



Wzorcowe materiały dydaktyczne w zakresie:

# INFORMATYKA

## POZIOM – SZKOŁA PONADPODSTAWOWA

Maciej M. Sysło  
syslo@ii.uni.wroc.pl  
<http://mmsyslo.pl>

## 6. Materiały szkoleniowe dla kształcenia informatycznego w szkole ponadpodstawowej

W szkole ponadpodstawowej, wszystkich uczniów obowiązuje przedmiot informatyka w zakresie podstawowym (po jednej godzinie tygodniowo przez trzy lata). Mogą również wybrać informatykę w zakresie rozszerzonym. W obu przypadkach mogą planować zdawanie egzaminu maturalnego z informatyki. W rzeczywistości, uczniów, którzy nie wybrali rozszerzonej informatyki informatyka w zakresie podstawowym również mniej interesuje, gdyż wybrali inny przedmiot w zakresie rozszerzonym. Z tego powodu informatyka w zakresie podstawowym powinna oferować uczniom tematy, które mogą się wiązać z ich nieinformatycznymi zainteresowaniami. Polecamy książkę [2] (była podręcznikiem dla LO), w której zaproponowano wiele interesujących tematów z innych obszarów szkolnej edukacji. Jeden z nich przed-

stawiamy w punkcie 6.2. Wiele tematów z innych przedmiotów jest zawartych również w książce [1] – zostały one wybrane z materiałów dla gimnazjów, ale nadają się również dla obecnej szkoły ponadpodstawowej. Jeden z takich tematów dotyczący obliczeń finansowych przedstawiamy w punkcie 6.1, a inny temat, związany z przybliżonymi obliczeniami, omawiamy w punkcie 6.9. Tematy 6.1 i 6.9 uczniowie realizacją, zapisując obliczenia w arkuszu kalkulacyjnym, a faktycznie go programując, natomiast w realizacji tematu 6.2 posługują się dokumentem tekstowym umieszczonym w chmurze. Inne tematy jednostek tematycznych w tym rozdziale powinny w pewnym zakresie być również interesujące dla uczniów, dla których informatyka nie jest pierwszą dziedziną zastosowań.

### 6.1 Obliczenia finansowe w arkuszu kalkulacyjnym

#### Temat jednostki:

#### Obliczenia finansowe w arkuszu kalkulacyjnym

#### Poziom kształcenia

Szkoła ponadpodstawowa, informatyka – zakres podstawowy, przedsiębiorczość

#### Streszczenie

Informatyka dostarcza narzędzi do analizy danych i wyciągania wniosków na podstawie symulacji obliczeń według odpowiednich formuł. Tym najbardziej dogodnym i dostępnym w szkole jest arkusz kalkulacyjny, który jest wykorzystywany powszechnie również w obliczeniach profesjonalnych i komercyjnych. Arkusz jest więc narzędziem o niemal nieograniczonych możliwościach obliczeń, analizy danych, symulacji modeli, w tym finansowych, co jest prezentowane w tej jednostce tematycznej. Arkusz można programować w specjalnym języku, ale tutaj korzystamy jedynie z formuł (wzorów) naturalnie dostępnych. Jakikolwiek jednak wykorzystanie arkusza w obliczeniach można traktować jako jego programowanie.

Arkusz kalkulacyjny regularnie pojawia się na egzaminie maturalnym z informatyki jako narzędzie analizy danych i wyciągania wniosków na podstawie danych analizowanych formułami arkusza.

#### Przygotowanie uczniów

Uczeń potrafi zapisać w arkuszu kalkulacyjnym proste obliczenia, posługując się jego formułami.

#### Cele szczegółowe zajęć – osiągnięcia uczniów

##### Uczeń:

- analizuje praktyczne problemy i na tej podstawie podejmuje decyzje,
- wypełnia arkusz kalkulacyjny odpowiednimi danymi i formułami: tworzy tabele, stosuje adresy względne i bezwzględne, wypełniania komórki seriami danych, dobiera wykresy do danych,
- dobiera formuły obliczeniowe i strukturę arkusza odpowiednio do zaplanowanych obliczeń,

- wykonuje symulacje obliczeń zapisanych w arkuszu i ewentualnie koryguje arkusz,
- wyciąga wnioski z wykonanych obliczeń odnoszące się do postawionych problemów.

### Treści programowe – wypisy z podstawy programowej informatyki

II. Programowanie i rozwiązywanie problemów z wykorzystaniem komputera [...].

Zakres podstawowy. Uczeń:

3. przygotowuje opracowania rozwiązań problemów, posługując się wybranymi aplikacjami:
  - c. gromadzi dane pochodzące z różnych źródeł w tabeli arkusza kalkulacyjnego, korzysta z różnorodnych funkcji arkusza w zależności od rodzaju danych, [...], analizuje dane, [...].

### Pojęcia i metody informatyczne

- obliczenia finansowe, w tym m.in. procent składany,
- algorytmy i obliczenia w arkuszu,
- symulacja obliczeń w arkuszu,

### Metody pracy w klasie

- **Słowna i Oglądowa:** kierowana przez nauczyciela dyskusja w klasie na temat dwóch podstawowych operacji bankowych, oszczędzania i spłacania kredytu; uczniowie bazują na wzorach, z którymi zapoznali się na matematyce lub przedsiębiorczości.
- **Czynna:** uczniowie indywidualnie zapisują w arkuszu kalkulacyjnym dane dotyczące wielkości finansowych oraz formuły obliczeń.
- **Aktywizująca:** uczniów aktywizują stawiane przez nauczyciela pytania dotyczące analizy różnych sytuacji finansowych; sami z ciekawością planują hipotetyczne oszczędzanie lub zadłużanie się w banku.

### Formy pracy uczniów

- zajęcia w grupie (cała klasa) i w podgrupach, poza komputerami, dyskusje nad postawionymi przez nauczyciela problemami finansowymi,
- praca indywidualna – zapisanie w arkuszu kalkulacyjnym algorytmu obliczeń finansowych, wykonanie eksperymentów obliczeniowych.

### Przygotowanie do zajęć – materiały, urządzenia, oprogramowanie

Ponieważ temat tych zajęć odnosi się do problemów praktycznych, nauczyciel powinien zgromadzić rzeczywiste dane, by zilustrować problemy, którymi mają zajmować się uczniowie. Te konkretne dane powinny uzasadnić ważność tematyki i umotywić uczniów do zajęć na podstawowe tematy analizy finansowej przeciętnego obywatela, jakimi są oszczędzanie i zapożyczanie się w banku.

Poza rzeczywistymi materiałami bankowymi nauczyciel powinien dysponować własnymi rozwiązaniami, by posłużyć się nimi w prezentacji problemów, będących przedmiotem zajęć.

### Uwagi metodyczne dla nauczyciela dotyczące realizacji przyjętych celów i zapisów podstawy programowej

Odsyłamy do szczegółowego omówienia tematów oszczędzania i spłat kredytu w rozdz. 2 w książce [1]. Zamieszczamy tutaj tylko podstawowe informacje i sugestie.

#### *Lokata kapitału*

Lokata pieniędzy na koncie bankowym przynosi właścicielowi konta zysk, który zależy m.in. od następujących czynników:

- kwoty oszczędności,
- oprocentowania, podawanego przez banki na ogół w skali rocznej,
- długości okresu kapitalizacji odsetek, czyli okresu, po upływie którego kwota odsetek jest dopisywana przez bank do lokaty,
- długości okresu przechowywania pieniędzy w banku,
- decyzji o podejmowaniu bądź niepodejmowaniu odsetek z banku.

W najprostszym przypadku, gdy:

- właściciel konta nie podejmuje nagromadzonych odsetek,
- oprocentowanie wynosi  $p$ ,
- kapitalizacja odbywa się w skali rocznej,
- wysokość lokaty na koncie po  $n$  latach, przy kwocie  $k$  na początku, jest obliczana ze wzoru na **procent składany**:

$$k(1 + p/100)^n$$

Ten wzór jest rachunkowo skomplikowany, głównie z powodu występującego w nim potęgowania liczb ułamkowych, arkusz wydaje się być idealnym narzędziem do zaplanowania oszczędzania.

Popularne są dwie strategie oszczędzania:

- odsetki pozostają na koncie w banku i są oprocentowane tak jak początkowa lokata,
- odsetki są pobierane z banku każdego roku, lub innymi słowy, oprocentowana jest jedynie początkowa lokata – jest to **procent prosty**.

Poniżej są pokazane fragmenty arkusza przygotowanego do symulacji strategii oszczędzania: odsłonięte formuły i wykonane obliczenia.

Lokata w dwóch wariantach - formuły						
	A	B	C	D	E	F
1						
2		Lokata	200			
3		Oprocentowanie	10			
4						
5		Okres oszczędzania (w latach)	Odsetki pozostawione na koncie		Odsetki pobierane każdego roku	
6			Stan konta	Odsetki	Stan konta	Odsetki
7	1		=C\$2*(1+C\$3/100)	=C7-C\$2	=C\$2	=C\$2*C\$3/100
8	2		=C7*(1+C\$3/100)	=C8-C\$2	=C\$2	=F7+C\$2*C\$3/100

Lokata w dwóch wariantach						
	A	B	C	D	E	F
1						
2		Lokata	200,00 zł			
3		Oprocentowanie	10			
4						
5		Okres oszczędzania (w latach)	Odsetki pozostawione na koncie		Odsetki pobierane każdego roku	
6			Stan konta	Odsetki	Stan konta	Odsetki
7	1		220,00 zł	20,00 zł	200,00 zł	20,00 zł
8	2		242,00 zł	42,00 zł	200,00 zł	40,00 zł
9	3		266,20 zł	66,20 zł	200,00 zł	60,00 zł
10	4		292,82 zł	92,82 zł	200,00 zł	80,00 zł
11	5		322,10 zł	122,10 zł	200,00 zł	100,00 zł
12						

### Splata rat kredytu bankowego

**Kredyt** to pożyczka z banku, którą pożyczkobiorca musi zwrócić bankowi i dodatkowo jeszcze zapłacić za tę usługę. W zamian za pożyczkę bank pobiera oprocentowanie, które na ogół jest o kilka punktów procentowych wyższe od najwyższego oprocentowania lokat przechowywanych w banku.

Bank oprocentowuje kredyt zależnie od wysokości kwoty i deklarowanego okresu spłaty. Ponadto przedstawia kredytobiorcy do wyboru różne **plany spłat** kredytu, określające wysokość miesięcznych rat. W większości planów spłat ustala się roczne oprocentowanie kredytu, które na ogół jest stałe przez cały okres spłacania. Różne plany spłat różnią się sposobem spłacania odsetek. Oto przykładowe sposoby doliczania odsetek do spłat:

- Odsetki są płacone z góry, czyli w chwili otrzymania kredytu. Wtedy otrzymujemy na rękę kwotę pożyczki pomniejszoną o odsetki, a całą kwotę kredytu spłacamy w równych miesięcznych ratach. Jest to najmniej korzystny układ dla pożyczkobiorcy.
- Do każdej miesięcznej raty spłacanego kredytu bank dolicza odsetki od spłacanej w danym miesiącu raty za cały okres zadłużenia tej raty. A zatem w kolejnych miesiącach spłacane raty są coraz większe, gdyż spłacana w danym miesiącu kwota dłużej pozostaje zadłużeniem.
- Do każdej miesięcznej raty spłacanego kredytu bank dolicza odsetki należne mu za jeden miesiąc bieżącej kwoty zadłużenia. A zatem w kolejnych miesiącach spłacane raty są coraz mniejsze, gdyż kwota zadłużenia maleje w każdym miesiącu.

Poniżej ilustrujemy przykładowy plan spłaty kredytu:

Plan spłat kredytu						
	A	B	C	D	E	F
<b>1</b>	<b>PLAN SPŁAT KREDYTU</b>					
<b>2</b>	Kwota kredytu				20 000,00 zł	
<b>3</b>	Oprocentowanie:		roczne:	18	miesięczne:	1,5
<b>4</b>	Liczba miesięcy spłaty				12	
<b>5</b>	Spłacana kwota miesięcznie				1 666,67 zł	
<b>7</b>			Odsetki od spłacanej raty kredytu		Odsetki od aktualnej kwoty zadłużenia	
<b>8</b>	Kolejne miesiące spłaty	Spłacana kwota kredytu	Spłacane odsetki	Miesięczna kwota spłaty kredytu	Spłacane odsetki	Miesięczna kwota spłaty kredytu
<b>9</b>						
<b>10</b>	1	1 666,67 zł	25,00 zł	1 691,67 zł	300,00 zł	1 966,67 zł
<b>11</b>	2	1 666,67 zł	50,00 zł	1 716,67 zł	275,00 zł	1 941,67 zł
<b>12</b>	3	1 666,67 zł	75,00 zł	1 741,67 zł	250,00 zł	1 916,67 zł
<b>13</b>	4	1 666,67 zł	100,00 zł	1 766,67 zł	225,00 zł	1 891,67 zł
<b>14</b>	5	1 666,67 zł	125,00 zł	1 791,67 zł	200,00 zł	1 866,67 zł
<b>15</b>	6	1 666,67 zł	150,00 zł	1 816,67 zł	175,00 zł	1 841,67 zł
<b>16</b>	7	1 666,67 zł	175,00 zł	1 841,67 zł	150,00 zł	1 816,67 zł
<b>17</b>	8	1 666,67 zł	200,00 zł	1 866,67 zł	125,00 zł	1 791,67 zł
<b>18</b>	9	1 666,67 zł	225,00 zł	1 891,67 zł	100,00 zł	1 766,67 zł
<b>19</b>	10	1 666,67 zł	250,00 zł	1 916,67 zł	75,00 zł	1 741,67 zł
<b>20</b>	11	1 666,67 zł	275,00 zł	1 941,67 zł	50,00 zł	1 716,67 zł
<b>21</b>	12	1 666,67 zł	300,00 zł	1 966,67 zł	25,00 zł	1 691,67 zł
<b>22</b>	<b>Razem:</b>	<b>20 000,00 zł</b>	<b>1 950,00 zł</b>	<b>21 950,00 zł</b>	<b>1 950,00 zł</b>	<b>21 950,00 zł</b>

## Przebieg zajęć

- Dyskusja w całej klasie.** Nauczyciel przedstawia dwa zagadnienia związane z bankowymi operacjami finansowymi, którym będą poświęcone zajęcia.  
Prezentacja nauczyciela i dyskusja w klasie ma doprowadzić do sformułowania dokładnych zasad oszczędzania w banku oraz zasad udzielania przez bank kredytów. Zasady te powinny mieć postać liczbową, by można było uwzględnić je w obliczeniach.
- Indywidualna praca uczniów.** Po ustaleniu zasad oszczędzania w banku i udzielania przez bank kredytów, każdy uczeń tworzy najpierw arkusz dla oszczędzania, a później do pobrania pożyczki z banku.  
Po utworzeniu odpowiednich arkuszy uczniowie wykonują ich symulacje dla różnych wielkości kwot i oprocentowania. Finalnie starają się utworzyć arkusz najbardziej odpowiadający ich hipotetycznym potrzebom.
- Nauczyciel.** Sugeruje odszukanie w mediach rzeczywistych ofert banków i uwzględnienia ich w arkuszach uczniów.
- Indywidualna praca uczniów.** Ponownie uczniowie pracują tym razem z ofertami banków i określają najkorzystniejszą z nich.  
Inne zadania  
*Można zaproponować uczniom jeszcze inne zadania.* Utwórz arkusz, w którym będziesz śledził swoje oszczędności w okresie uczęszczania do klasy/szkoły przez określoną liczbę lat, by uzbierać na nowy sprzęt elektroniczny lub nowy sprzęt sportowy. Przyjmij wysokość miesięcznej wpłaty ze swojego kieszonkowego, ustal realne oprocentowanie i utwórz wykres ilustrujący, jak będzie zmieniać się wysokość lokaty na koncie.
- Indywidualna praca uczniów i ich ocena.** W przypadku zadań programistycznych (także w arkuszu), nauczyciel motywuje uczniów, wyżej oceniając ich wkład, jeśli program (arkusz) został utworzony samodzielnie (przez ucznia lub w parze z innym uczniem), nie korzystając z gotowych rozwiązań, których obecnie jest bardzo wiele dostępnych w sieci.
- Dyskusja podsumowująca.** Dyskusja może dotyczyć zarówno przydatności arkusza do prowadzenia obliczeń, jak i strony finansowej omawianych zagadnień. Wykonane symulacje powinny umożliwić uczniom określenia najkorzystniejszych warunków finansowych. W dyskusji uczniowie mogą się pochwalić najkorzystniejszymi dla siebie rozwiązaniami.

## Modyfikacja, poszerzenie zakresu, kontynuacja zajęć

Zajęcia na tych lub kolejnych lekcjach można poszerzyć o inne zasady oszczędzania lub udzielania kredytów. Uczniowie mogą uzyskać dokładne i rzeczywiste dane, kontaktując się z przedstawicielem banku lub ekspertem bankowym.

## 6.2 Rozbudowany dokument tekstowy

### Temat jednostki:

#### Rozbudowany dokument tekstowy

### Poziom kształcenia

Szkoła ponadpodstawowa, informatyka – zakres podstawowy, język polski

### Streszczenie

Jest to bardzo nietypowa jednostka zajęć informatycznych, jej głównym celem informatycznym jest przygotowanie przez grupę uczniów rozbudowanego dokumentu tekstowego, który będzie zapisem realizacji projektu polegającego na przeprowadzeniu dyskusji na zadany temat. Tym projektem chcemy zwrócić uwagę uczniów, jak również nauczycieli, że nawet w przypadku projektów na tematy dalekie od informatyki, tutaj – literackie, realizowane ze wsparciem technologii, niezmiernie przydatnie jest myślenie komputacyjne. Jest to jednocześnie potwierdzeniem naczelnej tezy konstrukcjonizmu, wyrażonej w 1970 roku przez Seymoura Paperta, inicjatora tego kierunku rozwoju kształcenia:

*children learn by doing and by thinking about what they do*  
dzieci uczą się przez działanie i myśląc o tym, co robią

Opisane tutaj zajęcia były jednym z projektów w podręczniku [2], p. 4.1.

### Przygotowanie uczniów

Oczekiwana jest umiejętność korzystania przez uczniów z edytora tekstu na podstawowym poziomie oraz sprawnego wyszukiwania w sieci.

### Cele szczegółowe zajęć – osiągnięcia uczniów

#### Uczeń:

- wyszukuje informacje na ustalony temat w sieci, w bibliotekach tradycyjnych i dostępnych w sieci, w materiałach drukowanych,
- kolekcjonuje, porządkuje i indeksuje informacje oraz zasoby informacji na ustalony temat,
- posługuje się zaawansowanymi możliwościami edytora tekstu przy tworzeniu rozbudowanego dokumentu i w pracy nad tekstem,
- pracuje wraz z zespołem nad rozbudowanym tekstem dostępnym w chmurze.

### Treści programowe – wypisy z podstawy programowej informatyki

II. Programowanie i rozwiązywanie problemów z wykorzystaniem komputera [...].

Zakres podstawowy. Uczeń:

3. przygotowuje opracowania rozwiązań problemów, posługując się wybranymi aplikacjami:
  - b. opracowuje dokumenty o różnorodnej tematyce, [...], i o rozbudowanej strukturze, posługując się przy tym konspektem dokumentu, dzieli tekst na sekcje i kolumny, tworzy spisy treści, rysunków i tabel, stosuje własne style i szablony, pracuje nad dokumentem w trybie recenzji, [...].

### Pojęcia i metody informatyczne

- dokument tekstowy o rozbudowanej strukturze: konspekt dokumentu, tekst w sekcjach i kolumnach, spisy treści, rysunków i tabel, własne style i szablony, tryb recenzji,
- praca nad tekstem,
- praca zespołowa nad dokumentem w chmurze,
- abstrakcja, dekompozycja, uogólnienie,

## Metody pracy w klasie

- **Słowna i Oglądowa:** Nauczyciel przedstawia listę tematów, nad którymi mogą dyskutować grupy uczniów. Każdy z tematów umożliwia zajęcie jednego z dwóch stanowisk. Uczniowie jednocześnie wybierają tematy i dobierają się w grupy. W grupie uczniów, którzy wybrali dany temat, muszą znaleźć się uczniowie o przeciwstawnych poglądach.
- **Czynna:** Uczniowie dyskutują i pracują w grupach nad dokumentem na wybrany przez siebie temat; wspólnie tworzą rozbudowany dokument będący relacją z dyskusji.
- **Aktywizująca:** Aktywizacja uczniów jest powodowana przez wytaczanie coraz to nowych argumentów w dyskusji przez członków grupy o innych poglądach.

## Formy pracy uczniów

- zajęcia w grupach poza komputerami – dyskusje i zbieranie materiałów do dyskusji i do relacji z dyskusji,
- praca indywidualna – spisywanie relacji z dyskusji przez pojedynczych uczniów we wspólnym rozbudowanym dokumencie;

## Przygotowanie do zajęć – materiały, urządzenia, oprogramowanie

Nauczyciel przygotowuje listę tematów do dyskusji w grupach uczniów. Tematy powinny mieć związek z informatyką, technologią, społeczeństwem informacyjnym i umożliwiać zajęcie dwóch przeciwstawnych stanowisk przez członków grupy. Przykładowe tematy:

1. (Umberto Eco) Grozi nam, że cały dzisiejszy przemysł informatyczny, mnożąc informacje, nie będzie dostarczał już żadnej.
2. (Jean Claude Carriere, Umberto Eco) Nie myśl, że książki znikną.
3. (Jean Claude Carriere, Umberto Eco) Internet sprawił, że powróciliśmy do epoki alfabetu. Jeśli przypadkiem uznaliśmy, że oto nastąpiła cywilizacja obrazu, za sprawą komputera ponownie odkrywamy epokę Gutenberga, no i w rezultacie wszyscy jesteśmy zmuszeni do czytania.
4. Prawdziwa wiedza zaczyna się tam, gdzie Google nie sięga.
5. (Noam Chomsky) Internet jest niezwykłą rzeczą, jeśli wiesz, czego szukasz. (Wolfgang Goethe) Znajduję to, co znam.

Nauczyciel zapewnia również, że uczniowie mają dostęp do środowiska, w którym mogą tworzyć dokumenty w chmurze.

## Uwagi metodyczne dla nauczyciela dotyczące realizacji przyjętych celów i zapisów podstawy programowej

Przebieg tych zajęć jest szczegółowo opisany w p. 4.1 podręcznika [2].

Każda grupa ma przygotować rozbudowany dokument, zawierający m.in. tezy każdej ze stron wraz z uzasadnieniami. W dokumencie należy umieścić odnośniki do źródeł (tekstów) wiążących się z tematem dyskusji, z których korzystano.

Zwykle zadanie jest realizowane sprawniej, jeśli podzieli się je na mniej złożone etapy. Ponadto gdy zadanie jest realizowane w zespole, to każdy etap może realizować inny członek zespołu. Poszczególne etapy realizacji tego zadania, którego głównym produktem informatycznym ma być rozbudowany dokument, są przedstawione dalej w Przebiegu zajęć.

Tutaj zwrócimy uwagę, jak sposoby rozumowania zaliczane do myślenia komputacyjnego uczestniczą w realizacji tego zadania. Szerzej jest o tym mowa w książce [6].

Zadanie, projekt	Polemika z wypowiedzią Umberta Eco: <i>jeśli ktoś myśli, że książka zniknie, to się myli</i>
<b>Sytuacja problemowa</b>	teksty drukowane i elektroniczne związane z tematem
	znajdź argumenty za i przeciw; przeprowadź dyskusję, spisz ją
Dane	selekcja/wybór fragmentów ze źródeł – <b>abstrakcja</b>
Zasada w danych	dotyczą książki, w tym pióra U. Eco
<b>Reprezentacja danych</b>	szablon tekstu dyskusji z przeciwnymi argumentami (tabela, format)
<b>Dekompozycja</b>	argumenty „za” i „przeciw”
<b>Algorytm</b>	metoda postępowania: zorganizowana dyskusja, uporządkowany zapis, rozbudowany dokument
<b>Modyfikacje danych</b>	ważny dla konkluzji głos innych osób – uzasadnienie, włączenie
Komputer, program, projekt	<b>automatyzacja</b> przebiegu projektu, ocena własna i nauczyciela, udostępnienie w systemie prowadzenia projektów

## Przebieg zajęć

1. *Dyskusja w całej klasie.* Nauczyciel przedstawia zadanie, jakie stoi przed klasą. Uczniowie dyskutują o szczegółach realizacji.  
Nauczyciel przedstawia tematy do wyboru. Nauczyciel może zaakceptować tematy zasugerowane przez uczniów, które mogą ich bardziej interesować.
2. *Praca uczniów w zespołach nad wybranymi tematami.* Głównym informatycznym efektem pracy zespołu ma być rozbudowany dokument, będący zapisem dyskusji i zgromadzonych argumentów.  
Proponowane etapy pracy zespołów z wykorzystaniem edytora tekstu:
  - przygotowanie konspektu dokumentu,
  - wypełnienie dokumentu treścią zgodnie z utworzonym konspektem. Można przy tym utworzyć własne style nagłówków oraz tekstu i posłużyć się nimi,
  - zapisanie dokumentu w postaci szablonu i udostępnienie tego szablonu wszystkim członkom zespołu,
  - przygotowanie spisu treści i umieszczenie go w dokumencie,
  - udostępnienie dokumentu,
  - prezentacja projektu.
3. *Aktywność nauczyciela.* Nauczyciel śledzi przebieg realizacji zadań przez zespoły, doradza odnośnie do etapów podejmowanych działań zgodnie z podaną listą i elementami myślenia komputacyjnego.  
Utworzenie dokumentu o rozbudowanej strukturze może wymagać posłużenia się: konspektem, rozbicia tekstu na sekcje i kolumny, wstawienia spisu treści, rysunków i tabel, użycia własnych styli i szablonów, posłużenia się trybem recenzji przy korekcie ostatecznej postaci dokumentu. Nauczyciel służy radą i wsparciem przy korzystaniu z tych funkcjonalności edytora tekstu.
4. *Indywidualna praca uczniów i ich ocena.* W przypadku zadań z aplikacjami komputerowymi nauczyciel motywuje uczniów, wyżej oceniając ich wkład, jeśli jest ich samodzielnym dziełem (przez ucznia lub w grupie z innymi uczniami), nie korzystając z gotowych rozwiązań, których obecnie jest bardzo wiele dostępnych w sieci.
5. *Dyskusja podsumowująca zajęcia.* Może przyjąć formę relacji z przebiegu realizacji zadań przez poszczególne zespoły i prezentacji utworzonych dokumentów o rozbudowanej strukturze.

## Modyfikacja, poszerzenie zakresu, kontynuacja zajęć

Propozycja tej jednostki jest bardzo pojemna, umożliwia łatwe modyfikacje tematyczne, zarówno poszerzenie zakresu, jak i kontynuowanie o kolejne elementy pracy projektowej w grupie.



## 6.3 Obliczenia przybliżone

### Temat jednostki:

#### Obliczenia przybliżone

### Poziom kształcenia

Szkoła ponadpodstawowa, informatyka – zakres rozszerzony, matematyka

### Streszczenie

Może się wydawać, że komputery są w stanie rozwiązać każdy problem, jaki zostanie im przedstawiony w języku, który jest dla nich zrozumiały, np. w postaci programu. Niezupełnie, są bowiem przynajmniej dwa wyjątki od tego stwierdzenia, o których powinni wiedzieć uczący się informatyki. Po pierwsze, jest wiele problemów, dla rozwiązania których moc dzisiejszych komputerów, nawet superkomputerów, jest niewystarczająca dzisiaj i w dającej się przewidzieć przyszłości. Przykładem takiego problemu jest problem optymalnego upakowania plecaka, o którym wspominamy w jednostce tematycznej poświęconej programowaniu dynamicznemu. W przypadku takiego problemu stosowane są algorytmy, które dają rozwiązania przybliżone w tym sensie, że nie są one najlepsze, ale niezbyt daleko odbiegają od najlepszego. Inną grupę problemów stanowią te, dla których, np. na podstawie pewnych faktów matematycznych, nie jest możliwe otrzymanie rozwiązania dokładnego, wtedy musimy zadowolić się w miarę dobrym przybliżeniem.

Ta jednostka tematyczna jest poświęcona problemowi z tej drugiej grupy – obliczaniu wartości pierwiastka kwadratowego, której jak wiadomo z matematyki, nie zawsze można dokładnie obliczyć, np. wartości pierwiastka kwadratowego z liczby 2.

W wielu podręcznikach do matematyki znajduje się gotowy wzór na kolejne przybliżenia pierwiastka. Tutaj wyprowadzimy i uzasadnimy ten wzór w prostym rozumowaniu, a następnie zaproponujemy eksperymenty obliczeniowe, by ocenić, jak dobre przybliżenia może generować.

### Przygotowanie uczniów

Uczeń potrafi zapisać w arkuszu kalkulacyjnym proste obliczenia, posługując się jego formułami.

### Cele szczegółowe zajęć – osiągnięcia uczniów

#### Uczeń:

- modeluje geometrycznie rozwiązanie problemu,
- wypełnia arkusz kalkulacyjny odpowiednimi danymi i formułami: tworzy tabele, stosuje adresy względne i bezwzględne, wypełnia komórki seriami danych,
- dobiera formuły obliczeniowe i strukturę arkusza odpowiednio do zaplanowanych obliczeń,
- wykonuje symulacje obliczeń zapisanych w arkuszu i ewentualnie koryguje arkusz,
- wyciąga wnioski z wykonanych obliczeń odnoszące się do postawionych problemów,
- uogólnia problem i jego rozwiązanie.

### Treści programowe – wypisy z podstawy programowej informatyki

II. Programowanie i rozwiązywanie problemów z wykorzystaniem komputera [...].

Zakres podstawowy. Uczeń:

3. przygotowuje opracowania rozwiązań problemów, posługując się wybranymi aplikacjami:
  - c. gromadzi dane pochodzące z różnych źródeł w tabeli arkusza kalkulacyjnego, korzysta z różnorodnych funkcji arkusza w zależności od rodzaju danych, [...], analizuje dane, [...].

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

1. zapisuje za pomocą listy kroków, schematu blokowego lub pseudokodu i implementuje w wybranym języku programowania, [...] algorytmy:
  - g. obliczania przybliżonej wartości pierwiastka kwadratowego,

## Pojęcia i metody informatyczne

- obliczenia matematyczne,
- algorytmy i obliczenia w arkuszu,
- symulacja obliczeń w arkuszu.

## Metody pracy w klasie

- **Słowna i Oglądowa:** kierowana przez nauczyciela dyskusja w klasie na temat obliczania wartości pierwiastka kwadratowego; sugestia geometryczna, jak postąpić, gdy dokładna wartość pierwiastka kwadratowego nie jest znana.
- **Czynna:** uczniowie indywidualnie zapisują w arkuszu kalkulacyjnym wzór, który pojawił się w dyskusji i odpowiednio rozbudowują arkusz.
- **Aktywizująca:** uczniów aktywizuje ciekawość, jak zachowuje się zapisany w arkuszu algorytm; również sugestia nauczyciela, jak uogólnić otrzymany wzór dla obliczania wartości pierwiastka dowolnego stopnia.

## Formy pracy uczniów

- zajęcia w grupie (cała klasa) – dyskusja nad postawionym przez nauczyciela problemem matematycznym i geometrycznym sposobem jego modelowania,
- praca indywidualna – zapisanie w arkuszu kalkulacyjnym algorytmu i wykonanie eksperymentów obliczeniowych.

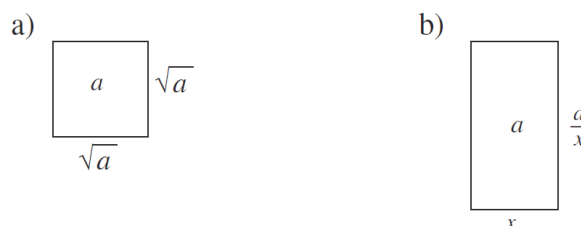
## Przygotowanie do zajęć – materiały, urządzenia, oprogramowanie

Nie są potrzebne jakieś specjalne materiały do przeprowadzenia tych zajęć.

## Uwagi metodyczne dla nauczyciela dotyczące realizacji przyjętych celów i zapisów podstawy programowej

Odsyłamy do szczegółowego omówienia sposobu obliczania wartości pierwiastka kwadratowego w p. 7.7 w książce [4]. Zamieszczamy tutaj tylko podstawowe informacje i sugestie.

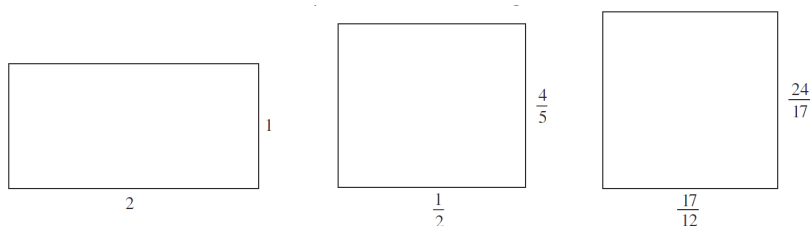
Wyprowadzenie wzoru na kolejne przybliżenia wartości pierwiastka kwadratowego jest przykładem modelowania geometrycznego. Na rysunku poniżej zilustrowano zasadnicze rozumowanie:



Oczywiste jest (rys. a)), że pierwiastek kwadratowy z  $a$  jest długością boku kwadratu o polu  $a$ . Nie znamy tej długości, ale chcemy ją obliczyć. Spróbujmy zgadnąć. Niech ta długość boku wynosi  $x$ , wtedy aby nie zmieniła się wielkość pola, drugi bok musi mieć długość  $a/x$ . Na ogół nie udaje się dobrze trafić z wartością  $x$ , co zilustrowano na rys. b). Jedno jest pewne, jeśli wartość  $x$  jest za mała, to wartość  $a/x$  jest za duża, a jeśli wartość  $x$  jest za duża, to wartość  $a/x$  jest za mała. A zatem szukana wartość pierwiastka kwadratowego z  $a$  leży gdzieś pomiędzy tymi wartościami. Ponieważ nie wiemy, gdzie leży, więc bierzemy średnią z tych dwóch wartości, czyli kolejne przybliżenie wartości pierwiastka kwadratowego z  $a$  ma postać:

$$(x + a/x)/2$$

Na rysunku poniżej zilustrowano trzy kolejne kroki dla  $a = 2$ .



Widać, że na kolejnych rysunkach prostokąt o polu 2 staje się coraz bardziej „kwadratowy”, można więc przypuszczać, że długość jego boku staje się coraz bliższa wartości pierwiastka kwadratowego z 2. W ramach tej jednostki lekcyjnej uczniowie mają się przekonać, że rzeczywiście tak jest, wykonując obliczenia w arkuszu, którego fragment może mieć postać jak poniżej:

Pierwiastek kwadratowy				
	A	B	C	D
1	Obliczanie wartości pierwiastka			
2	kwadratowego z liczby $a =$			
3				
4	Wartość dokładna pierwiastka z $a =$			
5				
6	Przybliżenie początkowe $x =$			
7				
8			Krok algorytmu	Wartość przybliżenia
9				
10			Początkowy	
11			1	
12			2	
13			3	
14			4	
15			5	
16			6	
17			7	
18			8	
19			9	
20			10	

Wyzwaniem dla uczniów może być poprowadzenie podobnego rozumowania, aby obliczyć wartość pierwiastka trzeciego stopnia z 2, czy z jakiegokolwiek innej liczby. *Podpowiedź:* w modelu geometrycznym należy posłużyć się sześcianem zamiast kwadratem.

### Przebieg zajęć

6. *Dyskusja w całej klasie.* Nauczyciel zaczyna dyskusję z uczniami na temat, jak obliczać wartość pierwiastka kwadratowego z jakiegokolwiek liczby. Pojawią się odpowiedzi dotyczące pełnych kwadratów, jak 4, 64 czy 400. Wartość pierwiastka kwadratowego z liczby 2 może być znana niektórym uczniom, zapewne tylko kilka początkowych cyfr (np. 1.4142), a chcielibyśmy znać ich więcej.

Aby otrzymać odpowiedni wzór na przybliżanie szukanej wartości, nauczyciel stara się skierować myślenie uczniów na interpretację geometryczną, naszkicowaną powyżej. W ten sposób zostaje otrzymany wzór, którym uczniowie posłużą się, by zaprojektować odpowiednie obliczenia w arkuszu.

7. *Indywidualna praca uczniów.* Po ustaleniu wzoru na kolejne przybliżenia każdy uczeń tworzy arkusz, w którym będą obliczane kolejne przybliżenia dla różnych liczb podpierwiastkowych oraz dla różnych początkowych przybliżeń. Struktura takiego arkusza jest pokazana powyżej.

Po utworzeniu odpowiednich arkuszy uczniowie wykonują obliczenia dla różnych danych. Ciekawe mogą być eksperymenty z różnymi wartościami początkowymi. Uczniowie mogą się przekonać, że nawet bezsensowne przybliżenia początkowe, np. 100 dla pierwiastka z 2, po niedużej liczbie iteracji prowadzi do dobrego przybliżenia.

Sprawdź, że dobrym przybliżeniem początkowym jest liczba  $(1 + a)/2$ . Nauczyciel sugeruje uczniom, by porównali dokładność wyników obliczeń dla tej liczby i na przykład dla przybliżenia początkowego równego  $a$ .

8. *Indywidualna praca uczniów.* Można zaproponować uczniom napisanie programu w Pythonie, który realizowałby obliczenia wykonywane tutaj w arkuszu.

W p. 7.7 w książce [4] podane są inne kryteria zakończenia obliczeń przybliżonych wartości – to odpowiednie zadania dla modyfikacji programu w Pythonie.

9. *Indywidualna praca uczniów i ich ocena.* W przypadku zadań programistycznych nauczyciel motywuje uczniów, wyżej oceniając ich wkład, jeśli program został utworzony samodzielnie (przez ucznia lub w parze z innym uczniem), nie korzystając z gotowych rozwiązań, których obecnie jest bardzo wiele dostępnych w sieci.
10. *Dyskusja podsumowująca.* Dyskusja może dotyczyć przydatności metod, które znajdują rozwiązania przybliżone. Uczniowie mogą dostarczyć argumentów ze swoich obliczeń, że takie metody dają całkiem niezłe przybliżenia.

Nauczyciel może podsumować dyskusję informując, że w komputerze tylko obliczenia w całości wykonywane na liczbach całkowitych dają dokładne wyniki, wszystkie inne obliczenia są przybliżone. Wynika to z faktu, że poza liczbami całkowitymi (typu *integer*) wszystkie inne liczby i wyniki operacji na nich są reprezentowane w komputerze w sposób przybliżony.

Dyskusja z uczniami może dotyczyć również uogólnienia rozważań w tej jednostce zajęć na obliczanie wartości pierwiastka trzeciego stopnia – należy również zacząć od modelu geometrycznego, ale w tym przypadku należy posłużyć się sześcianiem, a nie kwadratem.

### Modyfikacja, poszerzenie zakresu, kontynuacja zajęć

Jest wiele kierunków poszerzenia zakresu tej jednostki tematycznej, jedna taka możliwość została wymieniona na końcu dyskusji z uczniami. Innym kierunkiem rozważań mogą być zagadnienia dokładności reprezentacji liczb w komputerze i dokładności prowadzonych na nich obliczeń.

## 6.4 Schemat Hornera obliczania wartości wielomianu

### Temat jednostki:

#### Schemat Hornera obliczania wartości wielomianu

### Poziom kształcenia

Szkoła ponadpodstawowa, informatyka – zakres rozszerzony, matematyka

### Streszczenie

Wielomiany należą do najważniejszych funkcji w informatyce i w obliczeniach komputerowych. Wystarczy przypomnieć, że wartość każdej standardowej funkcji w komputerze jest przybliżana przez wartość odpowiedniego wielomianu lub ilorazu wielomianów. Wynika to z prostego faktu, że w komputerze są wykonywane tylko podstawowe działania arytmetyczne i do takich działań muszą być sprowadzone wszystkie obliczenia, a wartość wielomianu można obliczyć za pomocą ciągu mnożeń i dodawań.

Schemat Hornera to bardzo prosty sposób obliczania wartości wielomianu. Jeśli wielomian jest stopnia  $n$ , to jego wartość można obliczyć, wykonując  $n$  dodawań i  $n$  mnożeń. Jak udowodnił A. Borodin, jest to algorytm optymalny w tym sensie, że szybciej nie da się tego zrobić. Mamy więc kapitalną sytuację w obliczeniach komputerowych, posługując się wielomianami – wszystkie te działania są wykonywane możliwie najszybciej.

Schemat Hornera ma bardzo wiele zastosowań poza obliczaniem wartości wielomianu, zwłaszcza w interpretacji pozycyjnego systemu reprezentowania liczb, w szczególności w systemie binarnym. Jest o tym mowa w jednostce tematycznej 5.9 poświęconej reprezentacji binarnej liczb oraz w jednostce poświęconej szybkiemu potęgowaniu 6.7.

## Przygotowanie uczniów

Programują w języku Python.

## Cele szczegółowe zajęć – osiągnięcia uczniów

### Uczeń:

- przekształca wielomian do postaci Hornera,
- oblicza wartość wielomianu, posługując się schematem Hornera,
- zapisuje schemat Hornera w postaci algorytmu,
- programuje schemat Hornera w Pythonie,
- uruchamia swój program w Pythonie i testuje na różnych danych,
- oblicza, ile działań jest wykonywanych w schemacie Hornera.

## Treści programowe – wypisy z podstawy programowej informatyki

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

2. zapisuje za pomocą listy kroków, schematu blokowego lub pseudokodu, i implementuje w wybranym języku programowania, algorytmy poznane na wcześniejszych etapach oraz algorytmy:
  - h. obliczania wartości wielomianu za pomocą schematu Hornera,

## Pojęcia i metody informatyczne

- wielomian,
- obliczanie wartości wielomianu,
- obliczenia matematyczne,
- schemat Hornera.

## Metody pracy w klasie

- **Słowna i Oglądowa:** Nauczyciel przypomina wielomiany niższego stopnia, następnie poszerza postać na wielomian dowolnego stopnia. Zachęca uczniów do dyskusji, w jaki sposób obliczaliby wartość wielomianu, np. stopnia 2 i 3. Sugeruje w pewnym momencie przekształcenie wielomianu do postaci, z której uczniowie wyprowadzają pełną postać Hornera.

Przed tymi rozważaniami nauczyciel może poprosić o przypomnienie, w jaki sposób oblicza się dziesiętną wartość liczby danej w postaci binarnej (jest temu poświęcona jednostka tematyczna 5.9).
- **Czynna:** Uczniowie indywidualnie obliczają wartości przykładowych wielomianów, piszą algorytm, który następnie programują, uruchamiają i testują.
- **Aktywizująca:** Nauczyciel zachęca uczniów do policzenia, ile mnożeń i dodawań jest wykonywanych w schemacie Hornera.

## Formy pracy uczniów

- praca indywidualna – najpierw na papierze uczniowie obliczają wartości wielomianów stopnia 2 i 3, następnie tworzą zwarty zapis schematu Hornera i na końcu programują go w Pythonie.

## Przygotowanie do zajęć – materiały, urządzenia, oprogramowanie

Jako przygotowanie do zajęć nauczyciel może utworzyć planszę z obliczaniem dziesiętnej wartości liczby binarnej za pomocą schematu Hornera – uczniowie nie muszą wiedzieć na początku, co to za metoda.

## Uwagi metodyczne dla nauczyciela dotyczące realizacji przyjętych celów i zapisów podstawy programowej

**Wielomiany** występują w wielu szkolnych zadaniach z matematyki, np. jako **funkcja liniowa**  $y = ax + b$ , czyli wielomian pierwszego stopnia, lub **funkcja kwadratowa**  $y = ax^2 + bx + c$ , czyli wielomian drugiego stopnia. **Stopień wielomianu** to najwyższa potęga, w jakiej występuje  $x$ . Pojawiają się także na zajęciach z fizyki i chemii. Często obliczane są ich wartości dla konkretnych argumentów. Ponieważ są to bardzo proste funkcje, bez zastanowienia wykonujemy zapisane w nich działania, ale czy rzeczywiście wykonujemy możliwie najmniej działań? Rozważmy wielomian drugiego stopnia:

$$w(x) = ax^2 + bx + c.$$

Aby obliczyć jego wartość, zwykle podnosimy najpierw  $x$  do kwadratu (czyli mnożymy  $x$  przez siebie), a następnie wykonujemy mnożenia przez współczynniki  $a$  i  $b$  oraz dwa razy dodajemy. W sumie wykonujemy w ten sposób **trzy** razy mnożenie i **dwa** razy dodawanie.

Na zajęciach z matematyki uczniowie poznają operację wyłączania przed nawias wspólnego czynnika. W wielomianie drugiego stopnia na pierwszy rzut oka nie widać, by można było wykonać takie przekształcenie. Zauważamy jednak, że można zgrupować dwa pierwsze wyrazy i wyłączyć z nich  $x$ , jako wspólny czynnik, stąd otrzymujemy:

$$w(x) = (ax + b)x + c.$$

A zatem wartość wielomianu stopnia 2 można obliczyć, wykonując 2 mnożenia i 2 dodawania, czyli o jedno mnożenie mniej niż poprzednio. Zysk niewielki, ale staje się bardziej istotny, jeśli wartość tego wielomianu liczymy bardzo wiele razy, tysiące, miliony.

Stosując tę samą metodę grupowania wyrazów i wyłączania czynników, dla wielomianu stopnia 3 otrzymujemy najpierw:

$$v(x) = ax^3 + bx^2 + cx + d = (ax^2 + bx + c)x + d.$$

a następnie wyłączamy:

$$v(x) = ax^3 + bx^2 + cx + d = ((ax + b)x + c)x + d.$$

W ten sposób zamiast pięciu mnożeń i trzech dodawań wystarczy wykonać trzy mnożenia i trzy dodawania. Zauważmy, że w obu przypadkach wykonujemy tyle samo mnożeń i dodawań, ile wynosi stopień wielomianu.

W praktyce obliczeniowej występują wielomiany wyższych niż 3 stopni, wielomiany wyższych stopni o szczególnych współczynnikach pojawiają się także w innych zadaniach szkolnych, np. związanych z pozycyjnym systemem reprezentacji liczb (patrz 5.9). Postarajmy się więc uogólnić powyższe przekształcenie wielomianu na wielomian dowolnego stopnia  $n$ . Przyjmijmy, że taki wielomian ma postać:

$$w_n(x) = a_{0x}^n + a_{1x}^{n-1} + a_{2x}^{n-2} + \dots + a_{n-2x}^2 + a_{n-1x} + a_n$$

Zanim przekształcimy ten wielomian, można poprosić uczniów, by wyznaczyli, ile mnożeń i dodawań trzeba wykonać, by obliczyć jego wartość, przeprowadzając te działania od prawej do lewej.

Podobnie jak w przypadku wielomianów stopnia 2 o 3, rozpoczynamy od wyłączenia  $x$  ze wszystkich wyrazów, w których występuje, czyli z wyjątkiem ostatniego – otrzymujemy:

$$w_n(x) = (a_{0x}^{n-1} + a_{1x}^{n-2} + a_{2x}^{n-3} + \dots + a_{n-2x} + a_{n-1x})x + a_n$$

W następnym kroku wyłączamy  $x$  ze wszystkich wyrazów w nawiasie z wyjątkiem tego najbardziej na prawo i otrzymujemy:

$$w_n(x) = ((a_{0x}^{n-2} + a_{1x}^{n-3} + a_{2x}^{n-4} + \dots + a_{n-2})x + a_{n-1x})x + a_n$$

Kontynuując to postępowanie dla wielomianów w zagłębiających się nawiasach, uzyskujemy następującą postać wielomianu:

$$w_n(x) = ((\dots ((a_{0x} + a_1)x + a_2)x + \dots + a_{n-2})x + a_{n-1x})x + a_n$$

Rozstawienie nawiasów w tym wzorze wyznacza następującą kolejność wykonywania działań, które prowadzimy od najgłębszej pary nawiasów:

przyjmij współczynnik przy najwyższej potędze za początkową wartość wielomianu;

powtarzaj aż do wyczerpania listy współczynników: bieżącą wartość wielomianu pomnóż przez  $x$  i dodaj kolejny współczynnik wielomianu.

Przypuśćmy, że chcemy obliczyć  $w_n(z)$ , czyli wartość tego wielomianu dla wartości argumentu  $x = z$  – oznaczmy tę wartość przez  $y$ . Wtedy obliczenia przebiegające zgodnie z podanym wzorem można zapisać w postaci następującego algorytmu:

$$y = a_0$$

$$y = yz + a_i \text{ dla } i = 1, 2, \dots, n$$

$$w_n(z) = y$$

Postać wielomianu z rozstawionymi nawiasami i wynikający z niej powyższy sposób obliczania wartości wielomianu nazywa się **schematem Hornera**. Widać, że w tym algorytmie obliczenia wartości wielomianu stopnia  $n$ , jest wykonywanych  $n$  mnożeń i  $n$  dodawań.

Zapis schematu Hornera w języku Python jest równie prosty:

```
def Horner(n, lista, x):
    y=lista[0]
    for i in range(1, n, 1):
        y=y*x+lista[i]
    return (y)
```

Wszystkie współczynniki wielomianu powinny być wcześniej zapisane w liście `lista`, nawet jeśli niektóre z nich są równe zeru. To jest koszt, jaki czasem musimy ponosić w automatycznych obliczeniach. Jeśli mielibyśmy uwzględnić specjalne wartości współczynników, należałoby to zrobić w algorytmie – wtedy zburzylibyśmy prostotę i zniszczyli elegancję schematu Hornera.

Jako wyzwanie dla uczniów można im dać następujące zadanie. Masz do dyspozycji jedynie kalkulator. Oblicz dziesiętną wartość liczby ósemkowej 73051, posługując się schematem Hornera, przy tym postaraj się nie używać komórki pamięci kalkulatora.

To zadanie ilustruje, jak szybko można obliczać wartość dziesiętną liczb binarnych, a ogólnie – jak szybko oblicza się wartość wielomianu za pomocą prostego kalkulatora.

## Przebieg zajęć

1. *Dyskusja w całej klasie.* Na początku nauczyciel prosi uczniów o przypomnienie postaci wielomianów pierwszego i drugiego stopnia. Prosi, by przez analogię ktoś napisał postać wielomianu trzeciego stopnia.
2. *Indywidualna praca uczniów.* Nauczyciel prosi uczniów, by zaproponowali sposoby obliczania wartości wielomianów 1, 2 i 3 stopnia, w których jest wykonywana możliwie najmniejsza liczba dodawań i mnożeń. Czy znajdzie się uczeń, który wykona tyle samo dodawań i tyle samo mnożeń, ile wynosi stopień wielomianu?
3. *Ponownie wspólnie.* Jeśli znalazł się taki uczeń, to czy potrafi rozszerzyć swoją metodę na wielomian stopnia 4? Ostatecznie, nauczyciel z pomocą uczniów wypisuje na tablicy schemat Hornera dla wielomianów stopni 2, 3 i 4. Czy z tych rozważań wynika, jaka będzie ogólna postać schematu Hornera dla obliczania wartości wielomianu dowolnego stopnia?  
Efektem wspólnych rozważań jest postać schematu Hornera dla wielomianu dowolnego stopnia.
4. *Indywidualna praca uczniów.* Każdy z uczniów zapisuje schemat Hornera w języku Python, uruchamia program i testuje go na kilku wybranych wielomianach. Wśród danych testowych powinny się znaleźć wielomiany, w których niektóre współczynniki są równe 0. Schemat Hornera powinien działać poprawnie również dla takich wielomianów.
5. *Indywidualna praca uczniów i ich ocena.* W przypadku zadań programistycznych nauczyciel motywuje uczniów, wyżej oceniając ich wkład, jeśli program został utworzony samodzielnie (przez ucznia lub w parze z innym uczniem), nie korzystając z gotowych rozwiązań, których obecnie jest bardzo wiele dostępnych w sieci.
6. *Dyskusja podsumowująca.* Ma na celu m.in. obliczenie, ile działań elementarnych jest wykonywanych przez schemat Hornera – powinno być  $n$  mnożeń i  $n$  dodawań dla wielomianu stopnia  $n$ .
7. *Indywidualna praca uczniów.* Każdy z uczniów na koniec zajęć sprawdza, czy jego program wykonuje również dokładnie  $n$  mnożeń i  $n$  dodawań dla wielomianu stopnia  $n$ .

## Modyfikacja, poszerzenie zakresu, kontynuacja zajęć

Schemat Hornera jest stosowany w wielu innych algorytmach, które pojawiają się na informatyce szkolnej, np. w obliczeniach dziesiętnej wartości liczby binarnej lub liczby danej w innym systemie pozycyjnym (patrz jednostka tematyczna 5.9), a także w algorytmach szybkiego potęgowania (patrz jednostka tematyczna 6.7).

## 6.5 Strategia dziel i zwyciężaj w porządkowaniu

### Temat jednostki:

#### Strategia dziel i zwyciężaj w porządkowaniu

#### Poziom kształcenia

Szkoła ponadpodstawowa, informatyka – zakres rozszerzony

#### Streszczenie

Porządkowanie informacji, w szczególności liczb, zwane też sortowaniem, jest jedną z najważniejszych i najczęściej wykonywanych operacji w komputerze. Algorytmy porządkowania należą do klasyki informatyki i kształcenia informatycznego na każdym poziomie edukacji. W tym zbiorze propozycji materiałów szkoleniowych w jednostce p. 5.7 uczniowie poznają metodę porządkowania przez wybór, która jest naturalnym, spiralnym rozszerzeniem metody wyszukiwania liniowego.

Algorytmy porządkowania są przykładami zastosowania różnych technik algorytmicznych – przegląd tych algorytmów i zastosowanych technik jest tematem jednostki 6.6. Jedną z najczęściej stosowanych technik jest **metoda dziel i zwyciężaj**, która polega na dekompozycji problemu i użyciu rekurencji. W tej jednostce zajmujemy się **porządkowaniem przez scalanie**, które odbywa się w myśl zasady: aby uporządkować ciąg liczb, podziel ten ciąg na dwa w miarę równoliczne podciągi, uporządkuj je tą samą metodą i uporządkowane podciągi scal w jeden ciąg uporządkowany. Jest to jeden z najbardziej efektywnych algorytmów porządkowania.

Jednostka ta jest jednocześnie ilustracją zastosowania dekompozycji i rekurencji jako sposobów rozumowania składających się na myślenie komputacyjne.

#### Przygotowanie uczniów

Znajomość przynajmniej jednej metody porządkowania – porządkowanie przez wybór, patrz jednostka 5.7. Umiejętność programowania, w tym użycia rekurencji, patrz jednostka 5.6, gdzie użyto rekurencji w algorytmie rysowania.

#### Cele szczegółowe zajęć – osiągnięcia uczniów

##### Uczeń:

- stosuje dekompozycję problemu na mniejsze,
- scala dwa ciągi uporządkowane w jeden ciąg uporządkowany,
- stosuje rekurencję w porządkowaniu,
- porządkuje ciąg, stosując metodę przez scalanie,
- programuje funkcję i stosuje ją w programach rekurencyjnych,
- uruchamia swoje programy w Pythonie i ewentualnie poprawia je, gdy działają niezgodnie ze specyfikacją, testuje ich działanie na różnych danych.

#### Treści programowe – wypisy z podstawy programowej informatyki

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

1. zapisuje za pomocą listy kroków, schematu blokowego lub pseudokodu i implementuje w wybranym języku programowania, algorytmy poznane na wcześniejszych etapach oraz algorytmy:
  - e. sortowania ciągu liczb przez scalanie.

#### Pojęcia i metody informatyczne

- podejście algorytmiczne dziel i zwyciężaj,
- scalanie ciągów uporządkowanych w ciąg uporządkowany,
- porządkowanie ciągu przez scalanie,
- rekurencja,
- funkcja rekurencyjna,
- komputerowa realizacja porządkowania przez scalanie.



## Metody pracy w klasie

- **Słowna i Oglądowa:** dyskusja w klasie, moderowana przez nauczyciela, ma doprowadzić do ogólnego sformułowania algorytmu scalania, a później sortowania przez scalanie.
- **Czynna:** uczniowie najpierw dyskutują nad rozwiązaniem postawionego problemu, później tworzą algorytm i testują go odręcznie na przykładach, na końcu piszą odpowiedni program w Pythonie.
- **Aktywizująca:** uczniowie mogą pracować (programować) w parach, dzieląc zadania do wykonania między sobą, aktywizując się nawzajem.

## Formy pracy uczniów

- zajęcia w grupie (cała klasa) i w podgrupach, poza komputerami, dyskusje nad postawionymi przez nauczyciela pytaniami,
- praca indywidualna: odręczne wykonywanie algorytmu na przykładach,
- praca indywidualna nad stworzeniem algorytmu, a następnie zaprogramowaniem go,
- ewentualnie, programowanie w parach.

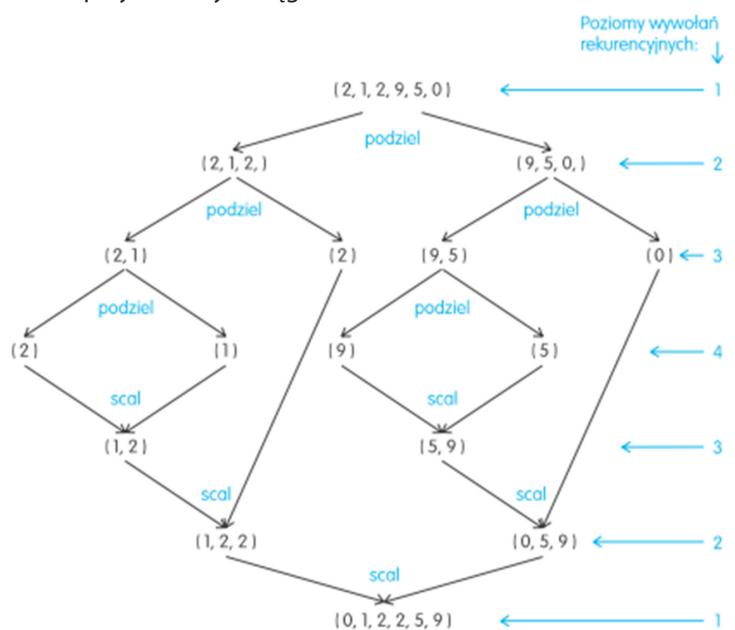
## Przygotowanie do zajęć – materiały, urządzenia, oprogramowanie

Do przybliżenia uczniom algorytmów scalania i porządkowania przez scalanie przydatna może być talia kart do gry (z naturalnym porządkiem kart kolorami i figurami) lub zestaw kart z liczbami. Przytoczony w następnym punkcie opis algorytmów uczniowie mogą w pierwszej kolejności zastosować właśnie do zestawów kart, by lepiej poznać te algorytmy. Przed przystąpieniem do ich programowania.

Szczegółowy opis porządkowania przez scalanie jest zawarty w punkcie 10.2 w książce M.M. Sysło *Algorytmy* [4], jak również w [5].

## Uwagi metodyczne dla nauczyciela dotyczące realizacji przyjętych celów i zapisów podstawy programowej

Jednostka ta jest poświęcona przybliżeniu uczniom algorytmu porządkowania przez scalanie. W ogólnym zarysie, w tym sposobie porządkowania, najpierw ciąg, który ma być uporządkowany, jest dzielony (**krok dekompozycji, podziału**) na dwa w miarę równoliczne podciągi, następnie te dwa podciągi są porządkowane (**rekurencyjnie**) ... tą samą metodą, i już uporządkowane są **scalane** w jeden ciąg uporządkowany. Poniżej ta metoda jest zilustrowana na przykładowym ciągu:



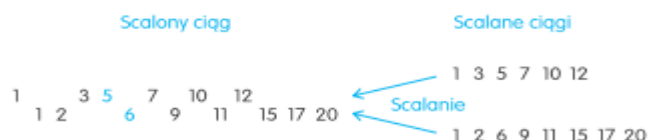
Z tego ogólnego opisu wynika, że do utworzenia algorytmu porządkowania przez scalanie należy wiedzieć:

- a) jak dzielić porządkowany ciąg na dwa podciągi,
- b) jak scalać dwa uporządkowane ciągi w jeden ciąg uporządkowany,

- c) jak wywoływać zagłębiające się etapy tego procesu podziału i scalania,
- d) kiedy kończą się wywołania rekurencyjne.

Odpowiedzi na te wszystkie punkty, z wyjątkiem b), są dość oczywiste: a) – ciąg jest dzielony na prawie połowy, c) – ten sam program jest wywoływany dla podciągów, d) – wywołania dla podciągu kończą się, gdy nie można go podzielić na dwa podciągi, czyli gdy składa się z jednego elementu. Te odpowiedzi są zilustrowane na poprzednim rysunku.

Pozostaje do wyjaśnienia, jak scalić dwa uporządkowane podciągi w jeden ciąg uporządkowany. Można to wykonać, przeglądając oba podciągi od najmniejszych elementów i przenosząc do tworzonego ciągu ten z widocznych elementów – patrz ilustracja poniżej:



Realizacja przedstawionych algorytmów w języku Python jest zamieszczona w punkcie 10.2 w książce [4].

## Przebieg zajęć

1. *Dyskusja w całej klasie.* Na początku nauczyciel prosi uczniów o przypomnienie, na czym polega porządkowanie informacji lub liczb. Można się odwołać również do porządku leksykograficznego, takiego jak w słownikach.  
W prezentacji algorytmów na tych zajęciach bardzo pomocna może być talia kart (do gry lub opatrzonych liczbami), każdy uczeń powinien mieć szansę posłużyć się nią, szukając wyjaśnienia, jak działają omawiane tutaj algorytmy.  
Nauczyciel najpierw opisuje algorytm scalania dwóch ciągów uporządkowanych i wykonuje go na dwóch stertach uporządkowanych kart. Dobrze jest pokazać, że przebieg algorytmu nie zależy od liczby kart w stertach.
2. *Indywidualna praca uczniów.* Wykonują algorytm scalania dwóch uporządkowanych ciągów liczb, na stertach kart, ale także w zeszytach – patrz schemat w poprzedniej sekcji.
3. *Ponownie wspólnie.* Nauczyciel stawia pytanie: ile porównań należy wykonać, aby scalić dwa uporządkowane ciągi odpowiednio o  $k$  i  $l$  elementach? Uczniowie liczą, posługując się swoimi przykładami.  
*Podpowiedź.* W najgorszym przypadku jest to  $k + l - 1$ , i ta liczba nie zależy od  $k$  i  $l$ , ale od tego, jakie są porządkowane liczby.
4. *Ponownie wspólnie.* Teraz nauczyciel wyjaśnia, w jaki sposób zastosować metodę dziel i zwyciężaj, aby uporządkować dowolny ciąg. Posługuje się w tym celu schematem wywołań rekurencyjnych, zamieszczonym w poprzedniej sekcji.
5. *Indywidualna praca uczniów.* Teraz uczniowie tworzą podobny schemat porządkowania przez scalanie dla swojego ciągu.
6. *Indywidualna praca uczniów.* Ostatecznie uczniowie są przygotowani, aby zapisać w Pythonie pełny algorytm porządkowania przez scalanie. Testują swój program na swoich i innych danych i ewentualnie debugują go.
7. *Indywidualna praca uczniów i ich ocena.* W przypadku zadań programistycznych nauczyciel motywuje uczniów, wyżej oceniając ich wkład, jeśli program został utworzony samodzielnie (przez ucznia lub w parze z innym uczniem), nie korzystając z gotowych rozwiązań, których obecnie jest bardzo wiele dostępnych w sieci.
8. *Dyskusja podsumowująca.* Jednym z wątków dyskusji powinna być rekurencja, jej działanie i korzyści z zapisu rekurencyjnego (bardzo zwarte) – patrz rozdz. 5 w książce [3], zatytułowany „Rekurencja – jak korzystać z tego, co już znamy lub jak „zrzucić robotę” na komputer”. Patrz również jednostka 6.9 w tych materiałach oraz cały rozdział w książce [5].

## Modyfikacja, poszerzenie zakresu, kontynuacja zajęć

Kontynuacja zajęć poszerzających zakres porządkowania przez scalanie może polegać na policzeniu, ile porównań jest wykonywanych w całym tym algorytmie, patrz p. 10.2 w [4]. Ponadto tę metodę porządkowania można porównać z innymi algorytmami porządkowania (patrz jednostka 6.6 w tych materiałach), a kontynuacją rozważań o rekurencji jest jednostka 6.9.

## 6.6 Algorytmy wyszukiwania i porządkowania

### Temat jednostki:

#### Algorytmy wyszukiwania i porządkowania

### Poziom kształcenia

Szkoła ponadpodstawowa, informatyka – zakres podstawowy i rozszerzony

### Streszczenie

Zajęcia mają na celu podsumowanie tematyki wyszukiwania i porządkowania informacji, w szczególności liczb. Główną uwagę powinna być skupiona na różnorodności sytuacji problemowych, w których niezbędne jest posłużenie się odpowiednim algorytmem wyszukiwania lub porządkowania, jego komputerową implementacją. Jak ilustrują treści programowe wypisane z podstawy programowej dla szkół ponadpodstawowych, tematy wyszukiwania i porządkowania zajmują znaczną część zajęć kształcenia informatycznego w szkołach podstawowych (jednostki 5.7, 5.8) i ponadpodstawowych (jednostka 6.2, 6.5).

Zajęcia powinny być poprowadzone zarówno z uczniami, którzy wybrali rozszerzenie informatyczne, jak i z pozostałymi, których jest zapewne znacznie więcej w każdej szkole. Zakres tych zajęć powinien więc być dostosowany do poziomu zagadnień, przewidzianego w podstawie programowej.

Nauczyciel powinien wcześniej poinformować uczniów o zakresie tych zajęć, by mogli się do nich dobrze przygotować.

### Przygotowanie uczniów

Uczniowie powinni być przygotowani w zakresie zagadnień wyszukiwania i porządkowania, które są przedmiotem kształcenia informatycznego w szkołach podstawowych i ponadpodstawowych. W szczególności wcześniej poznali:

- wyszukiwanie elementów z ciągu nieuporządkowanym i uporządkowanym,
- znajdowanie elementów o ustalonych własnościach, np. najmniejszych i największych,
- wyszukiwanie źródeł informacji na zadany temat,
- porządkowanie ciągów liczb.

### Cele szczegółowe zajęć – osiągnięcia uczniów

Uczeń porządkuje, pogłębia i poszerza, utrwała swoje osiągnięcia (kursywą oznaczono zagadnienia, które nie są przedmiotem rozważań w tych materiałach, ale występują w podstawie programowej informatyki):

- znajduje najmniejszy lub największy element w zbiorze/ciągu,
- wyszukuje elementy w zbiorze nieuporządkowanym,
- stosuje metodę połowienia do wyszukiwania elementów zbiorze uporządkowanym,
- porządkuje ciąg, stosując metodę przez wybór,
- porządkuje ciąg metodą bąbelkową,
- porządkuje ciąg metodą przez wstawianie,
- porządkuje zbiór przez zliczanie,
- scala dwa ciągi uporządkowane w jeden ciąg uporządkowany,
- porządkuje ciąg metodą przez scalanie,
- porządkuje ciąg metodą szybką,
- stosuje rekurencję w porządkowaniu,
- wyjaśnia ideę metody dziel i zwyciężaj w projektowaniu algorytmów wyszukiwania i porządkowania

### Treści programowe – wypisy z podstawy programowej informatyki

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

3. zapisuje za pomocą listy kroków, schematu blokowego lub pseudokodu i implementuje w wybranym języku programowania algorytmy poznane na wcześniejszych etapach oraz algorytmy:

- a. [...],
  - b. znajdowania określonego elementu w zbiorze: lidera, idola, elementu w zbiorze uporządkowanym metodą binarnego wyszukiwania,
  - c. [...],
  - d. jednoczesnego wyszukiwania elementu najmniejszego i największego,
  - e. sortowania ciągu liczb przez scalanie,
  - f. [...],
  - g. [...],
  - h. [...],
  - i. [...],
  - j. [...],
  - k. [...];
4. wykorzystuje znane sobie algorytmy przy rozwiązywaniu i programowaniu rozwiązań następujących problemów:
- a. [...],
  - b. [...],
  - c. [...],
  - d. [...],
  - e. [...],
  - f. [...];
5. objaśnia, a także porównuje podstawowe metody i techniki algorytmiczne oraz struktury danych, wykorzystując przy tym przykłady problemów i algorytmów, w szczególności:
- a. wyszukiwanie elementów liniowe i przez połowienie (do znajdowania elementów w zbiorze, sortowania przez wstawianie, przybliżonego rozwiązywania równań, sprawdzania przynależności punktu do wielokąta wypukłego),
  - b. [...],
  - c. metodę dziel i zwyciężaj (jednoczesne znajdowanie minimum i maksimum, sortowanie przez scalanie i szybkie),
  - d. podejście zachłanne ([...], szukania najkrótszej drogi),
  - e. programowanie dynamiczne ([...], szukania najdłuższego wspólnego podciągu),
  - f. [...],
  - g. metodę haszowania (wyszukiwanie wzorca w tekście),
  - h. [...],
  - i. struktury dynamiczne: stos, kolejka, lista (do realizacji algorytmu: ONP, symulacji problemu Flawiusza, sortowania leksykograficznego),
  - j. [...].

### Pojęcia i metody informatyczne

- porządek: liczb, słów, tekstów,
- wyszukiwanie: liniowe, przez połowienie, interpolacyjne,
- porządkowanie: kubelkowe, bąbelkowe, przez zliczanie, przez wybór, przez wstawianie, przez scalanie, szybkie,
- scalanie,
- strategia dziel i zwyciężaj,
- rekurencja.

### Metody pracy w klasie

- **Słowna i Oglądowa:** dyskusja w klasie moderowana przez nauczyciela na temat metod i algorytmów wyszukiwania i porządkowania.
- **Czynna:** zajęcia mają głównie formę czynną, nauczyciel ewentualnie koryguje wypowiedzi lub oceny uczniów; zajęcia w żadnym zakresie nie powinny mieć formy wykładu.

- **Aktywizująca:** nauczyciel stara się aktywizować uczniów pytaniami, które mają na celu spojrzenie na omawianą tematykę „z lotu ptaka”, z góry, ponad szczegółami związanymi z konkretnymi zagadnieniami, metodami i algorytmami.

### Formy pracy uczniów

- zajęcia głównie w grupie (cała klasa) i w podgrupach, poza komputerami, dyskusje nad postawionymi przez nauczyciela pytaniami i zadaniami do wykonania.
- praca w niewielkim stopniu indywidualna, raczej nie związana z programowaniem, ewentualnie skorzystanie z wcześniej wykonanych programów,

### Przygotowanie do zajęć – materiały, urządzenia, oprogramowanie

Nauczyciel powinien przygotować wykaz tych zagadnień programowych z podstawy programowej, które odnoszą się do zagadnień wyszukiwania i porządkowania.

Według własnego pomysłu powinien przygotować tabele, które na zajęciach posłużyłyby uczniom do utworzenia informacji zbiorczych, np. dotyczących użytych technik algorytmicznych i efektywności algorytmów, a także złożoności obliczeniowej (w sensie liczby wykonywanych operacji).

### Uwagi metodyczne dla nauczyciela dotyczące realizacji przyjętych celów i zapisów podstawy programowej

Ważne, by takie zajęcia były osobno przeprowadzone dla uczniów, którzy obrali informatykę w zakresie podstawowym, jak i dla tych, którzy wybrali rozszerzenie informatyczne. Dla tej pierwszej grupy mogą mieć większe znaczenie niż dla tej drugiej.

Spojrzenie i ukierunkowanie dyskusji przez nauczyciela powinno być „ponad” konkretnymi problemami, metodami i algorytmami. Uczniowie powinni nie tylko posługiwać się konkretnym algorytmem, ale także umieć go scharakteryzować, umieścić wśród innych metod rozwiązywania pokrewnych zagadnień podobnymi metodami, ocenić go na tle innych metod. Dostrzec także możliwości modyfikacji.

### Przebieg zajęć

Przebieg zajęć scharakteryzowano już wcześniej. Głównie są to zajęcia dyskusyjne, poza komputerem, dobrze przygotowane przez nauczyciela, w których biorą udział uczniowie również dobrze przygotowani, wcześniej poinformowani przez nauczyciela o zakresie zajęć.

Zajęcia te mogą również służyć wyjaśnieniu kwestii, które pojawią się u uczniów w trakcie przygotowywania się.

Dla uporządkowania dyskusji nauczyciel powinien rozdać uczniom wcześniej przygotowane tabele, w których umieści zagadnienie do dyskusji i pozostawi miejsce na wpisy uczniów.

Jedna z takich tabel powinna być poświęcona wszystkim algorytmom porządkowania. Poszczególne kolumny powinny zawierać: nazwę algorytmu (umieszcza nauczyciel), użytą technikę algorytmiczną, złożoność obliczeniową. W tej ostatniej kolumnie należy wyróżnić podkolumny odpowiadające różnym porządkowanym ciągom: losowym, uporządkowanym i odwrotnie uporządkowanym.

Jako zadanie ekstra uczniowie mogą utworzyć jeden master program, w którym umieszczą wszystkie swoje programy do porządkowania ciągu liczb, by przetestować ich praktyczną efektywność na tych samych danych. Ten master program powinien testować czas działania algorytmów na trzech rodzajach ciągów: losowych, uporządkowanych i odwrotnie uporządkowanych. Jako rezultat powinna być drukowana tabela z wszystkimi czasami obliczeń.

### Modyfikacja, poszerzenie zakresu, kontynuacja zajęć

Tego typu zajęcia powinny zostać zorganizowane przez nauczyciela na wiele różnych tematów, które wymagają od ucznia uporządkowania, pogłębienia i poszerzenia, utrwalenia swojej wiedzy, umiejętności i osiągnięć. Inna jednostka 6.9 jest poświęcona myśleniu rekurencyjnemu. Inne ważne zajęcia zbiorcze powinny być poświęcone myśleniu komputacyjnemu.

## 6.7 Szybkie potęgowanie

### Temat jednostki:

#### Szybkie potęgowanie

Jednostka na 2–3 godziny lekcyjne

### Poziom kształcenia

Szkoła ponadpodstawowa, informatyka – zakres rozszerzony

### Streszczenie

Potęgowanie, czyli podnoszenie liczb naturalnych do naturalnej potęgi, jest ważną operacją we współczesnej kryptografii. Podnoszone są liczby kilkusetcyfrowe do potęg, które również mają kilkaset cyfr. Takiej operacji nie można wykonać szkolnym algorytmem, czyli przez kolejne mnożenia, gdyż trwałoby to w nieskończoność, a szyfrowaną pocztę chcielibyśmy otrzymywać natychmiast. Ta jednostka jest poświęcona algorytmom szybkiego potęgowania. Uczniowie łatwo mogą się przekonać, że potęgowanie przez mnożenie jest bardzo wolnym algorytmem, natomiast algorytm szybkiego potęgowania można bardzo łatwo wyprowadzić uwzględniając parzystość wykładnika (algorytm rekurencyjny) lub przedstawiając wykładnik w postaci binarnej (algorytmy liniowe). Oba te rodzaje algorytmów są szalenie szybkie!

### Przygotowanie uczniów

Uczniowie wcześniej powinni poznać:

- binarną reprezentację liczb, jednostka 5.9;
- schemat Hornera, jednostka 6.4;
- rekurencję w projektowaniu algorytmów i w programowaniu, np. jednostki 5.6 i 6.9;
- rekurencję w programowaniu, np. jednostki 5.6 i 6.9.

### Cele szczegółowe zajęć – osiągnięcia uczniów

#### Uczeń:

- stosuje dekompozycję problemu na mniejsze,
- stosuje rekurencję,
- programuje funkcję i stosuje ją w programach rekurencyjnych,
- uruchamia swoje programy w Pythonie i ewentualnie poprawia je, gdy działają niezgodnie ze specyfikacją, testuje ich działanie na różnych danych.

### Treści programowe – wypisy z podstawy programowej informatyki

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

1. zapisuje za pomocą listy kroków, schematu blokowego lub pseudokodu, i implementuje w wybranym języku programowania algorytmy poznane na wcześniejszych etapach oraz algorytmy:
  - i. szybkiego potęgowania liczb w wersji iteracyjnej i rekurencyjnej

### Pojęcia i metody informatyczne

- interpretacja reprezentacji binarnej przy potęgowaniu,
- podejście algorytmiczne dziel i zwyciężaj,
- rekurencja,
- funkcja rekurencyjna,
- komputerowa realizacja szybkiego potęgowania, algorytm iteracyjny i rekurencyjny.

### Metody pracy w klasie

- **Słowna i Oglądowa:** dyskusja w klasie moderowana przez nauczyciela ma doprowadzić do ogólnego sformułowania algorytm potęgowania w wersji rekurencyjnej i iteracyjnej.

- **Czynna:** uczniowie najpierw dyskutują nad rozwiązaniem postawionego problemu, później tworzą algorytm i testują go odręcznie na przykładach, na końcu piszą odpowiednie programy w Pythonie.
- **Aktywizująca:** uczniowie mogą pracować (programować) w parach, dzielą zadania do wykonania między sobą, aktywizują się nawzajem.

### Formy pracy uczniów

- zajęcia w grupie (cała klasa) i w podgrupach, poza komputerami, dyskusje nad postawionymi przez nauczyciela pytaniami,
- praca indywidualna: odręczne wykonywanie algorytmu na przykładach,
- praca indywidualna nad stworzeniem algorytmu, a następnie zaprogramowaniem go,
- ewentualnie, programowanie w parach.

### Przygotowanie do zajęć – materiały, urządzenia, oprogramowanie

Polecamy wyprowadzenie algorytmów szybkiego potęgowania z wykorzystaniem rekurencji (punkt 8.1.2) i iteracji oraz schematu Hornera (punkt 7.3), oba odniesienia do książki [4]. Główne kroki powtarzamy poniżej w uwagach metodycznych.

### Uwagi metodyczne dla nauczyciela dotyczące realizacji przyjętych celów i zapisów podstawy programowej

W przebiegu zajęć podajemy drogę do otrzymania rekurencyjnej zależności, której realizacja prowadzi do rekurencyjnego programu obliczania wartości potęgi dla dowolnego wykładnika  $n$ .

$$x^n = \begin{cases} 1 & \text{dla } n = 0 \\ (x^{n/2})^2 & \text{dla parzystego } n \\ (x^{n-1})x & \text{dla nieparzystego } n \end{cases}$$

Dla określenia, ile mnożeń wykonuje rekurencyjny algorytm potęgowania, korzysta się, być może niespodziewanie ... z binarnego rozwinięcia wykładnika przedstawionego w postaci schematu Hornera. Dla przykładowego  $n = 22$  mamy reprezentację binarną i jej postać z wykorzystaniem schematu Hornera:

$$22 = (10110)_2 = 2^4 + 2^2 + 2^1 = (2^3 + 2 + 1)2 = ((2^2 + 1)2 + 1)2 = ((2 \bullet 2 + 1)2 + 1)2$$

Obliczmy teraz  $x^{22}$ , korzystając z tej postaci wykładnika:

$$x^{22} = x^{((2 \bullet 2 + 1)2 + 1)2} = (x^{(2 \bullet 2 + 1)2 + 1})^2 = ((x^{(2 \bullet 2 + 1)2})x)^2 = (((x^{2 \bullet 2})x)^2)x^2 = (((((x^2)x)^2)x)^2)x)^2$$

Otrzymaliśmy identyczną postać jak na zajęciach opisanych poniżej. Można to łatwo wytłumaczyć – potęgi 2 odpowiadają dzieleniu przez dwa podczas otrzymywania binarnej reprezentacji wykładnika, a mnożenie przez  $x$  odpowiada reszcie 1 przy tym dzieleniu. Widać to wyraźnie po podpisaniu pod sobą schematu Hornera reprezentacji binarnej wykładnika i sposobu obliczania potęgi.

$$22 = (10110)_2 ((2 \bullet 2 + 1)2 + 1)2$$

$$x^{22} = (((((x^2)x)^2)x)^2)x)^2$$

Aby obliczyć, ile jest wykonywanych mnożeń, porównajmy na schemacie powyżej binarną reprezentację wykładnika z kolejnością wykonywania mnożeń poniżej. Patrząc od prawej do lewej, widać, że z wyjątkiem najbardziej znaczącej pozycji w reprezentacji binarnej wykładnika  $n$  każdemu bitowi o wartości 1 odpowiada mnożenie przez  $x$ , a każdej pozycji odpowiada podniesienie do kwadratu. W tym konkretnym przypadku jest to  $(4 - 1) + (5 - 1) = 3 + 4 = 7$ . W ogólnym przypadku liczba mnożeń potrzebnych do obliczenia wartości  $x^n$  za pomocą algorytmu rekurencyjnego jest równa liczbie pozycji w binarnej reprezentacji liczby  $n$  minus 1 plus liczba bitów równych 1 w tej reprezentacji także minus 1. Ponieważ, jak wiadomo, długość binarnej reprezentacji liczby  $n$  wynosi około  $\log_2 n$ , liczba mnożeń potrzebnych do obliczenia wartości  $x^n$  algorytmem rekurencyjnym wynosi około  $2\log_2 n$ .

Sprawdźmy, jaki to ma efekt, gdy  $n = 12345678912345678912345678912345$ . W tym przypadku,  $2\log_2 n < 250$ . Zatem zamiast czekać  $10^8$  lat, wynik potęgowania dla tego wykładnika możemy otrzymać, wykonując nie więcej niż 250 mnożeń! To szokujące osiągnięcie naszego algorytmu!

## Przebieg zajęć

1. *Dyskusja w całej klasie.* Na początku nauczyciel prosi uczniów o przypomnienie zasad potęgowania: iloczyn potęg o tych samych podstawach (dodawanie wykładników) i potęga potęgi (mnożenie wykładników). Nauczyciel wspomina, że w kryptografii podnosi się do potęg kilkusetcyfrowych liczby kilkusetcyfrowe.
2. *Indywidualna praca uczniów.* Ponieważ na matematyce nie wprowadza się żadnej szybkiej metody potęgowania i jest ono wykonywane przez mnożenie, np.  $x^5 = x \cdot x \cdot x \cdot x \cdot x$ , nauczyciel prosi uczniów o policzenie na kalkulatorze, ile lat będzie trwało obliczenie następującej potęgi przez wielokrotne mnożenie przez  $x$  na superkomputerze o szybkości działania  $10^{15}$  oper/sek. (wykładnik jest bardzo mały w porównaniu ze stosowanymi w kryptografii):

$$x^{12345678912345678912345678912345}$$

Podpowiedź: ok.  $10^8$  lat.

3. *Ponownie wspólnie.* Nauczyciel pyta: jak obliczyć  $x^5$  szybciej niż przez kolejne mnożenia:  $x^5 = x \cdot x \cdot x \cdot x \cdot x$ ? Powinna pojawić się odpowiedź:  $x^5 = x \cdot x \cdot x \cdot x = (x^4) \cdot x = ((x^2)^2) \cdot x$ . A zatem, zamiast 4 mnożeń trzeba wykonać 3 mnożenia (podnoszenie do kwadratu wykonuje się jako mnożenie).
4. *Ponownie wspólnie.* Nauczyciel stawia pytanie, jak szybko obliczyć potęgę  $x^{22}$ ?  
Burza mózgów uczniów: wreszcie pojawia się propozycja:  $x^{11 \cdot 2} = (x^{11})^2$ . Super! A jak obliczyć  $x^{11}$ ? Trzeba pozbyć się nieparzystej potęgi! Pojawia się propozycja:  $x^{10+1} = (x^{10}) \cdot x$  – Super. Zauważmy, skorzystano jedynie z wzorów, które są znane z lekcji matematyki.
5. *Indywidualna praca uczniów.* Z tymi podpowiedziami ze wspólnej pracy każdy uczeń pochyla się teraz nad własnym zeszytem, by naszkicowaną metodą redukcji wykładnika otrzymać sposób obliczenia  $x^{22}$ .

Podpowiedź: powinni otrzymać:

$$x^{22} = (x^{11})^2 = ((x^{10})x)^2 = (((x^5)^2)x)^2 = (((((x^4)x)^2)x)^2 = ((((((x^2)^2)x)^2)x)^2)$$

Ostateczne rozstawienie nawiasów dyktuje następującą kolejność mnożeń:

$$x \cdot x = x^2; x^2 \cdot x^2 = x^4; x^4 \cdot x = x^5; x^5 \cdot x^5 = x^{10}; x^{10} \cdot x = x^{11}; x^{11} \cdot x^{11} = x^{22};$$

Zatem wartość potęgi  $x^{22}$  może być obliczona przy użyciu 6 mnożeń zamiast 21 w szkolnym algorytmie potęgowania.

6. *Ponownie wspólnie.* Uczniowie wraz z nauczycielem uogólniają wykryty algorytm dla dowolnego wykładnika  $n$ . Wzór jest podany w poprzedniej sekcji.
7. *Indywidualna praca uczniów.* Ostatecznie uczniowie są przygotowani, aby zapisać w Pythonie pełny rekurencyjny algorytm podnoszenia do potęgi. Testują swój program na swoich i innych danych i ewentualnie debugują go.
8. *Indywidualna praca uczniów i ich ocena.* W przypadku zadań programistycznych nauczyciel motywuje uczniów, wyżej oceniając ich wkład, jeśli program został utworzony samodzielnie (przez ucznia lub w parze z innym uczniem), nie korzystając z gotowych rozwiązań, których obecnie jest bardzo wiele dostępnych w sieci.
9. *Dyskusja podsumowująca.* W końcowej dyskusji nauczyciel informuje uczniów, że dla 32-cyfrowego wykładnika, od którego zaczęły się zajęcia, algorytm rekurencyjny wykona poniżej 200 mnożeń. To może wywołać SZOK u uczniów – 250 mnożeń zamiast  $10^8$  lat! To świadczy o POTĘDZE ALGORYTMÓW. Ciekawa może być reakcja uczniów na te informacje, nauczyciel może podać uzasadnienie, zawarłem je w sekcji powyżej.

Powyższy fakt potwierdza słuszność wypowiedzi Ralfa Gomory'ego, kiedyś szefa naukowego firmy IBM:

Najlepszym sposobem przyspieszania pracy komputerów jest obarczanie ich mniejszą liczbą działań.

## Modyfikacja, poszerzenie zakresu, kontynuacja zajęć

Poszerzenie i kontynuacja lekcji może być związana z powiązaniem algorytmu rekurencyjnego z binarną reprezentacją wykładnika w postaci schematu Hornera. Wyjaśniamy to krótko w sekcji Uwag metodycznych powyżej. Ta interpretacja jest źródłem iteracyjnych metod równie szybkiego potęgowania – piszemy o tym w punkcie 7.3 w książce [4]. Jeden z tych algorytmów podnosi do potęgi w trakcie znajdowania binarnej reprezentacji wykładnika.



## 6.8 Algorytm Euklidesa

### Temat jednostki:

#### Algorytm Euklidesa

### Poziom kształcenia

Szkoła ponadpodstawowa, informatyka – zakres rozszerzony

### Streszczenie

Algorytm Euklidesa służy do znajdowania największego dzielnika  $NWD(m, n)$  dwóch liczb całkowitych  $m$  i  $n$  (będziemy zakładali, że te liczby są nieujemne). Algorytm ten należy do klasyki informatyki. Powstał blisko 2500 lat temu i przez długie lata pojawiał się w rozważaniach i w literaturze jako synonim pojęcia algorytm<sup>1</sup>. Po ponad 2000 lat przeżywa swoją kolejną młodość. Po pierwsze, okazuje się, że jest bardzo efektywnym algorytmem w sensie dzisiejszej teorii złożoności obliczeniowej – jego złożoność jest logarytmiczna. Po drugie, znalazł zastosowanie w RSA, w jednej z najpopularniejszych metod szyfrowania z kluczem publicznym.

Ta jednostka jest poświęcona rekurencyjnej realizacji algorytmu Euklidesa, realizacja iteracyjna była przedmiotem zajęć w szkole podstawowej (nie ma jej w tych materiałach).

### Przygotowanie uczniów

Uczniowie wcześniej poznali:

- rozkładanie liczb na czynniki (na lekcjach matematyki i informatyki),
- obliczanie  $NWD(m, n)$  przez rozkład obu liczb na czynniki<sup>2</sup> – taką metodę znajdowania  $NWD(m, n)$  uczniowie powinni byli poznać na lekcjach matematyki w szkole podstawowej,
- obliczanie  $NWD(m, n)$  metodą iteracyjną (w szkole podstawowej),
- rekurencję w projektowaniu algorytmów i w programowaniu, np. jednostki 5.6 i 6.9,
- rekurencję w programowaniu, jednostki 5.6 i 6.9.

### Cele szczegółowe zajęć – osiągnięcia uczniów

#### Uczeń:

- oblicza  $NWD(m, n)$  rozkładając obie liczby na czynniki,
- oblicza  $NWD(m, n)$  stosując algorytm Euklidesa w wersji iteracyjnej,
- stosuje redukcję w rozwiązywaniu problemów,
- zna i umie zastosować równość dotyczącą ilorazu i reszty z dzielenia,
- stosuje rekurencję w zapisie algorytmu,
- programuje funkcję i stosuje ją w programach rekurencyjnych,
- uruchamia swoje programy w Pythonie i ewentualnie poprawia je, gdy działają niezgodnie ze specyfikacją, testuje ich działanie na różnych danych;

### Treści programowe – wypisy z podstawy programowej informatyki

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

1. zapisuje za pomocą listy kroków, schematu blokowego lub pseudokodu i implementuje w wybranym języku programowania algorytmy poznane na wcześniejszych etapach oraz algorytmy:
  - a. algorytm Euklidesa w wersji iteracyjnej i rekurencyjnej wraz z zastosowaniami.

<sup>1</sup> To pojęcie zaczęło się kształtować w X w. n.e.

<sup>2</sup> Jest to jedyny sposób obliczania  $NWD(m, n)$  uwzględniony w obowiązującej podstawie programowej matematyki, o algorytmie Euklidesa nie wspomina się w podstawie programowej matematyki.

*Uwaga.* Algorytm Euklidesa znajduje się również w zapisach podstawy programowej przedmiotu informatyka w szkole podstawowej dla klas VII–VIII:

Uczeń: [...] przedstawia działanie algorytmu Euklidesa w obu wersjach iteracyjnych (z odejmowaniem i z resztą z dzielenia),

W tej jednostce skupiamy uwagę na rekurencyjnej implementacji algorytmu Euklidesa, znacznie efektywniejszej niż wersja iteracyjna.

## Pojęcia i metody informatyczne

- równość wiążąca iloraz i resztę z dzielenia,
- algorytm Euklidesa,
- redukcja problemu,
- rekurencja,
- funkcja rekurencyjna,
- komputerowa realizacja algorytmu Euklidesa.

## Metody pracy w klasie

- **Słowna i Oglądowa:** dyskusja w klasie moderowana przez nauczyciela ma doprowadzić do sformułowania rekurencyjnej zależności dla NWD.
- **Czynna:** uczniowie najpierw dyskutują nad rozwiązaniem postawionego problemu, później tworzą algorytm i testują go odręcznie na przykładach, na końcu piszą odpowiedni program w Pythonie.
- **Aktywizująca:** uczniowie mogą pracować (programować) w parach, dzieląc zadania do wykonania między sobą, aktywizują się nawzajem.

## Formy pracy uczniów

- zajęcia w grupie (cała klasa) i w podgrupach, poza komputerami, dyskusje nad postawionymi przez nauczyciela pytaniami,
- praca indywidualna: odręczne wykonywanie algorytmu na przykładach,
- praca indywidualna nad stworzeniem algorytmu, a następnie by go zaprogramować.

## Przygotowanie do zajęć – materiały, urządzenia, oprogramowanie

Polecamy książki [3] (rozdz. 7) i [4] (punkty 7.4 i 7.5), w których przedstawiono różne sposoby wyprowadzenia algorytmu Euklidesa oraz jego zastosowania.

Zaleca się przygotować zadania do wykonania przez uczniów, które dotyczyłyby wcześniejszego materiału na temat NWD, z zajęć w szkole podstawowej – znajdowanie NWD( $m, n$ ) przez rozkład obu liczb na czynniki oraz iteracyjnej wersji algorytmu Euklidesa.

## Uwagi metodyczne dla nauczyciela dotyczące realizacji przyjętych celów i zapisów podstawy programowej

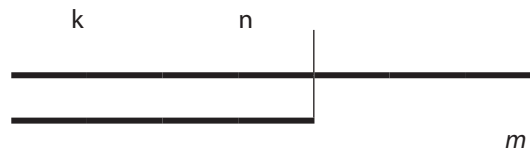
Ze sposobami znajdowania NWD uczniowie spotykają się na matematyce w szkole już w klasach IV–VI. W tym celu stosują metodę rozkładu obu liczb na czynniki. Na przykład:

$NWD(24, 44) = 4$ , gdyż  $24 = 2 \cdot 2 \cdot 2 \cdot 3$  i  $44 = 2 \cdot 2 \cdot 11$  i w obu liczbach występują czynniki  $2 \cdot 2 = 4$ . Ale jak znaleźć tą metodą  $NWD(4807, 7163)$ <sup>3</sup>? Z odpowiedzią na to pytanie uczniowie mogą sobie nie poradzić. Jak pokażemy, przyda się im w tym celu algorytm Euklidesa.

W wyprowadzeniu algorytmu Euklidesa proponujemy posłużyć się metaforą geometryczną. Dane są dwa odcinki o długościach  $m$  i  $n$ . Szukamy możliwie najdłuższego odcinka o długości  $k$ , który całkowitą liczbę razy mieści się w odcinkach o długościach  $m$  i  $n$ . Tymi odcinkami mogą być drewniane belki o długościach  $m$  i  $n$ , które chcemy pociąć na możliwie najdłuższe kawałki o takiej samej długości  $k$ .

<sup>3</sup> W podstawie programowej matematyki dla klas IV–VI występują przykłady znajdowania NWD dla liczb czterocyfrowych.

Na rysunku poniżej są przedstawione odcinki o długościach  $m$  i  $n$ , na których oznaczono również odcinek, który mieści się w nich całkowitą liczbę razy.



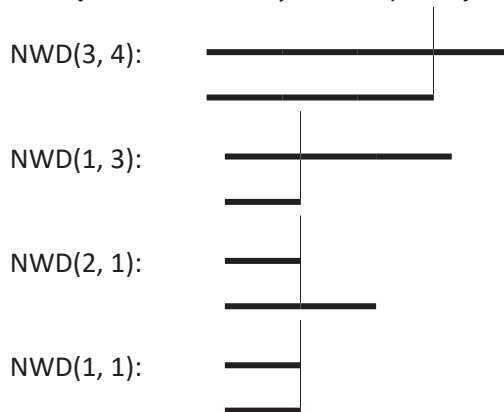
Zauważmy, że odcinek o długości  $k$  mieści się także całkowitą liczbę razy w odcinku o długości  $n - m$ , znacznie krótszym niż odcinek o długości  $n$ . Stąd możemy wyprowadzić następujący wniosek:

**jeśli**  $m = n$ , **to** oczywiście  $NWD(m, n) = m = n$

**w przeciwnym razie**  $NWD(m, n) = NWD(n - m, m)$

Konkluzja ta jest podstawą algorytmu znajdowania NWD przez odejmowanie, o którym była zapewne mowa w szkole podstawowej.

Abstrahując teraz od geometrycznej interpretacji, zapiszmy wykonaną operację za pomocą liczb przyjmując, że  $m = 4$ ,  $n = 7$ . Mamy więc:  $NWD(4, 7) = NWD(7 - 4, 4) = NWD(3, 4)$ . Powtórne wykonanie tej operacji daje nam  $NWD(3, 4) = NWD(1, 3)$ . W kolejnym kroku otrzymujemy:  $NWD(2, 1)$ . Teraz musimy zamienić rolami obie liczby, by druga była nie mniejsza niż pierwsza, i obliczyć  $NWD(1, 2)$ , co jest równe  $NWD(1, 1)$ , czyli 1. Kolejne kroki są zilustrowane na rysunkach poniżej w konwencji odcinków.



Jeśli metodą odcinania mniejszego odcinka od większego chcielibyśmy obliczyć, ile wynosi  $NWD(2, 501)$ , to musielibyśmy aż 250 razy odcinać odcinek o długości 2 od coraz krótszego dłuższego odcinka, który na początku ma długość 501, a na końcu 1. Można jednak tego uniknąć, obliczając, ile razy należy odciąć ten krótszy odcinek, by pozostał odcinek od niego krótszy. Wystarczy wykonać w tym celu dzielenie całkowite większej liczby przez mniejszą. W naszym przypadku będzie to  $\lfloor 501/2 \rfloor = 250$ . Na rysunku poniżej zilustrowano to szybsze odcinanie w przypadku obliczania  $NWD(4, 14)$  – po trzykrotnym odcięciu odcinka o długości 4 z odcinka o długości 14 redukujemy te obliczenia do  $NWD(2, 4)$ .



Rozważmy teraz ogólny przypadek  $NWD(m, n)$ . Na lekcjach matematyki uczniowie dowiadują się – zgodnie z podstawą programową – jak wyznaczyć  $q$ , wynik dzielenia z resztą liczby  $n$  przez liczbę  $m$  i zapisać liczbę  $n$  w postaci:  $n = q \cdot m + r$ , gdzie  $r$  jest resztą. Załóżmy dla jednoznaczności tego dzielenia, że reszta  $r$  spełnia nierówności:  $0 \leq r < m$ , jak powyżej przy odcinaniu krótszego odcinka z dłuższego aż do otrzymania reszty tego dłuższego odcinka, z której nie można już odciąć tego krótszego. To są naturalne warunki, jeśli bowiem  $n$  dzieli się bez reszty przez  $m$ , to  $r = 0$ , a jeśli  $n$  dzieli się z resztą przez  $m$ , to przyjmuje się na ogół naturalnie, że reszta jest mniejsza od liczby  $m$ , przez którą dzielimy. Przypominamy, że zakładamy, iż  $m \leq n$ . Otrzymujemy zatem następującą równość, zwaną **równością o ilorazie i reszcie**:

$$n = q \cdot m + r, \text{ gdzie } 0 \leq r < m$$

Wielkości  $q$  i  $r$  są odpowiednio **ilorazem** i **resztą** z dzielenia  $n$  przez  $m$ .

Z tej równości można wyciągnąć następujące dwa wnioski, które otrzymaliśmy powyżej, zajmując się odcinaniem krótszego odcinka z dłuższego odcinka:

- jeśli  $r = 0$ , to  $\text{NWD}(m, n) = m$ , czyli jeśli jedna z liczb dzieli drugą bez reszty, to mniejsza z tych liczb jest ich największym wspólnym dzielnikiem;
- jeśli  $r \neq 0$ , to równość o ilorazie i reszcie można zapisać w postaci  $r = n - qm$ , skąd wynika, że każda liczba dzieląca  $n$  i  $m$  dzieli całe wyrażenie po prawej stronie tej równości, a więc dzieli również  $r$ ; zatem największy wspólny dzielnik  $m$  i  $n$  dzieli również resztę  $r$ .

Zwłaszcza ten drugi wniosek jest fundamentalny dla naszych dalszych rozważań, powtórzmy go:

Każda liczba, która dzieli  $m$  i  $n$ , dzieli również resztę  $r$  z dzielenia  $n$  przez  $m$ .

W szczególności  $\text{NWD}(m, n)$ , czyli największy wspólny dzielnik  $m$  i  $n$ , dzieli także  $r$ .

W naszym przykładzie dla liczb  $m = 24$  i  $n = 44$ , dla których wiemy, że  $\text{NWD}(24, 44) = 4$ , mamy  $44 = 1 \cdot 24 + 20$  i rzeczywiście, reszta 20 jest podzielna przez 4.

Możemy zatem zapisać równość:

$$\text{NWD}(m, n) = \text{NWD}(r, m),$$

gdzie  $r$  jest resztą z dzielenia  $n$  przez  $m$ . Jak widać, zredukowaliśmy wartości  $m$  i  $n$ , czyli w naszym przykładzie mamy  $\text{NWD}(24, 44) = \text{NWD}(20, 24)$ . Co więcej, powyższy wniosek możemy teraz zastosować do  $r$  i  $m$ , a więc do reszty 20 i do 24, do liczby, przez którą dzieliliśmy. W naszym przykładzie to będzie:  $24 = 1 \cdot 20 + 4$ , a zatem kontynuując, mamy teraz:  $\text{NWD}(24, 44) = \text{NWD}(20, 24) = \text{NWD}(4, 20)$ . I ponownie stosujemy wniosek tym razem do liczb 4 oraz 20 i otrzymujemy:  $20 = 5 \cdot 4 + 0$ . Co potwierdza, że  $\text{NWD}(24, 44) = 4$ .

Wypiszmy ponownie kolejne równości z dzielenia i wynikające z nich równości NWD:

$$\begin{array}{ll} 44 = 1 \cdot 24 + 20 & \text{NWD}(24, 44) = \text{NWD}(20, 24) \\ 24 = 1 \cdot 20 + 4 & \text{NWD}(20, 24) = \text{NWD}(4, 20) \\ 20 = 5 \cdot 4 + 0 & \text{NWD}(4, 20) = \text{NWD}(0, 4) \end{array}$$

Wyjaśnienia wymaga, co to znaczy i jaką ma wartość  $\text{NWD}(0, 4)$ . Z definicji ma to być największy wspólny dzielnik liczb 0 i 4, i tym dzielnikiem jest 4, dlatego, że 0 dzieli się przez każdą liczbę. Uwaga, to nie jest dzielenie przez zero, tylko dzielenie liczby zero!

Przedstawione rozumowanie ma charakter rozumowania wstecz, charakterystycznego dla myślenia rekurencyjnego. Zapiszmy więc zależności, z których korzystaliśmy

$$\text{NWD}(m, n) = \text{NWD}(r, m), \quad \text{dla } n = q \cdot m + r \text{ gdzie } 0 \leq r < m$$

w szczególności, dla  $r = 0$  mamy  $\text{NWD}(0, m) = m$

Równość  $\text{NWD}(0, m) = m$  jest warunkiem zakończenia rekurencji.

Podsumujmy to rozumowanie. Wynika z niego, że możemy generować pary liczb o tym samym największym wspólnym dzielniku, elementy tych par tworzą malejący ciąg liczb naturalnych, a więc ten ciąg jest skończony i na jego końcu otrzymujemy szukany dzielnik. Jest to szkic rozumowania uzasadniającego poprawność i skończoność algorytmu Euklidesa.

Wróćmy do odłożonego na chwilę zadania, by znaleźć  $\text{NWD}(4807, 7163)$  i zastosujmy do niego algorytm Euklidesa:

$$\begin{array}{l} 7163 = 4807 \cdot 1 + 2356 \\ 4807 = 2356 \cdot 2 + 95 \\ 2356 = 95 \cdot 24 + 76 \\ 95 = 76 \cdot 1 + 19 \\ 76 = 19 \cdot 4 + 0 \end{array}$$

A zatem  $\text{NWD}(4807, 7163) = 19$ . Można mieć wątpliwości, czy równie szybko uczniowie byliby w stanie znaleźć rozkład tych liczb na czynniki,  $4807 = 11 \cdot 19 \cdot 23$  i  $7163 = 13 \cdot 19 \cdot 29$ , by znaleźć  $\text{NWD}(4807, 7163)$ . W powyższych obliczeniach jest wykonywane jedynie dzielenie (a faktycznie branie reszty z dzielenia całkowitego) i odejmowanie.

Opis algorytmu Euklidesa w Pythonie może mieć następującą postać.

```
def EuklidRek(m, n):
    if m == 0:
        return n
    else:
        return EuklidRek(n % m, m)
```

Zauważmy, że iloraz  $q$  nie występuje w następnych wywołaniach równości, zatem nie trzeba go wyznaczać. Poza tym w powyższej implementacji algorytmu Euklidesa nie sprawdzamy, która z liczb  $m$  i  $n$  jest mniejsza lub większa. Korzystamy przy tym z faktu, że jeśli byłoby  $m > n$ , to po wywołaniu `EuklidRek(m, n)` w pierwszym wywołaniu wewnątrz tej funkcji `EuklidRek(n % m, m)` faktycznie jest wykonywane wywołanie `EuklidRek(n, m)` dla  $n$  mniejszego od  $m$ , bo dla  $m > n$  wyrażenie  $n \% m$  jest równe  $n$ .

Proponujemy dwa zadania dla uczniów.

**Zadanie 1.** Jakie warunki muszą spełniać liczby  $m$  i  $n$ ,  $m \leq n$ , aby funkcja `EuklidRek(m, n)` wykonała dokładnie jedno wywołanie rekurencyjne, a jakie, by wykonała dokładnie dwa wywołania?

I nieco trudniejsze zadanie.

**Zadanie 2.** Zastosuj algorytm Euklidesa do liczb 34 i 55. Co ciekawego możesz powiedzieć o działaniu algorytmu w tym przypadku: ile wynoszą kolejne ilorazy? Wypisz od końca ciąg reszt tworzonych w tym przypadku – co ci przypomina? Podaj inną parę liczb, dla której algorytm Euklidesa działa podobnie.

Zadanie 2 jest związane z efektywnością działania algorytmu Euklidesa, piszemy o ty szczegółowo w książce [5]. Tutaj tylko podajmy, że ten algorytm jest bardzo efektywny, ma złożoność logarytmiczną, zatem dla liczb o kilkuset cęfrach wykonuje tylko kilka tysięcy działań, co jest niezmiernie istotne w wielu zastosowania, takich jak kryptografia.

## Przebieg zajęć

Przebieg tych zajęć jest zaproponowany w uwagach metodycznych. Powtórzmy w skrócie najważniejsze etapy.

1. *Dyskusja w całej klasie.* Na początku nauczyciel prosi uczniów o przypomnienie tego, co uczniowie dowiedzieli się o obliczaniu NWD w szkole podstawowej. Na matematyce w tym celu rozkładali liczby na czynniki, a na informatyce stosowali algorytm Euklidesa z dzieleniem.
2. *Indywidualna praca uczniów.* Uczniowie obliczają NWD dla wielu par liczb podanych przez nauczyciela, stosując metody poznane w szkole podstawowej.
3. *Ponownie wspólnie.* Nauczyciel przedstawia interpretację NWD, posługując się metaforą odcinków.
4. *Indywidualna praca uczniów.* Uczniowie obliczają NWD dla wybranych par liczb, stosując interpretację odcinkową.
5. *Ponownie wspólnie.* Nauczyciel podsumowuje dyskusję i przypomina równość o ilorazie i reszcie ze szkoły podstawowej. Na tej podstawie formułuje wniosek wraz z jego rekurencyjnym zapisem, który staje się podstawą rekurencyjnego algorytmu i programu obliczania NWD.
6. *Indywidualna praca uczniów.* Uczniowie stosują rekurencyjny wzór na NWD w przykładowych obliczeniach. Następnie programują algorytm rekurencyjny dla NWD i testują go na wcześniej sprawdzonych przykładach.
7. *Indywidualna praca uczniów – etap dodatkowej aktywizacji.* Nauczyciel proponuje dwa dodatkowe zadania wymienione w uwagach metodycznych. Uczniowie nie powinni mieć z nimi kłopotów.
8. *Indywidualna praca uczniów i ich ocena.* W przypadku zadań programistycznych nauczyciel motywuje uczniów, wyżej oceniając ich wkład, jeśli program został utworzony samodzielnie (przez ucznia lub w parze z innym uczniem), nie korzystając z gotowych rozwiązań, których obecnie jest bardzo wiele dostępnych w sieci.
9. *Dyskusja podsumowująca.* Ponieważ uczniowie znają trzy metody obliczania NWD, dyskusja powinna się skupić wokół pytania, którą z metod uważają za najbardziej dostosowaną zarówno dla obliczeń odręcznych, jak i komputerowych.

## Modyfikacja, poszerzenie zakresu, kontynuacja zajęć

Ten scenariusz zajęć może być poszerzony przynajmniej w dwóch kierunkach:

- dokładne wyprowadzenie, ile operacji wykonuje algorytm Euklidesa; potrzebna będzie znajomość logarytmu,
- przedstawienie zastosowań tego algorytmu.

Materiał dla tych zagadnień można znaleźć w książkach [3–5].

## 6.9 Myślenie rekurencyjne

### Temat jednostki:

#### Myślenie rekurencyjne

#### Poziom kształcenia

Szkoła ponadpodstawowa, informatyka – zakres podstawowy i rozszerzony

#### Streszczenie

**Rekurencja** to jedno z najważniejszych pojęć w informatyce. Liczne przykłady rekurencji w podstawie programowej informatyki mają zilustrować, że jest w tym pojęciu zarówno piękno, jak i prostota. By jednak dostrzec to pierwsze, trzeba to pojęcie zrozumieć, ale na przeszkodzie temu często stoi właśnie jego prostota. Powszechnie uważa się, że jest to jeden z trudniejszych do zrozumienia i władania sposobów rozumowania i rozwiązywania problemów. Zarówno dla uczniów i studentów, jak i, nie ukrywajmy tego, dla wielu nauczycieli również<sup>1</sup>.

Rekurencja to (Kahney, 1983): „proces, który jest w stanie wywoływać nowe instancje<sup>2</sup> samego siebie i sterować w tym procesie przejściem do kolejnych instancji i powrotem z zakończonych instancji”. Takie rozumienie rekurencji określa się **modelem kopii** i przyjmuje się je za właściwy/poprawny model umysłowy rekurencji. W nauczaniu dąży się, by uczniowie osiągnęli właśnie takie rozumienie rekurencji.

Rekurencja jest metodą (podejściem, techniką, narzędziem) rozwiązywania problemów poprzez dekompozycję problemu na podproblemy tego samego rodzaju i należy w tym podejściu określić: sposób rozkładu problemu na podproblemy i sposób połączenia rozwiązań podproblemów w rozwiązanie problemu. Jest to dobrze widoczne w działaniu sortowania przez scalanie. W swojej ogólności rekurencja występuje jako:

- (1) podejście do rozwiązywania problemów,
- (2) konstrukcja algorytmiczna, czyli sposób przedstawiania algorytmów,
- (3) konstrukcja programistyczna, czyli technika programowania.

W nauczaniu wszystkie te aspekty rekurencji powinny występować łącznie – myślenie rekurencyjne jest jednym z potężniejszych sposobów projektowania i wyprowadzenia rozwiązania, które następnie można zapisać w postaci rekurencyjnego algorytmu, by na końcu skorzystać z konstrukcji programistycznych w zapisie algorytmu jako programu dla komputera.

Z rekurencją, jako metodą rozwiązywania problemów, są mocno związane dwie najważniejsze cechy myślenia komputacyjnego, **abstrakcja** i **dekompozycja**. Posłużenie się abstrakcją jest niezbędne, by dostrzec strukturę problemu, poddającą się rekurencji, a dekompozycja problemu występuje na etapie projektowania rozwiązania, w którym na rozwiązanie problemu składają się rozwiązania mniejszych instancji tego samego problemu i ich połączenie w rozwiązanie problemu. Faktycznie, rekurencja jest przykładem zorganizowanego procesu zagłębiających się dekompozycji problemu na coraz mniejsze instancje tego samego problemu aż do osiągnięcia najmniejszej instancji problemu o znanym rozwiązaniu, przy tym ważny jest również proces powrotny łączenia rozwiązań mniejszych w rozwiązania instancji, z których zostały wygenerowane. Umiejętność śledzenia tego procesu wykonywania algorytmu rekurencyjnego pozwala uczniom poznać mechanizm działania rekurencji i przekonać się, na czym polega jej skuteczność.

Zajęcia na temat rekurencji powinny być poprowadzone zarówno z uczniami, którzy wybrali rozszerzenie informatyczne, jak i z pozostałymi, których jest zapewne znacznie więcej w każdej szkole. Zakres tych zajęć powinien więc być dostosowany do poziomu zagadnień, przewidzianego w podstawie programowej.

<sup>1</sup> Zaleca się, by nauczyciel traktował rekurencję jako jedno z wielu pojęć informatycznych, bez specjalnego ujawnienia, że jest to pojęcie złożone i trudne, by nie udzieliło się to uczniom.

<sup>2</sup> Używamy terminu instancja problemu na określenie wersji problemu dla określonych danych. W tym kontekście mniejsza instancja problemu to wersja problemu dla mniejszych wartości danych lub dla podzbioru danych rozważanego problemu.

Nauczyciel powinien wcześniej poinformować uczniów o zakresie tych zajęć, by mogli się do nich dobrze przygotować.

### Przygotowanie uczniów

Uczniowie powinni być przygotowani w zakresie zagadnień, w rozwiązaniach których stosowali rekurencję. Rekurencja końcowa pojawia się już w szkole podstawowej, patrz jednostka 5.6.. W szczególności wcześniej poznali:

- rekurencję końcową, programując potencjalnie nieskończone spirale,
- w algorytmach porządkowania – w sortowaniu przez scalanie oraz w innych algorytmach,
- w realizacji szybkiego potęgowania,
- w realizacji algorytmu Euklidesa.

### Cele szczegółowe zajęć – osiągnięcia uczniów

Uczeń porządkuje, pogłębia i poszerza, utrwała swoje osiągnięcia związane ze stosowaniem rekurencji w projektowaniu i implementacji bardzo efektywnych algorytmów:

- stosuje rekurencję końcową jako pewną odmianę iteracji,
- stosuje rekurencję w układaniu algorytmów, bazujących na technice dziel i zwyciężaj,
- stosuje rekurencję w realizacji zależności rekurencyjnych, takich jak w algorytmie Euklidesa,
- wiąże ideę metody dziel i zwyciężaj z reedukacją i rekurencją,
- wyjaśnia model działania rekurencji na poziomie jej implementacji,
- kształtuje umysłowy model rekurencji.

### Treści programowe – wypisy z podstawy programowej informatyki

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

6. zapisuje za pomocą listy kroków, schematu blokowego lub pseudokodu i implementuje w wybranym języku programowania algorytmy poznane na wcześniejszych etapach oraz algorytmy:
  - l. algorytm Euklidesa w wersji iteracyjnej i rekurencyjnej wraz z zastosowaniami,
  - m. [...],
  - n. [...],
  - o. [...],
  - p. sortowania ciągu liczb przez scalanie,
  - q. [...],
  - r. [...],
  - s. [...],
  - t. szybkiego potęgowania liczb w wersji iteracyjnej i rekurencyjnej,
  - u. [...],
  - v. rekurencyjnego tworzenia fraktali: zbiór Cantora, drzewo binarne, dywan Sierpińskiego, płatek Kocha;
7. wykorzystuje znane sobie algorytmy przy rozwiązywaniu i programowaniu rozwiązań następujących problemów:
  - g. [...],
  - h. [...],
  - i. [...],
  - j. zamiany wyrażenia na postać w odwrotnej notacji polskiej i obliczanie jego wartości na podstawie tej postaci,
  - k. [...],
  - l. [...];
8. objaśnia, a także porównuje podstawowe metody i techniki algorytmiczne oraz struktury danych, wykorzystując przy tym przykłady problemów i algorytmów, w szczególności:
  - k. [...],
  - l. rekurencję (do generowania ciągów liczb, potęgowania, sortowania liczb, generowania fraktali),
  - m. metodę dziel i zwyciężaj (jednoczesne znajdowanie minimum i maksimum, sortowanie przez scalanie i szybkie),

- n. [...],
- o. [...],
- p. [...],
- q. [...],
- r. [...],
- s. struktury dynamiczne: stos, kolejka, lista (do realizacji algorytmu: ONP, symulacji problemu Flawiusza, sortowania leksykograficznego),
- t. grafy (do przedstawiania abstrakcyjnego modelu sytuacji problemowych).

## Pojęcia i metody informatyczne

- abstrakcja,
- dekompozycja,
- rekurencja,
- rekurencja końcowa,
- strategia dziel i zwyciężaj,
- model rekurencji,
- rekurencja a iteracja – liczby Fibonacciego,
- rekurencyjne rozwiązywanie: porządkowanie, szybkie potęgowanie, algorytm Euklidesa, Odwrotna Notacja Polska (ONP, RPN).

## Metody pracy w klasie

- **Słowna i Oglądowa:** dyskusja w klasie, moderowana przez nauczyciela na temat metod i algorytmów wykorzystujących rekurencję.
- **Czynna:** zajęcia mają głównie formę czynną, nauczyciel ewentualnie koryguje wypowiedzi lub oceny uczniów; zajęcia w żadnym zakresie nie powinny mieć formy wykładu.
- **Aktywizująca:** nauczyciel stara się aktywizować uczniów pytaniami, które mają na celu spojrzenie na omawianą tematykę „z lotu ptaka”, z góry, ponad szczegółami związanymi z konkretnymi zagadnieniami, metodami i algorytmami.

## Formy pracy uczniów

- zajęcia głównie w grupie (cała klasa) i w podgrupach, poza komputerami, dyskusje nad postawionymi przez nauczyciela pytaniami i zadaniami do wykonania.
- praca w niewielkim stopniu indywidualna, raczej niezwiązana z programowaniem, ewentualnie skorzystanie z wcześniej wykonanych programów,

## Przygotowanie do zajęć – materiały, urządzenia, oprogramowanie

Nauczyciel powinien przygotować wykaz tych zagadnień z podstawy programowej, które są rozwiązywane z wykorzystaniem rekurencji.

Jednym z zagadnień na tych zajęciach powinno być porównanie rekurencji z iteracją na wybranych przykładach, np. liczb Fibonacciego.

W książce [5] cały rozdział jest poświęcony mechanizmom działania i implementacji rekurencji w komputerze. Na tych zajęciach warto przybliżyć uczniom wybrane mechanizmy na przykładach. **Diagram wywołań rekurencyjnych** jest bardzo prosty dla sortowania przez scalanie (patrz jednostka 6.5), również proste jest **drzewo wywołań rekurencyjnych** dla liczby Fibonacciego. Nauczyciel powinien przygotować ilustracje odnoszące się do tych mechanizmów, by uczniowie mogli samodzielnie popracować nad nimi.

## Uwagi metodyczne dla nauczyciela dotyczące realizacji przyjętych celów i zapisów podstawy programowej

Ważne, by takie zajęcia były osobno przeprowadzone dla uczniów, którzy obrali informatykę w zakresie podstawowym, jak i dla tych, którzy wybrali rozszerzenie informatyczne. Dla tej pierwszej grupy mogą mieć większe znaczenie niż dla tej drugiej.



Spojrzenie i ukierunkowanie dyskusji przez nauczyciela powinno być „ponad” konkretnymi problemami, metodami i algorytmami. Uczniowie powinni nie tylko posługiwać się konkretnym algorytmem, ale także umieć go scharakteryzować, umieścić wśród innych metod rozwiązywania pokrewnych zagadnień podobnymi metodami, ocenić go na tle innych metod. Dostrzec także możliwości modyfikacji.

### Przebieg zajęć

Przebieg zajęć scharakteryzowano już wcześniej. Głównie są to zajęcia dyskusyjne, poza komputerem, dobrze przygotowane przez nauczyciela, w których biorą udział uczniowie również dobrze przygotowani, wcześniej poinformowani o zakresie zajęć przez nauczyciela.

Zajęcia te mogą również służyć wyjaśnieniu kwestii, które pojawiają się u uczniów w trakcie przygotowywania się. Zwłaszcza, jak wiadomo, rekurencja to najtrudniejsze pojęcie w informatyce szkolnej i często się zdarza, że uczniowie stosują rekurencję w programach, ale nie bardzo wiedzą, dlaczego i jak działa.

Dobrym krokiem ku pogłębieniu zrozumienia rekurencji przez uczniów będzie utworzenie przez nich struktury wywołań wybranego programu rekurencyjnego dla wybranych danych. Taka przykładowa struktura znajduje się w jednostce 6.5 w odniesieniu do sortowania przez scalanie. Na tych zajęciach uczniowie powinni wrócić do tego programu i narysować taką strukturę dla innego ciągu.

Ciekawa struktura wywołań powstaje przy obliczaniu wartości liczb Fibonacciego – jest to drzewo, w którym wiele wartości jest liczonych wielokrotnie, stąd wniosek, że wersja iteracyjna jest bardziej efektywna. Podobnie uczniowie powinni łatwo poradzić sobie z tą strukturą.

Większym wyzwaniem i bardziej motywującym uczniów może być utworzenie struktur wywołań rekurencyjnych dla innych algorytmów rekurencyjnych – szybkiego podnoszenia do potęgi i algorytmu Euklidesa.

### Modyfikacja, poszerzenie zakresu, kontynuacja zajęć

Poszerzeniem tematu rekurencji może być analiza mechanizmów działania rekurencji, co może mieć istotny wpływ na rozumienie tej tematyki przez uczniów, zwłaszcza tych, którzy obrali informatykę w zakresie rozszerzonym.

Tego typu zajęcia powinny zostać zorganizowane przez nauczyciela na wiele różnych tematów, które wymagają od ucznia uporządkowania, pogłębienia i poszerzenia, utrwalenia swojej wiedzy, umiejętności i osiągnięć. Inna jednostka 6.6 jest poświęcona metodom i algorytmom wyszukiwania i porządkowania. Inne ważne zajęcia zbiorcze powinny być poświęcone myśleniu komputacyjnemu.

## 6.10 Programowanie dynamiczne

### Temat jednostki:

#### Programowanie dynamiczne

co najmniej dwie lekcje

### Poziom kształcenia

Szkoła ponadpodstawowa, informatyka – zakres rozszerzony

### Streszczenie

Zacznijmy od stwierdzenia, że programowanie dynamiczne ma niewiele wspólnego z programowaniem komputerów poza tym, że algorytmy programowania dynamicznego można zapisywać w postaci programów i wykonywać na komputerze.

Programowanie dynamiczne to technika algorytmiczna służąca do znajdowania najlepszych (**optymalnych**) rozwiązań. Algorytm programowania dynamicznego działa zgodnie z **Zasadą Bellmana**:

**na każdym kroku podejmuj najlepszą decyzję z uwzględnieniem stanu wynikającego z poprzednich decyzji**

Ta zasada brzmi podobnie jak strategia zachłanna zalecająca podejmowanie najlepszych decyzji na każdym kroku, jednak różni się od niej tym, że należy uwzględnić wszystkie możliwe stany wynikające z poprzednich decyzji. Podany poniżej przykład problemu plecakowego ilustruje tę różnicę.

Programowanie dynamiczne jest często stosowane do znajdowania najlepszych rozwiązań problemów, dla których nie są znane inne metody. Polega na dobrze uporządkowanym przeglądaniu wszystkich możliwości i podejmowaniu decyzji zgodnie z Zasadą Bellmana. W tej jednostce ilustrujemy programowanie dynamiczne klasycznym przykładem problemu plecakowego.

Problem, bliski oryginalnemu, jest zaprezentowany i rozwiązany w książce [3] w rozdz. 15. Szczegółowe omówienie programowania dynamicznego jest zawarte w punkcie 11.2 w książce [4].

## Przygotowanie uczniów

Uczniowie wcześniej poznali:

- podejście zachłanne – patrz jednostka 5.10.

## Cele szczegółowe zajęć – osiągnięcia uczniów

Uczeń:

- stosuje podejście zachłanne przy podejmowaniu decyzji.
- tworzy rozwiązania problemów, stosując strategię programowania dynamicznego,
- programuje algorytm programowania dynamicznego,
- uruchamia swoje programy w Pythonie i ewentualnie poprawia je, gdy działają niezgodnie ze specyfikacją, testuje ich działanie na różnych danych.

## Treści programowe – wypisy z podstawy programowej informatyki

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

9. objaśnia, a także porównuje podstawowe metody i techniki algorytmiczne oraz struktury danych, wykorzystując przy tym przykłady problemów i algorytmów, w szczególności:
  - d. podejście zachłanne (do wydawania reszty, pakowania plecaka [...]),
  - e. programowanie dynamiczne (do pakowania plecaka [...]).

## Pojęcia i metody informatyczne

- problem plecakowy,
- zasada optymalności Bellmana,
- strategia zachłanna znajdowania najlepszych rozwiązań,
- metoda programowania dynamicznego – dwie fazy,
- komputerowa realizacja programowania dynamicznego.

## Metody pracy w klasie

- **Słowna i Oglądowa:** dyskusja w klasie, moderowana przez nauczyciela; nauczyciel przedstawia problem, wsparty przykładem; najpierw sugeruje podejście zachłanne, a później programowania dynamicznego.
- **Czynna:** uczniowie na własnych przykładach stosują podejście zachłanne, a później programowania dynamicznego, aby otrzymać rozwiązanie problemu plecakowego; później tworzą algorytm i testują go odrębnie na przykładach, na końcu piszą odpowiedni program w Pythonie.
- **Aktywizująca:** uczniowie mogą pracować (programować) w parach, dzieląc zadania do wykonania między sobą, aktywizują się nawzajem; podział zadań w parze może dotyczyć faz programowania dynamicznego.


## Formy pracy uczniów

- zajęcia w grupie (cała klasa) i w podgrupach, poza komputerami, dyskusje nad postawionymi przez nauczyciela pytaniami,
- praca indywidualna: odrębne wykonywanie algorytmów na własnych przykładach,
- praca indywidualna nad utworzeniem algorytmu, a następnie nad jego zaprogramowaniem.

## Przygotowanie do zajęć – materiały, urządzenia, oprogramowanie

Problem, bliski oryginalnemu z prac R. Bellmana, jest zaprezentowany i rozwiązany w książce [4] w rozdz. 15. Polecamy również punkt 11.2 w książce [4], gdzie wszechstronnie omówiono problem plecakowy i metody jego rozwiązywania.

Myślenie w kategoriach programowania dynamicznego jest przydatne przy rozwiązywaniu niektórych zadań w konkursie Bóbr. Jedno z nich znalazło się w zestawie dla uczniów z klas IV–VI w 2019 roku, patrz <https://www.mauthor.com/present/6596165404590080> i poniżej. Polecamy wyszukanie podobnych zadań we wcześniejszych zestawach.



# XIV KONKURS BÓBR

Listopad 2019  
Poziom Benjamin

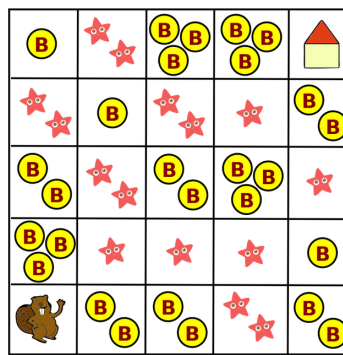
Droga do domu

Pytanie za 5 punktów

Bóbr Tadek chce wrócić do domu. Na poniższej mapie może poruszać się w górę lub w prawo.

W każdym kwadracie mapy znajdują się potwory lub monety. Tadek zbiera wszystkie monety wzdłuż ścieżki, którą podąża. Ale kiedy przechodzi przez kwadrat mapy, w którym są potwory, musi dać jedną monetę każdemu potworowi. Jeśli nie ma dość monet, by dać potworom, to może je przekazać później, gdy je zdobędzie na dalszej drodze.

Na mapie obok, klikając na odpowiednie pola, zaznacz drogę Tadeka do domu, wzdłuż której zbierze on największą liczbę monet.



## Uwagi metodyczne dla nauczyciela dotyczące realizacji przyjętych celów i zapisów podstawy programowej

W opisie Przebiegu zajęć proponujemy, by na początku uczniowie zajęli się rozwiązaniem wybranych zadań z konkursu Bóbr.

Inna ilustracja zastosowania programowania dynamicznego jest opisana w książce [3] w rozdz. 15. Polecamy do ewentualnego wykorzystania – jest bardzo przejrzysta i intuicyjna, geometryczna interpretacja kolejnych kroków rozwiązywania jest bardzo przystępna nawet dla początkującego ucznia.

**Problem plecakowy** polega na zapakowaniu do plecaka o ograniczonej pojemności możliwie najbardziej wartościowych rzeczy. W innych zastosowaniach może to dotyczyć pakowania walizek, paczek, samochodu, samolotu itp. W postaci specyfikacji ten problem definiujemy następująco:

*Dane:*  $n$  rzeczy (towarów, produktów itp.)  $R_1, R_2, \dots, R_n$ , każda w nieograniczonej ilości; rzecz  $R_i$  waży (lub zajmuje miejsce o wielkości)  $w_i$  jednostek i ma wartość  $p_i$ . Ponadto dana jest maksymalna pojemność plecaka, wynosząca  $W$  jednostek.

*Wyniki:* Ilości  $q_1, q_2, \dots, q_n$  poszczególnych rzeczy (mogą być zerowe), których całkowita waga nie przekracza  $W$  i których sumaryczna wartość jest największa wśród wypełnień plecaka rzeczami o wadze nie przekraczającej  $W$ .

Rozważania ilustrujemy przykładem:

i	1	2	3	4	5	6	W
$p_i$	6	4	5	7	10	2	
$w_i$	6	2	3	2	3	1	23

### Podejście zachłanne

Pierwsze próby zapakowania najbardziej wartościowego plecaka mogą polegać na zachłannym pakowaniu, a więc na podejściu, w którym na każdym kroku podejmujemy najlepszą decyzję. Dwie pierwsze strategie zachłanne są dość oczywiste: pakujemy najcenniejsze rzeczy najpierw albo pakujemy najpierw najlżejsze rzeczy. W obu przypadkach pamiętamy, by nie przekroczyć pojemności plecaka. W ten sposób otrzymujemy rozwiązania:

Po najcenniejszych rzeczach: 7 x rzecz 5; 1 x rzecz 4      wartość plecaka =  $7 \cdot 10 + 1 \cdot 7 = 77$   
 Po najlżejszych rzeczach: 23 x rzecz 6      wartość plecaka =  $23 \cdot 2 = 46$

Zapakujmy jeszcze plecak w kolejności nierosnących proporcji wartości do wagi. W naszym przykładzie nierosnącej kolejności tych ilorazów ( $7/2, 10/3, 4/2, 2/1, 5/3, 6/6$ ) odpowiada następująca kolejność rzeczy: (4, 5, 2, 6, 3, 1). Zatem otrzymujemy upakowanie plecaka:

Po nierosnących proporcjach: 11 x rzecz 4; 1 x rzecz 6      wartość plecaka =  $11 \cdot 7 + 1 \cdot 2 = 79$

Otrzymaliśmy najcenniejszą zawartość plecaka wśród zachłannie pakowanych. A czy jest to najlepsze upakowanie? Na to odpowie nam podejście programowania dynamicznego.

### Programowanie dynamiczne

Metodę programowania dynamicznego można uznać za pewne zastosowanie zasady dziel i zwyciężaj. Nie potrafimy jednak zawczasu przewidzieć, z których rozwiązań mniejszych problemów będziemy korzystać w następnych krokach i dlatego, przygotowując się na każdą ewentualność, rozwiązujemy wszystkie mniejsze problemy. W przypadku problemu plecakowego mniejsze problemy odpowiadają mniejszej liczbie rodzajów rzeczy, które do niego pakujemy i mniejszej pojemności plecaka. Rozwiązania wszystkich podproblemów można więc umieścić w tablicy, której wiersze odpowiadają kolejnym rzeczom, czyli są ponumerowane liczbami od 1 do  $n$ , a kolumny są kolejnymi pojemnościami plecaka, od 1 aż do maksymalnej dopuszczalnej, równej  $W$  – patrz poniżej.

$i, j$	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	
1	0	0	0	0	0	6	6	6	6	6	6	12	12	12	12	12	12	18	18	18	18	18	18	18
2	0	4	4	8	8	12	12	16	16	20	20	24	24	28	28	32	32	36	36	40	40	44	44	44
3	0	4	5	8	9	12	13	16	17	20	21	24	25	28	29	32	33	36	37	40	41	44	44	45
4	0	7	7	14	14	21	21	28	28	35	35	42	42	49	49	56	56	63	63	70	70	77	77	77
5	0	7	10	14	17	21	24	28	31	35	38	42	45	49	52	56	59	63	66	70	73	77	77	80
6	2	7	10	14	17	21	24	28	31	35	38	42	45	49	52	56	59	63	66	70	73	77	77	80

Pola w tej tablicy oznaczmy przez  $P_{ij}$ , gdzie  $i$  jest numerem (indeksem) wiersza i przebiega wszystkie rodzaje rzeczy, które mogą być umieszczone w plecaku. Kolejność rozpatrywania rzeczy nie ma znaczenia w algorytmie programowania dynamicznego. Z kolei indeks  $j$  przebiega od 1, co 1, aż do maksymalnej pojemności plecaka  $W$ . Zatem tablica  $P$  dla naszego przykładu ma wymiar 6 wierszy na 23 kolumny. Wartość w polu  $P_{ij}$  definiujemy następująco — jest to wartość optymalnego wypełnienia plecaka o pojemności  $j$  rzeczami, których numery mieszczą się między 1 a  $i$ .

Algorytm programowania dynamicznego jest sposobem wypełniania pól tablicy  $P$ . Pola te są wypełniane wierszami. Wypełniając pierwszy wiersz, dysponujemy tylko rzeczami pierwszego rodzaju, które ważą 6 jednostek każda i mają dla nas wartość 6. Wynika stąd, że dla  $i = 1$  oraz dla początkowych pojemności  $j = 1, 2, 3, 4, 5$  w odpowiednich polach tablicy znajdują się zera. Dopiero gdy  $j = 6$ , można w nim umieścić pierwszą sztukę rzeczy nr 1. Następne pozycje aż do  $j = 11$  są równe 6, gdyż tylko jedna sztuka pierwszej rzeczy może być włożona do plecaka. Gdy  $j = 12$ , możemy umieścić już dwie sztuki, a więc w polu o współrzędnych (1, 12) znajdzie się liczba 12 jako wartość dwóch sztuk pierwszej rzeczy. I tak dalej. Kolejna zmiana wartości nastąpi na pozycji (1, 18) i do końca tego wiersza nie ulegnie już zmianie.

Wypełnianie następnych wierszy tablicy  $P$  jest już bardziej złożone, gdyż dysponujemy większą ilością różnych rzeczy. Możemy przy tym korzystać z już wypełnionych wierszy. Reguła korzystania z wcześniej obliczonych wartości w polach tablicy  $P$  jest odbiciem Zasady Bellmana. W tym szczególnym przypadku, w drugim

wierszu, czyli dla  $i = 2$ , zgodnie z tą zasadą za najlepsze upakowanie plecaka o pojemności  $j$  wybieramy albo jego upakowanie rzeczami nr 1 (to znaczy z poprzedniego wiersza tablicy  $P$ , czyli dla  $i = 1$ ), albo dokładamy jedną rzecz nr 2 (gdy się zmieści), a resztę pojemności plecaka wypełniamy optymalnie rzeczami nr 1 lub 2. Bardziej konkretnie, jeśli pojemność plecaka  $j$  jest taka, że zmieści się w nim rzecz nr 2, czyli  $j \geq w_2$ , to wybieramy większą z dwóch wielkości:  $P_{1,j}$  i  $P_{2,j-w_2} + p_2$  – pierwsza jest wartością najlepszego wypełnienia plecaka o pojemności  $j$  rzeczami nr 1, a druga jest wartością wypełnienia plecaka, składającego się z jednej sztuki rzeczy nr 2 oraz z optymalnego wypełnienia reszty przestrzeni w plecaku, czyli  $j - w_2$ , rzeczami nr 1 lub 2. Po wypełnieniu drugiego wiersza zgodnie z tą zasadą okazuje się, że najwartościowsze upakowanie plecaka rzeczami nr 1 lub 2 otrzymujemy, zabierając tylko rzeczy nr 2.

Dalsze wiersze wypełniamy podobnie. Poniżej ilustrujemy decyzję na pozycji  $P_{3,15}$  – mamy możliwość dobrania jeszcze jednej rzeczy nr 3, albo rezygnacji z wszystkich rzeczy nr 3. Obliczenia pokazują, że wybieramy tę pierwszą ewentualność. Ostatecznie rozwiązanie naszego przykładu, czyli  $P_{3,15}$ , wynosi 80, zatem żadne rozwiązanie zachłanne nie jest optymalne.

$i \backslash j$	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
1	0	0	0	0	0	6	6	6	6	6	6	12	12	12	12	12	12	18	18	18	18	18	18
2	0	4	4	8	8	12	12	16	16	20	20	24	24	28	28	32	32	36	36	40	40	44	44
3	0	4	5	8	9	12	13	16	17	20	21	24	25	28	29	32	33	36	37	40	41	44	45
4	0	7	7	14	14	21	21	28	28	35	35	42	42	49	49	56	56	63	63	70	70	77	77
5	0	7	10	14	17	21	24	28	31	35	38	42	45	49	52	56	59	63	66	70	73	77	80
6	2	7	10	14	17	21	24	28	31	35	38	42	45	49	52	56	59	63	66	70	73	77	80

pojemność plecaka

max{24+5, 28}=29

rzeczy

Tablica  $P$  zawiera wartości upakowań plecaka dla różnych jego pojemności, od 1 do 23, i rodzajów rzeczy od nr 1 do nr 6, w szczególności jaki zestaw rzeczy tworzy najcenniejsze wypełnienie plecaka w naszym przykładzie. Można to osiągnąć, kojarząc z tablicą  $P$  tablicę  $Q$  o tych samych wymiarach, patrz poniżej. Wartością  $Q_{i,j}$  jest numer rzeczy, która ostatnia trafiła do plecaka o pojemności  $j$ , gdy dysponujemy rzeczami o numerach od 1 do  $i$ .

W jaki sposób można odczytać rozwiązanie problemu plecakowego z tych dwóch tablic? Dla naszego przykładu, największa wartość plecaka o pojemności 23 załadowanego rzeczami spośród od 1 do 6 jest dana w polu  $P_{6,23}$  i wynosi 80 (znajduje się na szarym tle). Natomiast rzeczy składające się na wypełnienie plecaka o tej wartości znajdujemy w tablicy  $Q$ , zaczynając od pola o tych samych indeksach – jest tam liczba 5. Oznacza to, że rzecz nr 5 była ostatnią rzeczą włożoną do plecaka. Ponieważ ta rzecz waży 3 jednostki, więc pozostaje do wypełnienia pojemność plecaka równa  $23 - 3 = 20$ , patrzymy więc na pole  $Q_{6,20}$  – znajduje się tam rzecz nr 4, która waży 2 jednostki. Zatem przenosimy się na pole  $Q_{6,18}$ , w którym jest również rzecz nr 4. Kontynuujemy to odczytywanie numerów rzeczy z tablicy  $Q$ , aż do wyczerpania pojemności plecaka. W tabeli poniżej na szarym tle są umieszczone numery rzeczy tworzących rozwiązanie naszego przykładu. Zatem optymalne upakowanie plecaka o pojemności 23 składa się z 10 sztuk rzeczy nr 4 i z jednej sztuki rzeczy nr 5.

Patrząc na to rozwiązanie, można uzasadnić, dlaczego trzecie rozwiązanie zachłanne było tak blisko od optymalnego, ale nie osiągnęło go. Byliśmy zbyt zachłanni pod koniec tworzenia rozwiązania – nie należało brać jeszcze jednej rzeczy nr 6 a później rzeczy nr 6, tylko wziąć rzecz nr 5.

$i, j$	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
1	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
2	0	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
3	0	2	3	2	3	2	3	2	3	2	3	2	3	2	3	2	3	2	3	2	3	2	3
4	0	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4
5	0	4	5	4	5	4	5	4	5	4	5	4	5	4	5	4	5	4	5	4	5	4	5
6	6	4	5	4	5	4	5	4	5	4	5	4	5	4	5	4	5	4	5	4	5	4	5

Zanim podamy szczegółowy opis przedstawionego algorytmu, zauważmy, że rozwiązanie przykładu z tabeli 11.1 można otrzymać, posługując się jedynie wektorami, czyli tablicami jednowymiarowymi zamiast dwuwymiarowych. Tablice  $P$  i  $Q$  są jednak przydatne, gdy interesują nas wyniki rozwiązania tego przykładu dla mniejszej liczby różnych rodzajów rzeczy. Powyższe rozważania zakończmy dokładniejszym opisem algorytmu programowania dynamicznego dla problemu plecakowego.

*Dane:*  $n$  rzeczy, każda w nieograniczonej ilości, ważących  $w_i$  i mających wartość  $p_i$  ( $i = 1, 2, \dots, n$ ). Ponadto dana jest maksymalna pojemność plecaka, wynosząca  $W$ .

*Wyniki:* Tablica wartości  $P_{i,j}$  najlepszych upakowań plecaka o pojemności  $j$  rzeczami rodzajów od 1 do  $i$ , dla  $i = 1, 2, \dots, n$  oraz  $j = 1, 2, \dots, W$ ; skojarzona tablica rodzajów rzeczy  $Q_{i,j}$  dołożonych w ostatnim dopakowaniu plecaka.

Krok 1. {Ustalenie wartości początkowych w tablicach  $P$  i  $Q$  rozszerzonych, dla ujednoczenia obliczeń, o wiersze i kolumny zerowe.} Dla  $j = 1, 2, \dots, W$  przypisz  $P_{0,j} := 0, Q_{0,j} := 0$ . Dla  $i = 1, 2, \dots, n$  przypisz  $P_{i,0} := 0, Q_{i,0} := 0$ .

Krok 2. Dla kolejnych rzeczy  $i = 1, 2, \dots, n$  wykonaj krok 3.

Krok 3. Dla kolejnych pojemności plecaka  $j = 1, 2, \dots, W$  wykonaj krok 4.

Krok 4. **Jeśli**  $j \geq w_i$  {czyli, gdy pojemność plecaka jest wystarczająca, by pomieścić rzecz  $i$ }

**oraz**  $P_{i-1,j} < P_{i,j-w_i} + p_i + p$

**to**  $P_{i,j} := P_{i,j-w_i} + p_i; Q_{i,j} := i$

**w przeciwnym razie**  $P_{i,j} := P_{i-1,j}; Q_{i,j} := Q_{i-1,j}$  {pozostawiamy wartości z poprzedniego wiersza}

Zauważmy, że wzory w Kroku 4 na wartości w tablicy  $P$  mają postać zależności rekurencyjnych, dla których warunki początkowe są określone w Kroku 1.

Zapisanie tego algorytmu w Pythonie pozostawiamy do samodzielnego wykonania – to dwie instrukcje pętli, jedna w drugiej, i dla każdej pary indeksów, po sprawdzeniu warunku, po jednym przypisaniu w każdej tablicy (patrz punkt 11.2 w [4]).

## Przebieg zajęć

Przebieg tych zajęć jest zaproponowany w Uwagach metodycznych i w przygotowaniu do zajęć. Powtórzmy w skrócie najważniejsze etapy.

1. *Dyskusja w całej klasie.* Na początku nauczyciel przedstawia wybrane zadanie z konkursu Bóbr i poleca uczniom rozwiązanie innych podobnych zadań, w których należy posłużyć się tą samą metodą.
2. *Indywidualna praca uczniów.* Uczniowie rozwiązują zadania z konkursu Bóbr. Swoje rozwiązania konsultują z wyjaśnieniami, które towarzyszą zadaniom, w których jest podane rozwiązanie wraz z uzasadnieniem.
3. *Indywidualna praca uczniów.* Nauczyciel rozdaje siatkę z liczbami, podobną do użytej w rozdz. 15 w [3] i poleca wypełnić ją zgodnie z zasadą, którą podaje.  
Dalsza część zajęć jest szczegółowo opisana w Uwagach metodycznych.
4. *Ponownie wspólnie.* Nauczyciel krótko wyjaśnia, jaki będzie przebieg dalszych zajęć i i aktywności uczniów. Jednocześnie rozdaje kartki z danymi oraz puste tabele do wypełnienia. Może skorzystać z przykładu podanego w Uwagach metodycznych.
5. *Indywidualna praca uczniów.* Uczniowie pracują samodzielnie nad otrzymaniem rozwiązań zachłanych i wypełnieniem tabel dla rozwiązania optymalnego.
6. *Indywidualna praca uczniów.* Uczniowie programują algorytm programowania dynamicznego, testują go na przykładach wcześniej odręcznie przeliczonych.
7. *Indywidualna praca uczniów i ich ocena.* W przypadku zadań programistycznych nauczyciel motywuje uczniów, wyżej oceniając ich wkład, jeśli program został utworzony samodzielnie (przez ucznia lub w parze z innym uczniem), nie korzystając z gotowych rozwiązań, których obecnie jest bardzo wiele dostępnych w sieci.
8. *Dyskusja podsumowująca.* Nauczyciel inicjuje dyskusję na temat programowania dynamicznego jako metody otrzymywania optymalnych rozwiązań. Dyskusja może także dotyczyć pracochłonności metody – czy zdaniem uczniów jest efektywna, czy nie? Sam program jest bardzo prosty, ale liczba wyników wypełniających tabele może się wydawać, że jest dość duża. A ile dokładnie – nauczyciel prosi uczniów, by policzyli. Teoretycznie algorytm programowania dynamicznego nie jest efektywny.

### **Modyfikacja, poszerzenie zakresu, kontynuacja zajęć**

W literaturze i w różnych konkursach można znaleźć wiele zadań, których rozwiązanie sprowadza się do zastosowania programowania dynamicznego.

Ze względu na dość dużą złożoność tematu i wielu aktywności uczniów sugeruje się, by temat programowania dynamicznego rozłożyć na 2–3 lekcje. Pierwsza powinna być poświęcona wyrobieniu intuicji na przykładach, takich jak zadania z konkursu Bóbr czy problem z rozdz. 15 w książce [3]. Kolejna lekcja może dotyczyć podejścia zachłannego. Wreszcie wydzielona godzina może wystarczyć na omówienie algorytmu programowania dynamicznego wraz z napisaniem dla niego programu.