Praca Dyplomowa Inżynierska

Konrad Maleńczak
205844

Projekt oraz realizacja aplikacji wspomagającej naukę arytmetyki modularnej

Project and implementation of an application supporting the learning of modular arithmetic

Praca dyplomowa na kierunku:
Informatyka

Praca wykonana pod kierunkiem
dr. Andrzeja Zembrzuskiego
Instytut Informatyki Technicznej
Katedra Systemów Informatycznych

Warszawa, rok 2024

SZKOŁA GŁÓWNA GOSPODARSTWA WIEJSKIEGO

Wydział Zastosowań Informatyki i Matematyki

Oświadczenie Promotora pracy

Oświadczam, że niniejsza praca została przygotowana pod moim kierunkiem i stwierdzam, że spełnia ona warunki do przedstawienia tej pracy w postępowaniu o nadanie tytułu zawodowego.

Data ..... Podpis promotora .....

Oświadczenie autora pracy

Świadom/a odpowiedzialności prawnej, w tym odpowiedzialności karnej za złożenie fałszywego oświadczenia, oświadczam, że niniejsza praca dyplomowa została napisana przeze mnie samodzielnie i nie zawiera treści uzyskanych w sposób niezgodny z obowiązującymi przepisami prawa, w szczególności z ustawą z dnia 4 lutego 1994 r. o prawie autorskim i prawach pokrewnych (Dz. U. 2019 poz. 1231 z późn. zm.)

Oświadczam, że przedstawiona praca nie była wcześniej podstawą żadnej procedury związanej z nadaniem dyplomu lub uzyskaniem tytułu zawodowego.

Oświadczam, że niniejsza wersja pracy jest identyczna z załączoną wersją elektroniczną. Przyjmuję do wiadomości, że praca dyplomowa poddana zostanie procedurze antyplagiatowej.

Data ..... Podpis autora pracy .....

Streszczenie

Projekt oraz realizacja aplikacji wspomagającej naukę arytmetyki modularnej

Temat tej pracy dyplomowej dotyczył zaprojektowania i implementacji aplikacji webowej wykorzystującej HTML, CSS i JavaScript, której celem było ułatwienie nauki arytmetyki modularnej dla studentów pierwszego roku informatyki na Wydziale Zastosowań Informatyki i Matematyki Szkoły Głównej Gospodarstwa Wiejskiego w Warszawie. Opisana została teoria matematyczna, implementacja wybranych algorytmów oraz wnioski.

Słowa kluczowe – arytmetyka modularna, praca dyplomowa, implementacja, SGGW, Szkoła Główna Gospodarstwa Wiejskiego, HTML, CSS, JavaScript

Summary

Project and implementation of an application supporting the learning of modular arithmetic

The subject of this thesis involved the design and implementation of a web application using HTML, CSS, and JavaScript aimed at facilitating the learning of modular arithmetic for first-year computer science students at the Faculty of Applied Informatics and Mathematics of the Warsaw University of Life Sciences. The mathematical theory, implementation of selected algorithms, and conclusions were described.

Keywords – modular arithmetic, thesis, implementation, SGGW, Warsaw University of Life Sciences, HTML, CSS, JavaScript

Spis treści

1 Wstęp

1.1 Cel i zakres pracy

Celem tej aplikacji jest oferowanie wsparcia edukacyjnego studentom pierwszego roku informatyki w zakresie arytmetyki modularnej. Została ona zaprojektowana przy użyciu czystych technologii HTML [1], CSS [2] i JavaScript [3], co zapewnia jej kompatybilność zarówno z urządzeniami mobilnymi, jak i stacjonarnymi. Aplikacja charakteryzuje się intuicyjnym i estetycznie przyjemnym interfejsem użytkownika.

Skonstruowana z myślą o edukacji, aplikacja obejmuje osiem różnorodnych tematów, każdy z nich koresponduje z innym zagadnieniem omawianym na zajęciach z Matematyki Dyskretnej 2 dla kierunku informatyka Wydziału Zastosowań Informatyki i Matematyki Szkoły Głównej Gospodarstwa Wiejskiego w Warszawie [4]. Każdy temat rozpoczyna się od szczegółowego wyjaśnienia teoretycznych podstaw, ilustrowanych konkretnym przykładem. Po zapoznaniu się z teorią, użytkownicy mają możliwość przetestowania swojej wiedzy poprzez rozwiązywanie zadań na poziomie podstawowym lub zaawansowanym. Wyniki wprowadzane przez użytkownika są następnie weryfikowane przez aplikację, co pozwala na natychmiastowe otrzymanie informacji zwrotnej o ich poprawności.

1.2 Tematyka pracy

Praca ta skupia się na zagadnieniach arytmetyki modularnej, oferując szczegółowy wgląd w rozumienie i implementację matematycznych problemów związanych z tą dziedziną.

W drugim rozdziale prezentowane są fundamentalne aspekty teorii matematycznej, obejmujące działania modulo, kongruencje, algorytm Euklidesa, równania modularne, funkcję Eulera, Małe Twierdzenie Fermata, oraz Twierdzenie Eulera. Ten segment pracy stanowi solidne podstawy teoretyczne, niezbędne do dalszego zrozumienia przedstawionych koncepcji [5-16].

W trzecim rozdziale szczegółowo omówiona została struktura aplikacji, prezentuje jej podział na poszczególne pliki i ukazuje ich wzajemne powiązania oraz określając, które pliki odpowiadają za konkretne funkcjonalności. Ponadto, zawarto instrukcje dotyczące sposobu uruchomienia aplikacji.

Czwarty rozdział skupia się na praktycznym zastosowaniu teorii w funkcjonowaniu aplikacji. Omówione są tu m.in. zasady działania arytmetyki modularnej, co stanowi klucz do pełnego zrozumienia mechanizmów leżących u podstaw aplikacji prezentowanej w tej pracy.

Piąty rozdział jest poświęcony prezentacji interfejsu użytkownika aplikacji, wraz z dogłębnym opisem jego funkcjonalności. Ta część pracy dostarcza czytelnikowi wizualnego oraz funkcjonalnego zrozumienia aplikacji.

W szóstym rozdziale, stanowiącym podsumowanie, dokonano refleksji na temat całokształtu pracy. Przedstawiono wnioski, jak również rozważania dotyczące możliwości dalszego rozwoju aplikacji, otwierając perspektywy na przyszłe badania i ulepszenia.

2 Teoria

Praca ma na celu stworzenie aplikacji, która wspomaga naukę arytmetyki modularnej. Niezbędna będzie do tego teoria matematyczna, która jest przedstawiona poniżej.

2.1 Operacja modulo

Modulo jest operacją matematyczną polegająca na wyznaczaniu reszty z dzielenia jednej liczby przez drugą. Przyjmijmy, że zapis $a \mod d = r$ oznacza, że r jest resztą z dzielenia a przez d [4][6][7].

Dzielenie z resztą można zapisać matematycznie jako:

$$a = bq + r, \tag{2.1}$$

gdzie:

- a jest dzielną (reprezentuje liczbę, którą dzielimy),
- b jest dzielnikiem (jest to liczba, przez którą dzielimy),
- q jest ilorazem (wynik dzielenia),
- r jest resztą (jest to różnica pozostała po dzieleniu [7].

Przykład 2.1.1 Rozpatrzmy dzielenie liczby 17 przez 5:

$$17 = 5 \cdot 3 + 2$$

- 17 to dzielna,
- 5 to dzielnik,
- 3 to iloraz,
- 2 to reszta.

Wynika z tego, że dzielenie 17 przez 5 daje iloraz 3 i resztę 2.

2.2 Przystawanie modulo

W matematyce termin *kongruencje modulo*, inaczej *przystawanie modulo*, odnosi się do sytuacji, kiedy dwie liczby całkowite dają tę samą resztę w procesie dzielenia przez trzecią liczbę całkowitą [4][7][9].
Relację $\equiv$, czyli kongruencji, która łączy ze sobą liczby dające tę samą resztę z dzielenia przez n, zapisuje się następująco:

$$a \equiv b \pmod{m}, \tag{2.2}$$

gdzie:

- a i b są liczbami całkowitymi,
- m jest liczbą całkowitą określającą operację modulo,
- symbol $\equiv$ oznacza relację przystawania modulo (kongruencji) [8][9].

Przykład 2.2.1
Jako przykład weźmy liczby 7 i 21, które przystają modulo 7. Oznacza to, że obie te liczby dają tę samą resztę (0) przy dzieleniu przez 7. Możemy to zapisać w formie:

$$7 \equiv 21 \pmod{7}.$$

Wynika z tego, że liczby 7 i 21 są kongruentne w kontekście modulo 7.

Przykład 2.2.2
Rozważmy teraz liczby 18 i 25, które są przystające modulo 7. Chociaż żadna z tych liczb nie daje reszty równej 0 przy dzieleniu przez 7, to obie dają tę samą resztę 4. Możemy to zapisać jako:

$$18 \equiv 25 \pmod{7}.$$

Oznacza to, że liczby 18 i 25 są kongruentne w kontekście modulo 7, ponieważ obie pozostawiają resztę 4 po podzieleniu przez 7.

2.3 Działania w zbiorze reszt modulo m

Działania w obrębie zbioru reszt modulo m odnoszą się do operacji matematycznych z użyciem liczb całkowitych, które skutkują resztą z dzielenia tych liczb przez ustaloną liczbę całkowitą m. W terminologii matematycznej, ten zbiór jest często reprezentowany jako $\mathbb{Z}_m$ [4][9].

- Dodawanie w zbiorze $\mathbb{Z}_m$: rezultat sumowania dwóch liczb całkowitych a oraz b w kontekście modulo m przedstawia się jako:

$$(a+b) \pmod{m}. \tag{2.3}$$

Przykład 2.3.1 $(10+3) \mod 4 = 13 \mod 4 = 1$ [10].

- Odejmowanie w zbiorze $\mathbb{Z}_m$: rezultat odejmowania dwóch liczb całkowitych a oraz b w kontekście modulo m reprezentuje się jako:

$$(a-b) \pmod{m}. \tag{2.4}$$

Przykład 2.3.2 $(10-3) \mod 7 = 7 \mod 7 = 0$.

- Mnożenie w zbiorze $\mathbb{Z}_m$: rezultat mnożenia dwóch liczb całkowitych a i b w kontekście modulo m określa się jako:

$$(a \cdot b) \pmod{m}. \tag{2.5}$$

Przykład 2.3.3 $(6 \cdot 4) \mod 5 = 24 \mod 5 = 4$.

- Dzielenie całkowite w zbiorze $\mathbb{Z}_m$: wynik dzielenia całkowitego liczby a przez liczbę b w kontekście modulo m można przedstawić jako:

$$\left(\left\lfloor \frac{a}{b} \right\rfloor\right) \pmod{m}. \tag{2.6}$$

Przykład 2.3.4 $\left\lfloor \frac{24}{4} \right\rfloor \mod 5 = 6 \mod 5 = 1$.

2.4 Rozszerzony algorytm Euklidesa

Największy wspólny dzielnik dwóch liczb całkowitych a i b, zapisywany jako NWD(a, b), jest największą liczbą całkowitą dzielącą bez reszty obie te liczby.

Tradycyjny algorytm Euklidesa [4][11][12] jest sprawdzoną metodą ustalania największego wspólnego dzielnika (NWD) między dwoma liczbami całkowitymi. Proces ten polega na powtarzanym obliczaniu reszty z dzielenia jednej liczby przez drugą (operacja modulo), aż jedna z nich osiągnie wartość zero. W tym momencie, druga liczba reprezentuje NWD. Jest to jedna z najstarszych metod matematycznych wykorzystywanych w teorii liczb i charakteryzuje się wysoką efektywnością.

Rozszerzony algorytm Euklidesa [4][11][12] jest metodą pozwalającą na obliczenie wartości zmiennych x i y w równaniu:

$$a \cdot x + b \cdot y = NWD(a,b). \tag{2.7}$$

Rozszerzony algorytm Euklidesa jest użyteczny w różnych dziedzinach matematyki i informatyki. W kryptografii algorytm ten znajduje jedno ze swoich najważniejszych zastosowań, szczególnie w algorytmie RSA [5], który jest jednym z najbardziej znaczących algorytmów kryptografii asymetrycznej. Ten algorytm jest kluczowym narzędziem do obliczania kluczy prywatnych i publicznych, co jest fundamentem bezpiecznej komunikacji cyfrowej.

Rozszerzony algorytm Euklidesa jest ulepszoną formą klasycznego algorytmu, skoncentrowaną na wyznaczaniu współczynników x i y dla równania (2.7).

Proces wykonywania rozszerzonego algorytmu Euklidesa obejmuje kilka kluczowych etapów:

- Rozpoczynamy od liczb a i b. Należy przypisać wartość 1 do dwóch pomocniczych elementów, a dla kolejnych dwóch ustalić wartość 0 i 1.
- Obliczyć iloraz i resztę z dzielenia a przez b, wykonując dzielenie z resztą.
- Zaktualizować pomocnicze elementy: pierwszy przyjmuje wartość drugiego, drugi przyjmuje wartość reszty, a następne dwa są aktualizowane w oparciu o obliczony wcześniej iloraz.
- Algorytm należy powtórzyć tak długo, aż nowa wartość reszty stanie się równa 0 [12].

Przykład 2.4.1 Analizujemy działanie rozszerzonego algorytmu Euklidesa dla $n = 1547$ i $m = 560$ [4]. Najpierw należy wykonać tradycyjny algorytm Euklidesa:

$$\begin{aligned}
1547 &= 2 \cdot 560 + 427 \\
560 &= 1 \cdot 427 + 133 \\
427 &= 3 \cdot 133 + 28 \\
133 &= 4 \cdot 28 + 21 \\
28 &= 1 \cdot 21 + 7 \\
21 &= 3 \cdot 7 + 0.
\end{aligned}$$

Z tego wynika, że NWD(1547, 560) = 7.

Aby wyznaczyć współczynniki x i y w równaniu (2.7), obliczamy:

$$\begin{aligned}
7 &= 28 - 1 \cdot 21 = 28 - 1 \cdot (133 - 4 \cdot 28) \\
&= -1 \cdot 133 + 5 \cdot 28 = -1 \cdot 133 + 5 \cdot (427 - 3 \cdot 133) \\
&= 5 \cdot 427 - 16 \cdot 133 = 5 \cdot 427 - 16 \cdot (560 - 1 \cdot 427) \\
&= -16 \cdot 560 + 21 \cdot 427 = -16 \cdot 560 + 21 \cdot (1547 - 2 \cdot 560) \\
&= 21 \cdot 1547 - 58 \cdot 560.
\end{aligned}$$

Z tych obliczeń, z wykorzystaniem rozszerzonego algorytmu Euklidesa, wynika że:

$$NWD(1547, 560) = 7 = 21 \cdot 1547 - 58 \cdot 560$$

2.5 Równania modularne

Istnieje wiele rodzajów kongruencji, ale jedna z najbardziej znaczących to kongruencja liniowa, opisana równaniem [13]:

$$A \cdot x \equiv B \pmod{m}, \tag{2.8}$$

gdzie:

- A jest współczynnikiem przy zmiennej x,
- B jest stałą liczbą,
- m określa liczbę stosowaną w operacji modulo.

Dla $A, B \in \mathbb{Z}, A \neq 0$, metoda rozwiązania równania modularnego (2.8) z pojedynczą niewiadomą x przedstawiona jest poniżej [14].

- Przekształcamy dany wzór do postaci:

$$ax \equiv b \pmod{m}, \tag{2.9}$$

  gdzie $a \equiv A \pmod{m}$ i $b \equiv B \pmod{m}$ przy założeniu, że $0 \leq a, b < m$.
- Następnie obliczamy największy wspólny dzielnik a i m, oznaczony jako $d = NWD(a, m)$.
- Sprawdzamy, czy d nie dzieli b, wówczas kongruencja nie ma rozwiązania, w przeciwnym razie istnieje nieskończenie wiele rozwiązań, które wymagają dalszych obliczeń.
- Przeprowadzamy dzielenie przez d:

$$\frac{a}{d} \cdot x \equiv \frac{b}{d} \pmod{\frac{m}{d}}. \tag{2.10}$$

- Otrzymujemy rozwiązanie:

$$x \equiv \left(\frac{a}{d}\right)^{-1} \cdot \frac{b}{d} \pmod{\frac{m}{d}}. \tag{2.11}$$

- Ustalamy odwrotność $\left(\frac{a}{d}\right)^{-1}$ poprzez metodę prób i błędów lub korzystając z rozszerzonego algorytmu Euklidesa:

$$1 = NWD\left(\frac{a}{d}, \frac{m}{d}\right) = y \cdot \frac{a}{d} + z \cdot \frac{m}{d}, \tag{2.12}$$

$$1 \equiv y \cdot \frac{a}{d} \pmod{\frac{m}{d}}, \tag{2.13}$$

$$\left(\frac{a}{d}\right)^{-1} \equiv y \pmod{\frac{m}{d}}. \tag{2.14}$$

- Jedno z rozwiązań ma postać:

$$x_0 = \left(\frac{a}{d}\right)^{-1} \cdot \frac{b}{d}. \tag{2.15}$$

- Wszystkie rozwiązania można zapisać jako:

$$x = x_0 + k \cdot \frac{m}{d}, \tag{2.16}$$

gdzie k jest dowolną liczbą całkowitą.

2.6 Funkcja Eulera

Funkcja Eulera φ [4][15] przyporządkowuje każdej liczbie naturalnej liczbę liczb względnie pierwszych z nią, ale mniejszych lub równych jej samej.
Innymi słowy, $\varphi(n)$ to liczba liczb mniejszych od n, które mają największy wspólny dzielnik z n równy 1. Na przykład $\varphi(6) = 2$ (ponieważ tylko 1 i 5 są względnie pierwsze z 6), lub $\varphi(15) = 8$ (ponieważ liczby 1, 2, 4, 7, 8, 11, 13 i 14 są względnie pierwsze z 15). Funkcja Eulera, reprezentowana jako $\varphi(n)$, odgrywa istotną funkcję w dziedzinie teorii liczb, będąc nie tylko podstawą dla Małego Twierdzenia Fermata i Twierdzenia Eulera, ale także mając zastosowanie w różnorodnych algorytmach kryptograficznych, takich jak RSA [5]. W innych słowach, $\varphi(n)$ określa liczba liczb względnie pierwszych z n w zakresie od 1 do n. Jest to interesujące zwłaszcza w przypadku liczb pierwszych, ponieważ dla każdej liczby pierwszej p, wartość funkcji Eulera $\varphi(p)$ równa się $p-1$, co jest wynikiem faktu, że wszystkie liczby mniejsze od liczby pierwszej są z nią względnie pierwsze. Na przykład liczba 13 jako liczba pierwsza jest względnie pierwsza do dwunastu liczb od 1 do 12, więc $\varphi(13) = 12$.
Następująca tabela wartości przedstawia wartości funkcji dla pierwszych piętnastu liczb naturalnych.

n	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$\varphi(n)$	1	1	2	2	4	2	6	4	6	4	10	4	12	6	8

Tabela 2.1. Tabela prezentująca pierwsze 15 wartości funkcji Eulera [15].

Obliczenie funkcji Eulera $\varphi(n)$ wiąże się z rozkładem liczby n na czynniki pierwsze. Dokładniej, jeśli liczba n jest przedstawiona jako iloczyn potęg liczb pierwszych, tj.

$$n = \prod_{p|n} p^{a_p}, \tag{2.17}$$

gdzie p jest liczbą pierwszą, a a_p jest odpowiadającym mu wykładnikiem potęgi, wówczas wartość $\varphi(n)$ wynika z następującego wzoru:

$$\varphi(n) = \prod_{p|n} p^{a_p-1}(p-1) = n\prod_{p|n}\left(1-\frac{1}{p}\right). \tag{2.18}$$

Oznacza to, że $\varphi(n)$ jest iloczynem liczby n i czynników postaci $\left(1-\frac{1}{p}\right)$ dla każdej liczby pierwszej p, która jest dzielnikiem n.

Przykład 2.6.1

Dla $n = 120$, czynniki pierwsze to $2^3 \cdot 3^1 \cdot 5^1$. Obliczając $\varphi(120)$ korzystając z wzoru, otrzymujemy:

$$\varphi(120) = 120 \cdot \left(1 - \frac{1}{2}\right) \cdot \left(1 - \frac{1}{3}\right) \cdot \left(1 - \frac{1}{5}\right) = 120 \cdot \frac{1}{2} \cdot \frac{2}{3} \cdot \frac{4}{5} = 32.$$

Oznacza to, że $\varphi(120) = 32$, co oznacza, iż 32 liczby całkowite w zakresie od 1 do 120 są względnie pierwsze z 120.

2.7 Małe Twierdzenie Fermata

Małe Twierdzenie Fermata [4][16] stanowi znaczący element w teorii liczb, analizując właściwości liczb całkowitych w ramach arytmetyki modularnej, szczególnie w kontekście potęgowania. Twierdzenie to ma szerokie zastosowanie w kryptografii, teorii kodowania oraz w innych dziedzinach matematyki.

Sformułowanie Małego Twierdzenia Fermata.

Dla całkowitej liczby a i liczby pierwszej p, obowiązuje następująca ogólna kongruencja:

$$a^p \equiv a \pmod{p}. \tag{2.19}$$

Jeśli liczby a i p są względnie pierwsze, to istnieje odwrotność modulo. Innymi słowy,

$$a^{p-1} \equiv 1 \pmod{p}. \tag{2.20}$$

Ogromne znaczenie tego twierdzenia odnajduje się w kryptografii, gdzie liczby pierwsze są często kluczowym elementem w generowaniu kluczy szyfrowych. Małe Twierdzenie Fermata umożliwia kontrolę zgodności z kluczami i odgrywa kluczową rolę w projektowaniu bezpiecznych systemów szyfrujących.

Przykład 2.7.1
Obliczmy $5^{75} \mod 13$.
Ponieważ 13 jest liczbą pierwszą, a 5 liczbą całkowitą, możemy zastosować Małe Twierdzenie Fermata:

$$5^{12} \equiv 1 \pmod{13},$$

$$75 \mod 12 = 3,$$

$$5^{75} = (5^{12})^6 \cdot 5^3 \equiv 1 \pmod{13}.$$

Następnie obliczamy 5^3 modulo 13:

$$5^3 \equiv 8 \pmod{13},$$

$$5^{75} \equiv (5^{12})^6 \cdot 5^3 \equiv 1^6 \cdot 8 \equiv 8 \pmod{13}.$$

W konsekwencji, $5^{75} \mod 13$ równa się 8.

2.8 Twierdzenie Eulera

Twierdzenie Eulera [4][17], znane również jako Twierdzenie Eulera-Fermata, nazwane na cześć Leonharda Eulera [1707-1783] i Pierre'a de Fermata [1601-1665], stanowi uogólnienie małego twierdzenia Fermata. Twierdzenie to ma zastosowanie w systemach szyfrujących, jak RSA [5].

Sformułowanie Twierdzenia Eulera.

Dla dowolnych liczb całkowitych m i n, które są wzajemnie względnie pierwsze,
(czyli NWD(m, n) jest równy 1), zachodzi związek:

$$n^{\varphi(m)} \equiv 1 \pmod{m}. \tag{2.21}$$

Warto dodać, że Twierdzenie Eulera ma znaczenie w teorii liczb oraz kryptografii, szczególnie w algorytmach bazujących na teorii liczb, takich jak RSA [5]. W kontekście RSA, gdzie bezpieczeństwo opiera się na trudności rozkładu na czynniki pierwsze dużych liczb pierwszych, właśnie funkcja Eulera jest wykorzystywana do generowania kluczy.
Ponieważ dla liczb pierwszych zachodzi $\varphi(m) = m - 1$, w tych przypadkach Twierdzenie Eulera przechodzi w Małe Twierdzenie Fermata.

Przykład 2.8.1

Obliczmy $7^{85} \mod 15$.
Ponieważ 7 i 15 są względnie pierwsze, możemy zastosować Twierdzenie Eulera.

$$\varphi(15) = \varphi(3) \cdot \varphi(5) = 2 \cdot 4 = 8,$$

$$7^{85} = 7^{8 \cdot 10 + 5} \equiv 7^5 \pmod{15}.$$

Następnie obliczamy wybrane potęgi 7 modulo 15:

$$7^1 \equiv 7 \pmod{15},$$

$$7^4 \equiv 4^2 = 16 \equiv 1 \pmod{15},$$

$$7^{85} \equiv 7^5 = 7^4 \cdot 7 \equiv 1 \cdot 7 = 7 \pmod{15}.$$

Tak więc, wynik $7^{85} \mod 15$ to 7.

3 Struktura aplikacji

W celu pełniejszego zrozumienia funkcjonalności aplikacji, kluczowe jest omówienie jej struktury. Aplikacja jest zorganizowana w sposób zapewniający łatwość nawigacji oraz intuicyjność użytkowania.

Aplikacja zawiera łącznie czterdzieści trzy pliki, zorganizowane w precyzyjnie określonym porządku. Głównym plikiem jest `index.html`, który inicjuje działanie aplikacji. Po jego uruchomieniu użytkownikowi prezentowana jest strona główna, zawierająca wprowadzenie do arytmetyki modularnej, pasek nawigacyjny po lewej stronie oraz informacje kontaktowe twórcy aplikacji na dole strony.

Każdy z ośmiu tematów aplikacji składa się z pięciu dedykowanych plików, a mianowicie trzech plików HTML [1] oraz dwóch plików JavaScript [3]. Pierwszy z plików HTML przedstawia teorię dotyczącą danego tematu. Dwa kolejne to strony z zadaniami - podstawowymi i zaawansowanymi. Funkcjonalności tych stron są wspierane przez przypisane im pliki JavaScript, które odpowiadają za losowanie zadań oraz weryfikację poprawności odpowiedzi użytkownika.

Pozostałe dwa pliki to `kontakt.html` zawierający informacje kontaktowe dostępne na każdej podstronie oraz `styles.css` [2], odpowiadający za stylizację i układ elementów na stronie.

Z poziomu strony głównej użytkownik może przejść do wybranego tematu, aby zgłębić jego treść teoretyczną lub rozwiązać zadania podzielone na poziomy trudności. Strony teoretyczne bezpośrednio łączą się ze stronami z zadaniami, co zapewnia płynność i przejrzystość nawigacji. Istnieje również opcja powrotu do strony głównej, aby wybrać inny temat do nauki.

4 Teoria w praktyce

Po zaprezentowaniu teoretycznych podstaw można przejść do opisu działania algorytmów oraz ich wdrożenia w ramach aplikacji. Każdy temat w aplikacji obejmuje zestaw czterech zadań, sklasyfikowanych jako *podstawowe* i *zaawansowane*. Zaleca się rozpoczynanie nauki od zadań podstawowych, ponieważ charakteryzują się one niższym poziomem trudności i stanowią doskonały punkt wyjścia. Po opanowaniu materiału podstawowego, użytkownik może przystąpić do rozwiązywania zadań zaawansowanych. Te ostatnie różnią się od poprzednich bardziej skomplikowanymi obliczeniami matematycznymi, stanowiąc wyzwanie dla bardziej zaawansowanych użytkowników.

4.1 Implementacja algorytmów w JavaScript

Algorytmy stosowane w aplikacji zostały zaimplementowane przy użyciu języka JavaScript, idealnie nadającego się do tworzenia dynamicznych aplikacji webowych [18]. Każde z zadań obsługiwane jest przez specjalnie przygotowaną funkcję, która dostarcza rozwiązanie. W przypadku wprowadzenia przez użytkownika poprawnej odpowiedzi, na ekranie przez trzy sekundy pojawia się zielony komunikat “Dobrze!”, zaś przy błędnej odpowiedzi wyświetlany jest czerwony komunikat “Źle!”. Kod JavaScript jest zintegrowany z HTML, co gwarantuje poprawne wyświetlanie treści w różnych przeglądarkach internetowych. Poniższe algorytmy pochodzą z własnych źródeł i są zaimplementowane w aplikacji.

4.1.1 Inicjowanie zadań

Struktura kodu JavaScript skupia się na efektywnej obsłudze formularzy w aplikacji. Zadania inicjowane są przez funkcję `initializeTask`, która wykonuje następujące czynności:

- wczytywanie elementów formularza z interfejsu użytkownika,
- generowanie losowych danych wejściowych i ich prezentacja,
- ustawianie obsługi zdarzeń dla przesyłania formularza.

4.1.2 Zarządzanie zdarzeniem przesłania formularza

Kolejna kluczowa funkcja, `handleTaskSubmit`, jest odpowiedzialna za:

- odbiór i analizę odpowiedzi użytkownika,
- ocenę poprawności udzielonych odpowiedzi,
- informowanie użytkownika o rezultacie jego odpowiedzi.

4.1.3 Organizacja kodu

Kod programu skoncentrowany jest wokół dwóch podstawowych funkcji: `initializeTask` i `handleTaskSubmit`. Funkcja `initializeTask` aktywuje się automatycznie dla każdego z czterech zadań, gdy strona jest ładowana.

4.1.4 Proces weryfikacji odpowiedzi

Kod porównuje odpowiedzi użytkownika z oczekiwanymi wynikami. W zależności od poprawności odpowiedzi, wyświetlany jest odpowiedni komunikat: zielony w przypadku poprawnej odpowiedzi, a czerwony przy błędzie. Komunikat znika automatycznie po upływie trzech sekund.

4.2 Przykłady implementacji algorytmów

Aby skutecznie rozwiązywać zadania w aplikacji, kluczowe jest zaimplementowanie odpowiednich algorytmów. Algorytmy te stanowią fundament działania aplikacji, umożliwiając użytkownikowi interaktywne i efektywne rozwiązywanie problemów matematycznych oraz innych zadań logicznych. Poniżej przedstawiono wybrane algorytmy, które odegrały znaczącą rolę w procesie tworzenia aplikacji.

4.2.1 Algorytm generujący losowe liczby

Podstawowym algorytmem w aplikacji jest generator losowych liczb. Funkcja ta znacząco przyczynia się do dywersyfikacji zadań, podnosząc zarówno ich walor edukacyjny, jak i atrakcyjność. Algorytm ten wybiera liczby losowo z określonego zakresu, zapewniając zróżnicowanie w problemach matematycznych (Skrypt 4.1).

```
1  function randomInt(min, max) {
2    return Math.floor(Math.random() * (max - min + 1) + min);
3  }
```

Skrypt 4.1. Funkcja generująca losowe liczby w JavaScript

Funkcja przyjmuje dwa parametry:

- `min` - minimalna wartość zakresu (włącznie),
- `max` - maksymalna wartość zakresu (włącznie).

Wynikiem jest losowa liczba całkowita z przedziału od `min` do `max`, generowana według poniższych kroków:

- wywołanie funkcji `Math.random()` generuje liczbę zmiennoprzecinkową pomiędzy 0 a 1 (bez 1),
- wynik mnożony jest przez `(max - min + 1)`, co pozwala na dostosowanie zakresu do żądanych rozmiarów,
- dodanie wartości `min` przesuwa zakres, gwarantując, że jego dolny limit wynosi `min`,
- na koniec, funkcja `Math.floor()` zaokrągla wynik w dół do najbliższej mniejszej liczby całkowitej, co zapewnia, że wynik pozostaje w ustalonym zakresie.

4.2.2 Algorytm generujący losowe liczby pierwsze

Funkcja `randomPrime` w JavaScript jest zaprojektowana do generowania losowej liczby pierwszej w określonym przedziale. Liczby pierwsze odgrywają kluczową rolę w wielu obszarach matematyki, włączając w to kryptografię i algorytmikę (Skrypt 4.2).

```
function randomPrime(min, max) {
  function isPrime(num) {
    if (num <= 1) return false;
    if (num <= 3) return true;
    if (num % 2 === 0 || num % 3 === 0) return false;
    let i = 5;
    while (i * i <= num) {
      if (num % i === 0 || num % (i + 2) === 0) return false
          ;
      i += 6;
    }
    return true;
  }
  let primeNumber = randomInt(min, max);
  while (!isPrime(primeNumber)) {
    primeNumber = randomInt(min, max);
  }
  return primeNumber;
}
```

Skrypt 4.2. Funkcja w JavaScript do generowania liczb pierwszych

Procedura działania funkcji `randomPrime` jest następująca:

- wewnętrzna funkcja `isPrime` ocenia, czy dana liczba `num` jest pierwsza. Sprawdza ona kryteria liczby pierwszej, czyli wyłączną podzielność przez jedynkę i samą siebie, a także, czy liczba jest większa od jedynki,
- centralna część funkcji `randomPrime` generuje liczbę w zakresie od `min` do `max`, a następnie wykorzystuje `isPrime` do weryfikacji jej pierwszości,
- w przypadku, gdy wylosowana liczba nie jest pierwsza, proces losowania jest powtarzany do momentu uzyskania liczby pierwszej,
- ostatecznie funkcja zwraca wylosowaną liczbę pierwszą z zadanego przedziału.

4.2.3 Algorytm Euklidesa do obliczania NWD

Obliczanie największego wspólnego dzielnika (NWD) dwóch liczb jest kluczowe w teorii liczb i ma zastosowanie w różnych dziedzinach, takich jak kryptografia, analiza numeryczna i algorytmika. NWD to największy wspólny dzielnik, który dzieli obie liczby bez reszty (Skrypt 4.3).

```
1    function gcd(a, b) {
2      if (b === 0) return a;
3      return gcd(b, a % b);
4    }
```

Skrypt 4.3. Funkcja do obliczania NWD w JavaScript

Proces obliczania NWD za pomocą algorytmu Euklidesa obejmuje poniższe kroki:

- algorytm rozpoczyna od sprawdzenia, czy drugi argument (`b`) jest równy 0, co stanowi warunek bazowy rekurencji, gdy `b` jest równy 0, zwracana jest wartość `a` jako NWD, zgodnie z definicją NWD, w przeciwnym wypadku, algorytm przechodzi do kolejnego etapu,
- w przypadku, gdy `b` jest różne od zera, algorytm kontynuuje poprzez rekurencyjne wywołanie funkcji `gcd`,
- w kolejnym wywołaniu, algorytm zamienia miejscami wartości `a` i `b`, przy czym nowa wartość `b` to reszta z dzielenia `a` przez `b`,
- proces ten jest powtarzany aż do spełnienia warunku bazowego, czyli do momentu, gdy `b` stanie się równe 0. Wówczas aktualna wartość `a` reprezentuje NWD obu liczb.

4.2.4 Rozszerzony algorytm Euklidesa

Rozszerzony algorytm Euklidesa jest rozbudowaną wersją klasycznego algorytmu obliczania NWD dwóch liczb całkowitych. Jego unikalność polega na zdolności do identyfikacji współczynników całkowitoliczbowych x i y dla danych liczb a i b, które spełniają równanie $ax+by = \text{NWD}(a,b)$. Ta funkcjonalność czyni go nieocenionym narzędziem w rozwiązaniu równań diofantycznych i w kryptografii, szczególnie w kontekście obliczania odwrotności modularnych (Skrypt 4.4).

```
function extendedEuclideanAlgorithm(a, b) {
  let x0 = 1, y0 = 0, x1 = 0, y1 = 1;
  while (b !== 0) {
    const q = Math.floor(a / b);
    [x0, x1] = [x1, x0 - q * x1];
    [y0, y1] = [y1, y0 - q * y1];
    [a, b] = [b, a - q * b];
  }
  return [a, x0, y0];
}
```

Skrypt 4.4. Funkcja Rozszerzonego Algorytmu Euklidesa w JavaScript

Algorytm inicjuje zmienne $x0$, $y0$, $x1$, $y1$ do przechowywania współczynników. Następnie, w pętli `while` wykonuje następujące operacje:

- oblicza iloraz q z dzielenia a przez b,
- aktualizuje wartości $x0$, $x1$, $y0$, $y1$ na podstawie obliczonego ilorazu,
- zamienia wartości a i b dla kontynuacji algorytmu z nowymi parametrami.

Iteracje trwają, aż b osiągnie wartość zero. W rezultacie, algorytm zwraca NWD(a, b) oraz współczynniki $x0$ i $y0$, które spełniają równanie $ax+by = \text{NWD}(a,b)$.
W kontekście obliczeń, algorytm rozpoczyna od sprawdzenia, czy wartość `b` jest równa 0, co stanowi warunek bazowy. Gdy `b` nie jest równe 0, algorytm wykonuje rekurencyjne wywołanie, zamieniając argumenty i stosując operację modulo. Proces ten kontynuowany jest do momentu, gdy wartość `b` stanie się równa 0, co wskazuje, że aktualna wartość `a` jest NWD obu liczb.

4.2.5 Obliczanie odwrotności modularnej

Odwrotność modularna to kluczowe pojęcie w teorii liczb, mające znaczenie w rozwiązywaniu kongruencji. Odwrotność modularna liczby a w module m jest to taka liczba x, która spełnia równanie $ax \equiv 1 \pmod{m}$. Rozwiązywanie kongruencji obejmuje znalezienie takich wartości x, które spełniają równanie $Ax \equiv B \pmod{m}$. Funkcje `modularInverse` i `solveCongruence` w języku JavaScript wykonują te obliczenia.

Funkcja `modularInverse`

Odwrotność modularna liczby a w module m jest liczbą x, dla której zachodzi $ax \equiv 1 \pmod{m}$, co można zapisać jako:

$$ax \equiv 1 \pmod{m}. \tag{4.1}$$

To pojęcie odgrywa istotną rolę w wielu algorytmach matematycznych i kryptograficznych, które działają w ograniczonym systemie reszt. Kod realizujący tę funkcję jest przedstawiony w skrypcie 4.5.

```
1  function modularInverse(a, m) {
2    a %= m;
3    for (let x = 1; x < m; x++) {
4      if ((a * x) % m === 1) {
5        return x;
6      }
7    }
8    return -1;
9  }
```

Skrypt 4.5. Funkcja obliczająca odwrotność modularną liczby a w module m w JavaScript

Proces obliczania odwrotności modularnej w funkcji `modularInverse` obejmuje:

- dostosowanie wartości a do zakresu modułu m,
- iteracyjne szukanie takiej wartości x, która spełnia równanie $(a \cdot x) \mod m = 1$,
- zwrócenie wartości x, jeśli taka istnieje. W przeciwnym razie zwrócona zostaje wartość -1.

Funkcja `solveCongruence`

Znalezienie rozwiązań kongruencji liniowych jest istotnym aspektem w teorii liczb i metodach algebraicznych. Kongruencje liniowe mają postać $Ax \equiv B \pmod{m}$, gdzie A, B i m są zadanymi liczbami całkowitymi, a szukana liczba x jest niewiadomą. Proces rozwiązywania polega na odnalezieniu wszystkich możliwych wartości x, które spełniają równanie w danym systemie modularnym. Kod realizujący tę funkcję jest przedstawiony w skrypcie 4.6.

```
function solveCongruence(A, B, m) {
  A %= m;
  B %= m;
  const d = gcd(A, m);
  const inv = modularInverse(A / d, m / d);
  if (B % d !== 0 || inv === -1) {
    return -1;
  } else {
    const x = (B / d) * inv;
    return x % (m / d);
  }
}
```

Skrypt 4.6. Funkcja rozwiązująca kongruencję liniową w JavaScript

Funkcja `solveCongruence` jest wykorzystywana do rozwiązywania równań kongruencji postaci $Ax \equiv B \pmod{m}$. Jej działanie obejmuje:

- dopasowanie wartości A i B do zakresu modułu m,
- obliczenie NWD(A,m) oraz odwrotności modularnej,
- weryfikację istnienia rozwiązania poprzez sprawdzenie, czy B dzieli się przez d oraz czy odwrotność modularna istnieje (czy jej wynik nie jest równy -1),
- jeżeli rozwiązanie istnieje, obliczenie wartości x wykorzystując odwrotność modularną, w przeciwnym razie funkcja zwraca -1,
- zwrócenie x jako wyniku równania kongruencji.

4.2.6 Funkcja Eulera

Funkcja Eulera, znana również jako tocjent, odgrywa istotną rolę w teorii liczb. Dla każdej liczby całkowitej n, funkcja Eulera, oznaczana jako $\varphi(n)$, określa liczbę liczb całkowitych w zakresie od 1 do n, które są względnie pierwsze z n. Oznacza to, że zwraca liczbę liczb mniejszych od n, które nie mają żadnych wspólnych dzielników z n, poza liczbą 1.

```
function eulerFunction(n) {
  let result = n;
  for (let p = 2; p * p <= n; ++p) {
    if (n % p == 0) {
      while (n % p == 0) {
        n /= p;
      }
      result -= result / p;
    }
  }
  if (n > 1) {
    result -= result / n;
  }
  return result;
}
```

Skrypt 4.7. Funkcja obliczająca wartość funkcji Eulera w JavaScript

Algorytm funkcji `eulerFunction` przebiega następująco:

- początkowo zmienna `result` jest ustawiona na wartość zmiennej n,
- funkcja iteruje przez liczby całkowite p od 2 do pierwiastka kwadratowego z n, sprawdzając, czy p jest dzielnikiem n,
- jeśli p jest dzielnikiem n, to n jest dzielone przez p, dopóki nie przestanie być podzielne przez p. Następnie `result` jest aktualizowana przez odjęcie jej części proporcjonalnej do p,
- jeżeli po zakończeniu pętli n pozostaje większe niż 1, oznacza to, że ma ona dzielnik pierwszy większy od jej pierwiastka kwadratowego. W tym przypadku zmienna `result` jest modyfikowana,
- na zakończenie, funkcja zwraca wartość `result`, będącą wynikiem $\varphi(n)$.

4.2.7 Potęgowanie modularne

Potęgowanie modularne jest podstawowym elementem teorii liczb i kryptografii, polegającym na obliczeniu reszty z dzielenia potęgi przez ustalony moduł. Ta metoda jest niezbędna w systemach modularnych, gdzie wyniki operacji muszą pozostać w określonym zakresie (Skrypt 4.8).

```
1  function calculateModuloExponentiation(base, exponent,
       modulus) {
2    let result = 1;
3    for (let i = 0; i < exponent; i++) {
4      result = (result * base) % modulus;
5    }
6    return result;
7  }
```

Skrypt 4.8. Funkcja potęgowania modularnego w JavaScript

Działanie funkcji `calculateModuloExponentiation` obejmuje:

- inicjalizacja zmiennej `result` wartością 1,
- iteracyjne mnożenie `result` przez `base` (podstawę) i obliczanie reszty z dzielenia przez `modulus` po każdym mnożeniu,
- kontynuacja powyższego procesu aż do osiągnięcia liczby iteracji równej `exponent` (wykładnikowi),
- zwrócenie końcowej wartości `result`, będącej wynikiem potęgowania modularnego.

4.2.8 Funkcja sprawdzająca względną pierwszość

Względna pierwszość liczb jest podstawowym pojęciem w teorii liczb. Mówimy, że dwie liczby są względnie pierwsze, gdy ich jedynym wspólnym dzielnikiem jest liczba 1. Ta koncepcja jest szczególnie istotna w matematyce, w tym w teorii liczb i kryptografii (Skrypt 4.9).

```
1    function relativelyPrime(a, b) {
2      if (gcd(a, b) === 1) return true;
3      return false;
4    }
```

Skrypt 4.9. Funkcja do weryfikacji względnej pierwszości w JavaScript

Funkcja `relativelyPrime` przeprowadza test na względną pierwszość dwóch licz całkowitych poprzez następujące kroki:

- wywołanie funkcji `gcd` do obliczenia największego wspólnego dzielnika (NWD) liczb a i b,
- ocena, czy NWD tych liczb wynosi dokładnie 1, co jest kluczowe do stwierdzenia, że są one względnie pierwsze,
- zwrócenie wartości `true`, jeśli liczby są względnie pierwsze, lub `false` w przeciwnym razie.

5 Graficzny interfejs użytkownika

5.1 Opis ogólny

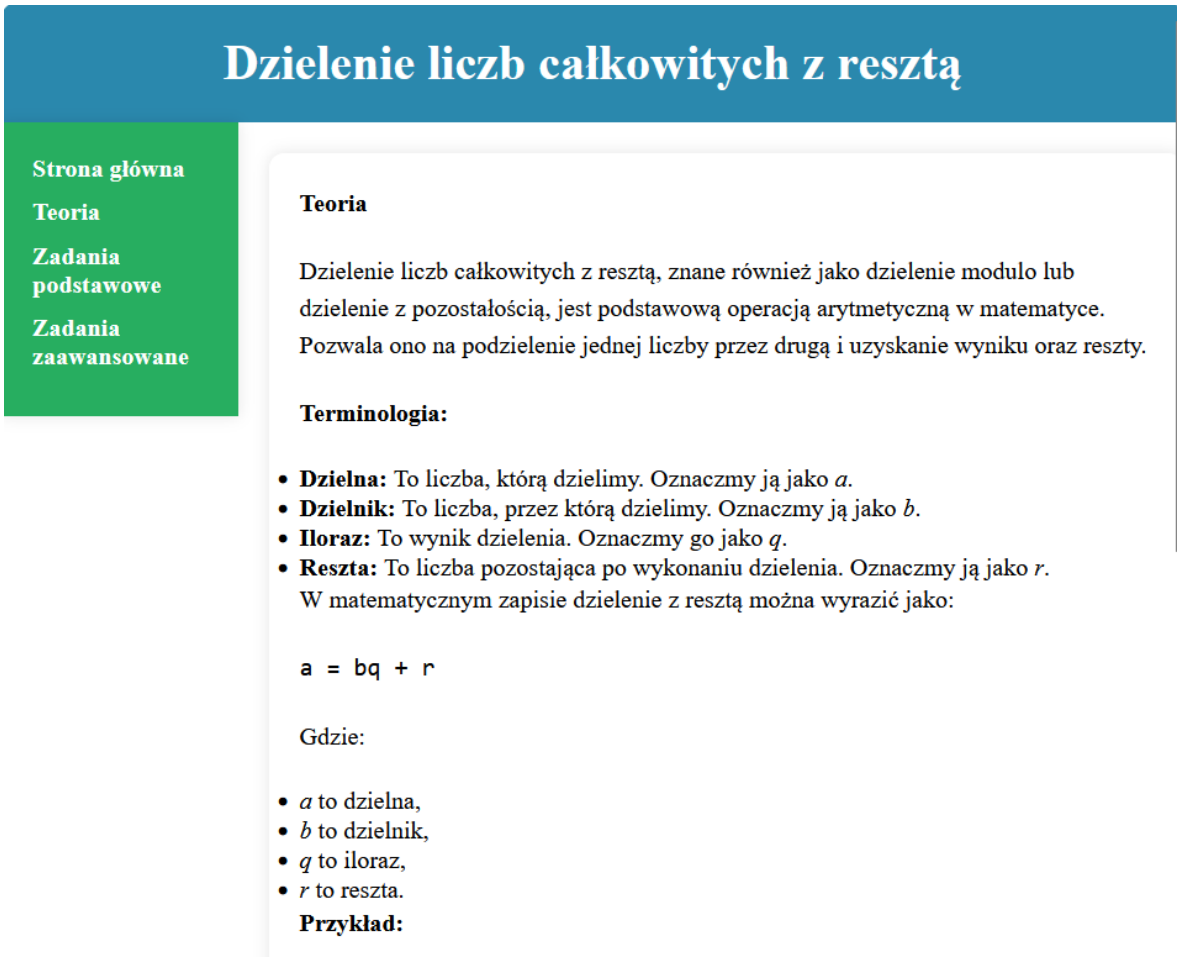
Aby zapewnić łatwy i uniwersalny dostęp, aplikacja została zaprojektowana do użytku w przeglądarkach internetowych, co umożliwia jej wykorzystanie zarówno na urządzeniach stacjonarnych, jak i mobilnych. Duży nacisk położono na responsywność interfejsu, co pozwala na płynne dostosowanie się do różnych rozmiarów i rozdzielczości ekranów, co jest kluczowe dla dostępności aplikacji na urządzeniach takich jak smartfony i tablety, zwłaszcza w kontekście mobilności i dostępności edukacyjnej. W planach jest udostępnienie aplikacji poprzez jej hosting na serwerze, a także publikacja kodu źródłowego wraz z dokumentem LaTeX na GitHubie twórcy [19]. Głównym zamierzeniem aplikacji jest wsparcie edukacyjne, szczególnie w zakresie nauki arytmetyki modularnej, która jest istotną częścią programu nauczania przedmiotu Matematyka Dyskretna 2, obowiązkowego w drugim semestrze studiów informatycznych na wydziale Zastosowań Informatyki i Matematyki w Szkole Głównej Gospodarstwa Wiejskiego w Warszawie. Uruchomienie aplikacji będzie możliwe poprzez plik `index.html`. Aplikacja składa się również z dodatkowych plików HTML, odpowiadających za różne tematy, oraz pliku `styles.css`, który zapewnia estetyczny i intuicyjny interfejs użytkownika, z odpowiednią kolorystyką, wielkością elementów oraz ich układem [20].
W aplikacji kluczową rolę odgrywają pliki JavaScript, które są odpowiedzialne za weryfikację poprawności obliczeń. Dla każdego zadania zaimplementowano dedykowane skrypty, które obsługują specyficzne algorytmy. Te algorytmy są niezbędne do przeprowadzania obliczeń i zapewnienia interaktywności aplikacji. W celu lepszego zobrazowania funkcjonalności aplikacji, poniżej zaprezentowane są zrzuty ekranu ilustrujące działanie poszczególnych komponentów strony.

5.2 Strona główna aplikacji

Rysunek 5.1. Ekran główny aplikacji. Źródło: opracowanie własne.

Strona główna aplikacji została zaprojektowana w sposób umożliwiający łatwą nawigację oraz intuicyjne zapoznanie się z podstawowymi zasadami arytmetyki modularnej. Jej układ harmonijnie łączy wstęp teoretyczny z praktycznymi aspektami nawigacji.
Centralna część ekranu powitalnego zawiera wstęp do świata arytmetyki modularnej. Ten segment stanowi podstawę dla użytkowników, dostarczając im kluczowych informacji i wprowadzając w tematykę arytmetyki modularnej. Ta przestrzeń umożliwia użytkownikom rozpoczęcie edukacji, zapewniając podstawowe umiejętności potrzebne do dalszego zgłębiania tematu. Po lewej stronie ekranu znajduje się nawigacja, która umożliwia łatwe przemieszczanie się między różnymi tematami arytmetyki modularnej. Menu nawigacyjne zawiera linki do poszczególnych tematów. Ta część interfejsu użytkownika została zaprojektowana w taki sposób, aby zapewnić szybki dostęp do różnorodnych tematów, umożliwiając użytkownikom dowolność w wyborze materiału.

5.3 Strony teorii



Rysunek 5.2. Przykładowa strona teorii. Źródło: opracowanie własne.

Powyższa strona służy jako wszechstronny przewodnik po wybranym temacie arytmetyki modularnej, stanowiąc źródło szczegółowej wiedzy teoretycznej, precyzyjnej terminologii oraz praktycznych przykładów. Jej projekt został starannie przemyślany, aby zagwarantować użytkownikom solidne i spójne rozumienie prezentowanych zagadnień arytmetyki modularnej.

Strona ta skrupulatnie wyjaśnia każde pojęcie i termin związany z arytmetyką modularną, od fundamentalnych definicji takich jak “modulo", aż po bardziej skomplikowane koncepcje. Dzięki temu użytkownicy mogą z łatwością przyswajać i stosować te pojęcia w praktyce.

Uwaga do rysunku 5.2: Niniejsza strona aplikacji jest jedną z ośmiu stron, która wyjaśnia teorię arytmetyki modularnej. Na rysunku jest przedstawiona przykładowa strona z teorią.

5.4 Strony z zadaniami

Rysunek 5.3. Przykładowa strona z zadaniami podstawowymi. Źródło: opracowanie własne.

Każda z szesnastu stron z zadaniami zawiera po cztery zadania, które umożliwiają użytkownikom stosowanie i utrwalanie wiedzy z zakresu arytmetyki modularnej. Zadania te zostały opracowane, aby zapewnić praktyczne doświadczenie w rozwiązywaniu problemów i stosowaniu teorii w zadaniach. Każde z zadań jest unikalnym problemem do rozwiązania i jest przedstawione w formie formularza, umożliwiając użytkownikom wprowadzenie własnych rozwiązań i bezpośrednie otrzymanie informacji zwrotnej.

W formularzu każdego zadania znajdują się następujące elementy:

- pola wprowadzania: pola do wpisywania odpowiedzi, gdzie użytkownicy mogą wprowadzać swoje odpowiedzi do danego zadania,
- przycisk sprawdzenia: przycisk umożliwiający przesłanie odpowiedzi i uzyskanie natychmiastowej informacji zwrotnej,
- komunikat zwrotny: miejsce, w którym użytkownik otrzymuje informacje o poprawności swojej odpowiedzi.

Uwaga do rysunku 5.3: Powyższa strona jest jedną z szesnastu stron, w tym ośmiu podstawowych i ośmiu zaawansowanych, na których można rozwiązywać zadania dotyczące arytmetyki modularnej. Na rysunku jest przedstawiona przykładowa strona z zadaniami podstawowymi.

6 Podsumowanie i wnioski

Głównym zamierzeniem przedstawionej pracy było stworzenie i wdrożenie aplikacji internetowej, mającej na celu wspieranie procesu edukacyjnego w obszarze arytmetyki modularnej. Opracowana aplikacja zapewnia wsparcie dydaktyczne dla studentów, którzy napotykają trudności w zrozumieniu matematyki na poziomie uniwersyteckim. Solidne podstawy teoretyczne, opisane na wstępie pracy, w połączeniu z opracowanymi algorytmami, zapewniają, że aplikacja efektywnie służy swoim celom edukacyjnym.

W ramach aplikacji zaprojektowano każdy temat, uwzględniając zarówno sekcję teoretyczną, jak i praktyczne zadania. Te dwa aspekty są ze sobą nierozerwalnie powiązane, umożliwiając użytkownikom skuteczne praktykowanie i stosowanie wiedzy teoretycznej w zadaniach o zróżnicowanym stopniu trudności, od podstawowego do zaawansowanego.

Istotnym wyzwaniem projektowym było zarządzanie strukturą plików projektu. Ostatecznie skompletowano czterdzieści trzy pliki, w tym szesnaście plików JavaScript, przeznaczonych do weryfikacji odpowiedzi użytkowników i generowania odpowiednich informacji zwrotnych. Dodatkowo, dwadzieścia sześć plików HTML odpowiada za właściwe wyświetlanie poszczególnych części aplikacji, a pojedynczy plik CSS kształtuje jej wygląd. Wszystkie te elementy wspólnie tworzą zintegrowaną całość aplikacji.

W perspektywie rozwoju aplikacji, możliwe jest dodanie nowych funkcjonalności, takich jak kalkulator pozwalający użytkownikom na wprowadzanie i obliczanie własnych danych. Możliwe jest również rozszerzenie zakresu edukacyjnego o nowe tematy i różnorodne typy zadań. Do rozważenia jest implementacja systemu logowania do śledzenia postępów oraz wprowadzenie elementu rywalizacji między użytkownikami. Aczkolwiek, nadrzędnym celem aplikacji jest aspekt edukacyjny, co skłoniło do skupienia na doskonaleniu algorytmów i eliminacji potencjalnych niedociągnięć, kosztem implementacji dalszych funkcji.

7 Bibliografia

[1] *Wikipedia. HTML.* https://pl.wikipedia.org/wiki/HTML [Dostęp 01.01.2024]

[2] *Wikipedia. Kaskadowe arkusze stylów.* https://pl.wikipedia.org/wiki/Kaskadowe_arkusze_styl%C3%B3w [Dostęp 01.01.2024]

[3] *Wikipedia. JavaScript.* https://pl.wikipedia.org/wiki/JavaScript [Dostęp 01.01.2024]

[4] Jakubiec A. *Prezentacje z wykładów z matematyki dyskretnej dla kierunku informatyka na Wydziale Zastosowań Informatyki i Matematyki SGGW.* 2017-2020.

[5] *RSA (kryptografia).* https://pl.wikipedia.org/wiki/RSA_(kryptografia) [Dostęp 01.01.2024]

[6] *Wikipedia. Modulo.* https://pl.wikipedia.org/wiki/Modulo [Dostęp 01.01.2024]

[7] Kennweth A. Ross, Charles R.B. Wright *Matematyka dyskretna.* Wydawnictwo naukowe PWN 2006, 2013, ISBN 978-83-01-14380-0.

[8] *Wikipedia. Arytmetyka modularna - przystawanie.* https://pl.wikipedia.org/wiki/Arytmetyka_modularna#Przystawanie [Dostęp 01.01.2024]

[9] Lewis H., Zax R. *Matematyka dyskretna. Niezbędnik dla informatyków.* Wydawnictwo naukowe PWN, 2021, ISBN 978-0-691-17929-2.

[10] *Wikipedia. Arytmetyka modularna - działania.* https://pl.wikipedia.org/wiki/Arytmetyka_modularna#Dzia%C5%82ania [Dostęp 01.01.2024]

[11] *Wikipedia. Algorytm Euklidesa - Rozszerzony algorytm Euklidesa.* https://pl.wikipedia.org/wiki/Algorytm_Euklidesa#Rozszerzony_algorytm_Euklidesa [Dostęp 01.01.2024]

[12] Grygiel J. *Wprowadzenie do matematyki dyskretnej.* Akademicka Oficyna Wydawnicza EXIT, 2007, ISBN 83-60434-26-3.

[13] *Ważniak. Teoria liczb II - Rozwiązywanie równań modularnych.* https://wazniak.mimuw.edu.pl/index.php?title=Matematyka_dyskretna_1/Wyk%C5%82ad_11:_Teoria_liczb_II#Rozwi%C4%85zywanie_r%C3%B3wna%C5%84_modularnych [Dostęp 01.01.2024]

[14] Zembrzuski A. *Materiały pomocnicze do ćwiczeń z Matematyki Dyskretnej 2 na Wydziale Zastosowań Informatyki i Matematyki Szkoły Głównej Gospodarstwa Wiejskiego w Warszawie.* 2022.

[15] *Wikipedia. Funkcja φ.* https://pl.wikipedia.org/wiki/Funkcja_%CF%86 [Dostęp 01.01.2024]

[16] *Wikipedia. Małe twierdzenie Fermata.* https://pl.wikipedia.org/wiki/Ma%C5%82e_twierdzenie_Fermata [Dostęp 01.01.2024]

[17] *Wikipedia. Twierdzenie Eulera.* https://pl.wikipedia.org/wiki/Twierdzenie_Eulera_(teoria_liczb) [Dostęp 01.01.2024]

[18] Thau D. JavaScript. *Podręcznik tworzenia interaktywnych stron internetowych. Wydanie II.* Helion, 2007, ISBN 978-83-246-1079-2.

[19] *GitHub.* Odnośnik do konta z pracą. https://github.com/konradmal [Dostęp 01.01.2024]

[20] Duckett J. *HTML i CSS. Zaprojektuj i zbuduj witrynę WWW. Podręcznik Front-End Developera.* Helion, 2011, ISBN 978-83-283-4481-5.

Wyrażam zgodę na udostępnienie mojej pracy w czytelniach Biblioteki SGGW w tym w Archiwum Prac Dyplomowych SGGW.

. . . . . .
(czytelny podpis autora pracy)