edukacyjnej.

2.1. Scenariusze zajęć – gotowe kompletne zestawy do lekcji informatyki

Sekcja **„Scenariusze zajęć"** to rezultat współpracy z dr Krzysztofem Skowronkiem – emerytowanym nauczycielem informatyki i autorem wielu publikacji dydaktycznych [1,2,3,4]. Materiały te są przeznaczone dla nauczycieli, którzy szukają gotowych, sprawdzonych rozwiązań do przeprowadzenia kilku lekcji poświęconych konkretnemu, często trudnemu zagadnieniu z zakresu algorytmiki i programowania.

Każdy zestaw scenariuszy obejmuje pełen cykl lekcji (zazwyczaj 3–4 godziny lekcyjne), który można łatwo dostosować do konkretnego poziomu edukacji (szkoła podstawowa, gimnazjum, liceum, technikum). Struktura każdego zestawu została zaprojektowana tak, by zapewnić płynne przejście od teorii do praktyki, a także uwzględnia różne style uczenia się uczniów.

Budowa scenariusza

- **Lekcja 1: Wprowadzenie bez komputera (unplugged)**
  Pierwsza lekcja skupia się na zrozumieniu podstawowego pojęcia lub mechanizmu bez użycia technologii. Na przykład:

  - **Zmienne** – uczniowie umieszczają kubeczki w fizycznym, uproszczonym modelu pamięci komputera, aby zrozumieć zasadę funkcjonowania zmiennych.
  - **Rekurencja** – uczniowie przeszukują zagnieżdżone w sobie pudełka, by zasymulować działanie algorytmu przeszukiwania w głąb w praktyce.

    W tym etapie nauczyciel otrzymuje gotowe ćwiczenia praktyczne, wskazówki metodyczne oraz materiały pomocnicze (np. karty pracy, karty instrukcji dla uczniów).
- **Lekcja 2: Wprowadzenie do programowania**
  Druga lekcja przenosi zagadnienie do świata cyfrowego, wykorzystując zarówno programowanie blokowe (Blockly), jak i języki tekstowe (Python, C++). Przykłady:
  - **Algorytmy** – uczniowie wykonują ćwiczenia z Blockly Labirynt by za pomocą sekwencji instrukcji stworzyć algorytm wyjścia z labiryntu.
  - **Funkcje** – uczniowie implementują funkcje obliczające różne wartości, jak np. NWD.
- **Lekcja 3: Ewaluacja i konsolidacja wiedzy**
  Ostatnia lekcja ma charakter diagnozujący i utrwalający. Uczniowie rozwiązują zadania o zróżnicowanym poziomie trudności, które są oparte na wcześniej poruszanych zagadnieniach, ale przedstawiają problem w nowym świetle, co ma na celu sprawdzenie poziomu zrozumienia tematu, a nie wyuczonej na pamięć wiedzy.

Zalety scenariuszy

- **Kompletność:** każdy zestaw zawiera konspekty lekcji, prezentacje multimedialne, zadania, rozwiązania oraz potrzebne materiały dodatkowe
- **Elastyczność:** scenariusze można dostosować do czasu trwania lekcji (np. skrócić do 2 godzin lub rozszerzyć do 5).
- **Zaoszczędzenie czasu:** nauczyciel nie musi samodzielnie tworzyć materiałów – wszystko jest gotowe do wykorzystania.
- **Zgodność z podstawą programową:** materiały są dostosowane do wymagań edukacyjnych dla informatyki na różnych etapach kształcenia.

2.2. Dydaktyka Informatyki – inspiracje i narzędzia do samodoskonalenia

Sekcja **„Dydaktyka Informatyki”** to zbiór prezentacji, które pomagają nauczycielom pogłębić wiedzę na temat efektywnych metod nauczania informatyki i wyzwań z tym związanych. Materiały te są szczególnie przydatne w kontekście ciągłego rozwoju

zawodowego oraz inspiracji do tworzenia własnych lekcji. Znaleźć można tutaj omówienie podstawy programowej, ze szczególnym uwzględnieniem treści algorytmicznych i programistycznych i tego, jak się przekładają na konkretne zagadnienia związane z programowaniem. Dostępne jest także omówienie różnych metod i sposobów nauczania programowania w szkole na poszczególnych etapach, ze wskazaniem konkretnych narzędzi, języków programowania, ich potencjalnych zalet i wad. Nauczyciele zainteresowani tym, jak uczniowie bawią się kosztem innych na lekcjach, mogą zapoznać się z różnymi technikami stosowanymi przez uczniów by nabierać swoich nauczycieli informatyki oraz koleżanki i kolegów z klasy.

Z tematów, które można bezpośrednio wykorzystać na lekcjach, znaleźć można np. omówienie historii otwartego oprogramowania, jego rozwoju z biegiem lat czy znaczenia we współczesnym świecie. Jeden z tematów porusza także zagadnienie grywalizacji w kontekście nauki zaawansowanych algorytmów. Przedstawione są gry mające na celu aktywizację uczniów i umożliwienie im lepszego zrozumienia wybranych problemów algorytmicznych.

Podsumowując, sekcja ta zawiera najróżniejsze prezentacje, które mogą posłużyć za inspirację do przeprowadzenia lekcji i zmotywować do samodzielnej nauki.

3. Materiały dla uczniów

Materiały dostępne na mojej platformie edukacyjnej (https://edu.cs-htiew.pl) są zaprojektowane tak, aby wspierać samodzielną naukę uczniów na różnych etapach edukacji. Dwa główne obszary skupiające szczególne zainteresowanie to:

- Nauka programowania poprzez tworzenie gier – sekcja, która łączy kreatywność z praktycznymi umiejętnościami programistycznymi.
- Przygotowanie do matury z informatyki – zbiór zadań i rozwiązań dostosowanych do wymagań egzaminacyjnych.

Oba te podejścia nie tylko odpowiadają na potrzeby uczniów, ale mogą być również efektywnie wykorzystane przez nauczycieli w pracy na lekcjach.

3.1. Nauka programowania poprzez tworzenie gier

Jednym z najskuteczniejszych sposobów na przyswajanie wiedzy programistycznej jest praktyczne jej zastosowanie. W sekcji **„Nauka programowania poprzez tworzenie gier”** uczniowie krok po kroku tworzą proste, ale angażujące gry dwuwymiarowe w języku **Python** z wykorzystaniem biblioteki **Pygame Zero**. Ta biblioteka została specjalnie zaprojektowana, aby uprościć naukę tworzenia gier, eliminując złożoność związane z grafiką i fizyką, co pozwala skupić się na logice programowania.

Struktura projektów

Każdy projekt gry składa się z następujących elementów:

- **Opis problemu i celu gry** – krótkie wprowadzenie, które wyjaśnia, jak gra działa i jakie umiejętności programistyczne zostaną wykorzystane.
- **Krokowe instrukcje** – szczegółowe etapy tworzenia gry, od inicjalizacji projektu po finalne funkcje. Przykład: W grze „Łapanie słoneczka" pierwszym krokiem jest wyświetlenie grafiki słoneczka na ekranie, a kolejnym – dodanie mechanizmu losowego przemieszczania się obiektu.
- **Kod źródłowy** – każdy fragment kodu jest pokazany i opisany, a także dostępny do pobrania na stronie.
- **Dodatkowe wyzwania** – po stworzeniu podstawowej wersji gry uczniowie mogą rozszerzyć funkcjonalność, np. dodać licznik punktów, poziomy trudności lub nowe mechaniki.

Przykładowe gry dostępne na stronie

- Gra w łapanie słoneczka
    - **Opis**: słoneczko przemieszcza się losowo po ekranie, a zadaniem gracza jest kliknąć na nie jak najczęściej.
    - **Zastosowane koncepcje**: obsługa zdarzeń myszy, generowanie losowych współrzędnych, aktualizacja stanu gry.
    - **Rozszerzenia**: dodanie czasomierza, systemu punktacji, grafik animowanych.
- Głodna świnia
    - **Opis**: świnia porusza się po ekranie, zbiera buraki i przyspiesza z każdym zdobytym punktem.
    - **Zastosowane koncepcje**: kontrola ruchu, kolizje obiektów, zmienne dynamiczne (np. prędkość).
    - **Rozszerzenia**: dodanie przeszkód, systemu życia, dźwięków.
- Kopia gry „Flappy Bird"
    - **Opis**: gracz kontroluje ptaka, który musi unikać przeszkód (rur), klikając, aby utrzymać się w powietrzu.
    - **Zastosowane koncepcje**: fizyka ruchu (przyspieszenie grawitacji), kolizje, przesuwanie tła.
    - **Rozszerzenia**: dodanie animacji ptaka, rankingów, trybu wieloosobowego.

Zalety tej metody

- **Motywacja:** gry są ciekawe i angażujące, co sprzyja długotrwałej nauce.

- **Praktyczne umiejętności:** uczniowie uczą się programowania w kontekście realnego projektu, co ułatwia zrozumienie abstrakcyjnych pojęć (np. pętle, zmienne, funkcje).
- **Dostępność:** wszystkie grafiki są pobierane z licencji open-source, co eliminuje barierę techniczną związane z tworzeniem własnych zasobów graficznych.
- **Elastyczność:** projekty można dostosować do poziomu zaawansowania ucznia – od podstawowych zadań po rozbudowane mechaniki.

Zastosowanie w pracy nauczyciela

Nauczyciele mogą wykorzystać te projekty jako:

- **Zadania domowe** – np. stworzenie gry „Flappy Bird" jako projekt końcowy semestru.
- **Warsztaty programistyczne** – organizacja konkursów na najlepsze rozszerzenie gry.
- **Wprowadzenie do programowania** – wykorzystanie prostszych gier (np. „Łapanie słoneczka") jako punktu startowego dla początkujących.

3.2. Przygotowanie do matury z informatyki

Egzamin maturalny z informatyki wymaga nie tylko znajomości teorii, ale także umiejętności praktycznego rozwiązywania zadań. W sekcji „Przygotowanie do matury" uczniowie znajdują:

- **Zadania z arkusza kalkulacyjnego** – analiza danych, formuły, wykresy.
- **Bazy danych (SQL i Access)** – tworzenie zapytań, relacji.
- **Algorytmika na kartce** – projektowanie algorytmów, analiza złożoności.
- **Programowanie w Pythonie i C++** – implementacja rozwiązań do typowych problemów maturalnych.

Struktura zadań

Każde zadanie zostało opracowane zgodnie z formatem i poziomem trudności matury, a także zawiera:

- **Opis problemu** – jasne sformułowanie zagadnienia, np. „Zaprojektuj algorytm, który znajdzie największy wspólny dzielnik dwóch liczb".
- **Wzorcowe rozwiązanie** – kod źródłowy lub szczegółowy opis kroków.
- **Filmy instruktażowe** – dla wybranych zadań dostępne są nagrania pokazujące krok po kroku, jak dojść do rozwiązania.

Dodatkowe zasoby

- **Lista kluczowych zagadnień** – np. najważniejsze algorytmy, jakie formuły arkusza kalkulacyjnego są przydatne.
- **Rozwiązania zadań z poprzednich matur** – analiza typowych problemów i sposobów ich rozwiązywania.

Zalety tej sekcji

- **Sprawdzenie w boju**: zadania zostały przetestowane w praktyce – wykorzystywałem je na zajęciach z uczniami.
- **Wielopoziomowość**: od prostych ćwiczeń dla początkujących po rozbudowane zadania dla zaawansowanych.
- **Wizualizacja**: schematy blokowe, grafiki i filmy wspierają zrozumienie skomplikowanych pojęć.
- **Dostępność**: wszystkie materiały są bezpłatne i dostępne online, co eliminuje barierę finansową.

Zastosowanie w pracy nauczyciela

Nauczyciele mogą wykorzystać te zasoby jako:

- **Materiały do powtórek przed maturą** – np. zestawy zadań z arkusza kalkulacyjnego.
- **Przykłady rozwiązań** – pokazanie uczniów, jak analizować i rozwiązywać zadania maturalne.
- **Testy diagnozujące** – wykorzystanie zadań jako quizów lub kartkówek.

4. Materiały ogólne – algorytmy dla wszystkich

Na mojej stronie znaleźć można także uniwersalne materiały dotyczące algorytmów, przeznaczone zarówno dla uczniów, jak i nauczycieli. Główny nacisk położony został na klarowną prezentację zagadnień z podstawy programowej (np. sortowanie, przeszukiwanie), ale również tematy wykraczające poza nią.

Cechy wyróżniające:

- **Przystępne tłumaczenie**: każdy algorytm omówiony jest „łopatologicznie”, z użyciem analogii, przykładów z życia codziennego i prostego języka.
- **Wizualizacje**: szczególnie przy sortowaniu (np. bąbelkowe, szybkie) dostępne są animacje i schematy blokowe wyjaśniające działanie.
- **Pseudokod i kod źródłowy**: algorytmy opisane są w pseudokodzie oraz zaimplementowane w różnych językach, takich jak:
    - **C++ i Python** – podstawowe wersje, z komentarzami.
    - **Kotlin, Haskell** – przykłady w językach funkcyjnych i alternatywnych.

Materiały te mogą stanowić zarówno punkt startowy do nauki, jak i narzędzie do powtórek czy inspirację dla projektów klasowych.

5. Podsumowanie

W artykule przedstawiłem zasoby dostępne na mojej platformie edukacyjnej (https://edu.cs-htiew.pl), skierowane zarówno do nauczycieli, jak i uczniów. Materiały zostały zaprojektowane z myślą o praktycznym wykorzystaniu w klasie i poza nią, łącząc głęboką wiedzę informatyczną z efektywnymi metodami jej przekazywania.

Dla **nauczycieli** przygotowane są gotowe scenariusze lekcji z algorytmiki i programowania, wsparte prezentacjami, zadaniami i rozwiązaniami, które oszczędzają czas na przygotowaniach. Dodatkowo sekcja poświęcona dydaktyce informatyki oferuje inspiracje do rozwijania własnych kompetencji metodycznych.

Dla **uczniów** dostępne są projekty programistyczne (tworzenie gier w Pythonie z użyciem Pygame Zero) oraz zadania maturalne z arkusza kalkulacyjnego, baz danych i algorytmiki, wspomagane filmami i szczegółowymi rozwiązaniami.

Uniwersalne materiały, takie jak algorytmy przedstawione w przystępny sposób z wizualizacjami i implementacjami w różnych językach (C++, Python), uzupełniają zasoby, odpowiadając na potrzeby zarówno początkujących, jak i zaawansowanych odbiorców.

Wszystkie materiały są bezpłatne, dostosowane do różnych kontekstów edukacyjnych (stacjonarnych, zdalnych, hybrydowych) i dostępne w języku polskim, wypełniając lukę w dostępie do wysokiej jakości zasobów informatycznych w naszym kraju.

Literatura

1. Mirosława Firszt, Marlena Kamińska, Martyna Kaszlewicz, Damian Kurpiewski, Kamil Palusiński, Krzysztof Skowronek, Propozycja Wykorzystania Systemów Blockly w Nauczaniu Informatyki, IWE 2015
2. Damian Kurpiewski, Krzysztof Skowronek, Mirosława Firszt, Trudne Tematy w Najprostszy Sposób, IWE 2018: 244-247
3. Damian Kurpiewski, Krzysztof Skowronek, Mirosława Firszt, Trudne Tematy w Najprostszy Sposób: Rekurencja, IWE 2018: 248-252
4. Damian Kurpiewski, Krzysztof Skowronek, Trudne Tematy w Najprostszy Sposób: Wprowadzenie do Zmiennych, IWE 2019: 289-297