

# Spis treści

O autorze	4
O korektorach merytorycznych	4
Podziękowania	17

<b>WPROWADZENIE</b>	<b>19</b>
Kto powinien przeczytać tę książkę?	20
Co znajdziesz w tej książce?	21
Jak używać tej książki?	23
Wpisywanie kodu źródłowego	23
Sprawdzanie pod kątem błędów	23
Konwencje zastosowane w książce	24
Zasoby w internecie	24
Pobieranie i instalowanie Pythona	24
Instalacja Pythona w systemie Windows	24
Instalacja Pythona w systemie macOS	25
Instalacja Pythona w systemie Ubuntu	25
Pobieranie pliku pyperclip.py	25
Uruchamianie środowiska IDLE	26
Podsumowanie	27

<b>1</b>	
<b>PAPIER JAKO NARZĘDZIE KRYPTOGRAFICZNE</b>	<b>29</b>
Co to jest kryptografia?	30
Kod a szyfr	30
Szyfr Cezara	32
Krążek szyfrowania	32
Szyfrowanie wiadomości za pomocą krążka szyfrowania	33
Deszyfrowanie za pomocą krążka szyfrowania	34
Szyfrowanie i deszyfrowanie z użyciem arytmetyki	35
Dlaczego podwójne szyfrowanie nie działa?	36
Podsumowanie	36

<b>2</b>	
<b>PROGRAMOWANIE W POWŁOCE INTERAKTYWNEJ</b>	<b>39</b>
Kilka prostych wyrażeń matematycznych	40
Wartości całkowite i wartości zmiennoprzecinkowe	41
Wyrażenia	41
Kolejność wykonywania działań	42
Obliczanie wartości wyrażeń	42

Przechowywanie wartości w zmiennych .....	43
Nadpisywanie zmiennej .....	45
Nazwy zmiennych .....	46
Podsumowanie .....	47

### 3

<b>CIĄGI TEKSTOWE I TWORZENIE PROGRAMÓW .....</b>	<b>49</b>
Praca z tekstem przy użyciu wartości w postaci ciągu tekstowego .....	50
Konkatenacja ciągu tekstowego za pomocą operatora + .....	51
Replikacja ciągu tekstowego przy użyciu operatora * .....	52
Pobieranie znaków z ciągu tekstowego przy użyciu indeksów .....	53
Wyświetlanie wartości za pomocą funkcji print() .....	56
Wyświetlanie znaków sterujących .....	57
Apostrof i cudzysłów .....	58
Tworzenie programów w edytorze pliku IDLE .....	59
Kod źródłowy programu typu Witaj, świecie! .....	59
Sprawdzanie kodu źródłowego za pomocą narzędzia Online Diff Tool .....	61
Użycie środowiska IDLE w celu późniejszego uzyskania dostępu do programu .....	62
Zapisywanie programu .....	62
Uruchamianie programu .....	63
Otwieranie wcześniej zapisanych programów .....	64
W jaki sposób działa program Witaj, świecie!? .....	64
Komentarze .....	64
Wyświetlanie wskazówek dla użytkownika .....	65
Pobieranie danych wejściowych od użytkownika .....	65
Zakończenie programu .....	66
Podsumowanie .....	66

### 4

<b>SZYFR ODWROTNY .....</b>	<b>69</b>
Kod źródłowy programu wykorzystującego szyfr odwrotny .....	70
Przykładowe uruchomienie programu .....	70
Definiowanie komentarzy i zmiennych .....	71
Określanie długości ciągu tekstowego .....	72
Wprowadzenie do pętli while .....	73
Boolowski typ danych .....	73
Operatory porównania .....	74
Blok kodu .....	76
Konstrukcja pętli while .....	77
„Rośnięcie” ciągu tekstowego .....	78
Usprawnianie programu za pomocą funkcji input() .....	81
Podsumowanie .....	82

### 5

<b>SZYFR CEZARA .....</b>	<b>85</b>
Kod źródłowy programu wykorzystującego szyfr Cezara .....	86
Przykładowe uruchomienie programu .....	87
Importowanie modułu i przypisywanie zmiennych .....	88
Stałe i zmienne .....	89

Pętla for .....	90
Przykład pętli for .....	90
Pętla while będąca odpowiednikiem pętli for .....	91
Konstrukcja if .....	92
Przykład użycia polecenia if .....	92
Polecenie else .....	92
Polecenie elif .....	93
Operatory in i not in .....	94
Metoda find() .....	95
Szyfrowanie i deszyfrowanie symboli .....	96
Obsługa zawinięcia .....	97
Obsługa symboli spoza zbioru symboli .....	98
Wyświetlanie i kopiowanie skonwertowanego ciągu tekstowego .....	98
Szyfrowanie innych symboli .....	99
Podsumowanie .....	100

## 6

<b>ŁAMANIE SZYFRU CEZARA ZA POMOCĄ ATAKU BRUTE FORCE .....</b>	<b>103</b>
Kod źródłowy programu wykorzystującego szyfr odwrotny .....	104
Przykładowe uruchomienie programu .....	105
Definiowanie zmiennych .....	106
Iteracja z użyciem funkcji range() .....	106
Deszyfrowanie wiadomości .....	108
Stosowanie formatowania ciągu tekstowego do wyświetlenia klucza i deszyfrowanej wiadomości .....	109
Podsumowanie .....	110

## 7

<b>SZYFROWANIE ZA POMOCĄ SZYFRU PRZESTAWIENIOWEGO .....</b>	<b>113</b>
Sposób działania szyfru przestawieniowego .....	113
Ręczne szyfrowanie wiadomości .....	114
Tworzenie programu szyfrującego .....	116
Kod źródłowy programu wykorzystującego szyfr kolumnowy .....	117
Przykładowe uruchomienie programu .....	118
Samodzielne definiowanie funkcji za pomocą polecenia def .....	118
Definiowanie funkcji pobierającej argumenty .....	119
Zmiana parametru istniejącego tylko wewnątrz funkcji .....	120
Definiowanie funkcji main() .....	121
Przekazywanie klucza i wiadomości jako argumentów .....	122
Typ danych listy .....	123
Ponowne przypisywanie elementów na liście .....	124
Lista list .....	125
Stosowanie funkcji len() i operatora in z listą .....	126
Konkatenacja listy i replikacja za pomocą operatorów + i * .....	127
Algorytm szyfrowania przestawieniowego .....	127
Rozszerzone operatory przypisania .....	128
Iteracja currentIndex przez wiadomość .....	129
Metoda join() .....	131
Wartość zwrotna i polecenie return .....	132
Przykład polecenia return .....	132
Zwrot szyfrogramu .....	133

Zmienna <code>__name__</code> .....	133
Podsumowanie .....	134
<b>8</b>	
<b>DESZYFROWANIE WIADOMOŚCI CHRONIONEJ SZYFREM PRZESTAWIENIOWYM .....</b>	<b>137</b>
Łamanie szyfru przestawieniowego za pomocą kartki i ołówka .....	138
Kod źródłowy programu deszyfrującego wiadomość chronioną szyfrem przestawieniowym .....	139
Przykładowe uruchomienie programu .....	141
Importowanie modułów i definiowanie funkcji <code>main()</code> .....	141
Deszyfrowanie wiadomości za pomocą klucza .....	142
Funkcje <code>round()</code> , <code>math.ceil()</code> i <code>math.floor()</code> .....	142
Funkcja <code>decryptMessage()</code> .....	143
Operatory boolowskie .....	145
Dostosowywanie wartości zmiennych <code>column</code> i <code>row</code> .....	148
Wywoływanie funkcji <code>main()</code> .....	150
Podsumowanie .....	150
<b>9</b>	
<b>TWORZENIE PROGRAMU DO TESTOWANIA INNYCH PROGRAMÓW .....</b>	<b>153</b>
Kod źródłowy programu do testowania innych programów .....	154
Przykładowe uruchomienie programu .....	155
Importowanie modułów .....	156
Generowanie liczb pseudolosowych .....	156
Tworzenie losowo wybranego ciągu tekstowego .....	158
Powielanie ciągu tekstowego losowo wybraną liczbę razy .....	158
Zmienna listy używa odwołania .....	159
Przekazywanie odwołania .....	162
Stosowanie funkcji <code>copy.deepcopy()</code> do powielenia listy .....	162
Funkcja <code>random.shuffle()</code> .....	163
Losowe mieszanie ciągu tekstowego .....	163
Testowanie poszczególnych wiadomości .....	164
Sprawdzanie poprawności szyfrowania i zakończenie programu .....	165
Wywoływanie funkcji <code>main()</code> .....	166
Testowanie programu .....	166
Podsumowanie .....	167
<b>10</b>	
<b>SZYFROWANIE I DESZYFROWANIE PLIKÓW .....</b>	<b>169</b>
Pliki zwykłego tekstu .....	170
Kod źródłowy programu wykorzystującego szyfr przestawieniowy do szyfrowania pliku .....	170
Przykładowe uruchomienie programu .....	171
Praca z plikami .....	172
Otwieranie pliku .....	172
Zapisywanie i zamykanie pliku .....	173
Odczyt danych z pliku .....	174
Funkcja <code>main()</code> programu .....	175
Sprawdzanie istnienia pliku .....	175
Funkcja <code>os.path.exists()</code> .....	176
Sprawdzanie za pomocą funkcji <code>os.path.exists()</code> istnienia pliku danych wejściowych .....	176

Stosowanie metod ciągu tekstowego do zapewnienia większej elastyczności danych wejściowych .....	177
Metody ciągu tekstowego upper(), lower() i title() .....	177
Metody ciągu tekstowego startswith() i endswith() .....	177
Stosowanie metod ciągu tekstowego w programie .....	178
Odczyt pliku danych wejściowych .....	179
Pomiar czasu operacji szyfrowania i deszyfrowania .....	179
Moduł time i funkcja time.time() .....	179
Stosowanie funkcji time.time() w programie .....	180
Zapis danych wyjściowych do pliku .....	181
Wywoływanie funkcji main() .....	181
Podsumowanie .....	182

## 11

<b>PROGRAMOWE WYKRYWANIE JĘZYKA ANGIELSKIEGO .....</b>	<b>183</b>
Jak komputer może zrozumieć język angielski? .....	184
Kod źródłowy modułu do wykrywania języka angielskiego .....	186
Przykładowe uruchomienie programu .....	187
Polecenia i definiowanie stałych .....	187
Typ danych w postaci słownika .....	188
Różnice między słownikiem i listą .....	189
Dodawanie lub modyfikowanie elementów słownika .....	190
Stosowanie funkcji len() ze słownikiem .....	191
Stosowanie operatora in ze słownikiem .....	191
Wyszukiwanie elementów w słowniku odbywa się szybciej niż na liście .....	192
Stosowanie pętli for w słowniku .....	192
Implementacja pliku słownika .....	193
Metoda split() .....	193
Podział słownika na poszczególne słowa .....	194
Zwrot danych słownika .....	194
Zliczanie liczby słów angielskich w wiadomości .....	195
Błąd dzielenia przez zero .....	196
Zliczanie dopasowań słów w języku angielskim .....	196
Funkcje float(), int() i str() oraz dzielenie całkowite .....	197
Określanie proporcji angielskich słów w wiadomości .....	198
Usuwanie znaków innych niż litery .....	198
Metoda append() typu listy .....	199
Tworzenie ciągu tekstowego liter .....	200
Wykrywanie słów angielskich .....	200
Stosowanie argumentów domyślnych .....	200
Obliczanie wartości procentowych .....	201
Podsumowanie .....	203

## 12

<b>ŁAMANIE SZYFRU PRZESTAWIENIOWEGO .....</b>	<b>205</b>
Kod źródłowy programu umożliwiającego złamanie szyfru przestawieniowego .....	206
Przykładowe uruchomienie programu .....	207
Importowanie modułów .....	208
Wielowierszowy ciąg tekstowy ujęty w potrójny cudzysłów .....	208
Wyświetlanie wyniku deszyfrowania wiadomości .....	209

Pobranie deszyfrowanej wiadomości .....	210
Metoda strip() ciągu tekstowego .....	212
Stosowanie metody strip() ciągu tekstowego .....	213
Nieudana próba deszyfrowania wiadomości .....	213
Wywoływanie funkcji main() .....	214
Podsumowanie .....	214

### 13

<b>MODUŁ ARYTMETYKI MODULARNEJ DLA SZYFRU AFINICZNEGO .....</b>	<b>215</b>
Arytmetyka modularna .....	216
Operator reszty z dzielenia .....	217
Wyszukiwanie dzielników do obliczenia największego wspólnego dzielnika .....	218
Przypisanie wielokrotne .....	220
Algorytm Euklidesa do wyszukiwania największego wspólnego dzielnika .....	221
Sposób działania szyfrów multiplikatywnego i afinicznego .....	222
Wybór poprawnego klucza multiplikatywnego .....	223
Szyfrowanie z użyciem szyfru afinicznego .....	224
Deszyfrowanie szyfru afinicznego .....	225
Określanie odwrotności modularnej .....	226
Operator dzielenia całkowitego .....	226
Kod źródłowy modułu cryptomath .....	227
Podsumowanie .....	228

### 14

<b>PROGRAMOWANIE SZYFRU AFINICZNEGO .....</b>	<b>231</b>
Kod źródłowy programu wykorzystującego szyfr afiniczny .....	232
Przykładowe uruchomienie programu .....	233
Importowanie modułów i stałych oraz definiowanie funkcji main() .....	234
Generowanie i weryfikowanie kluczy .....	236
Typ danych w postaci krotki .....	236
Sprawdzanie pod kątem stałych kluczy .....	237
Ile kluczy może mieć szyfr afiniczny? .....	238
Tworzenie funkcji szyfrującej .....	240
Tworzenie funkcji deszyfrującej .....	241
Generowanie losowych kluczy .....	242
Wywoływanie funkcji main() .....	243
Podsumowanie .....	244

### 15

<b>ŁAMANIE SZYFRU AFINICZNEGO .....</b>	<b>245</b>
Kod źródłowy programu umożliwiającego złamanie szyfru afinicznego .....	245
Przykładowe uruchomienie programu .....	247
Importowanie modułów i stałych oraz definiowanie funkcji main() .....	248
Funkcja odpowiedzialna za złamanie szyfru afinicznego .....	249
Operator wykładniczy .....	249
Obliczanie całkowitej liczby kluczy, których można użyć .....	250
Polecenie continue .....	251
Stosowanie polecenia continue do pominięcia kodu .....	252
Wywoływanie funkcji main() .....	253
Podsumowanie .....	254

<b>16</b>	
<b>PROGRAMOWANIE PROSTEGO SZYFRU PODSTAWIENIOWEGO .....</b>	<b>255</b>
Jak działa prosty szyfr podstawieniowy? .....	256
Kod źródłowy programu wykorzystującego szyfr podstawieniowy .....	257
Przykładowe uruchomienie programu .....	259
Importowanie modułów i stałych oraz definiowanie funkcji main() .....	259
Metoda sort() listy .....	261
Funkcje opakowujące .....	262
Funkcja translateMessage() .....	263
Metody isupper() i islower() ciągu tekstowego .....	265
Zachowywanie wielkości liter dzięki metodzie isupper() .....	266
Generowanie losowego klucza .....	267
Wywoływanie funkcji main() .....	268
Podsumowanie .....	268
<b>17</b>	
<b>ŁAMANIE PROSTEGO SZYFRU PODSTAWIENIOWEGO .....</b>	<b>271</b>
Stosowanie wzorca słowa do deszyfrowania .....	272
Znajdowanie wzorca słowa .....	272
Wyszukiwanie potencjalnych liter odszyfrowujących .....	273
Omówienie procesu łamania szyfru .....	275
Moduł wzorca słowa .....	275
Kod źródłowy programu wykorzystującego szyfr podstawieniowy .....	276
Przykładowe uruchomienie programu .....	280
Importowanie modułów i stałych .....	280
Wyszukiwanie znaków za pomocą wyrażeń regularnych .....	281
Konfigurowanie funkcji main() .....	281
Wyświetlanie użytkownikowi wyniku operacji łamania szyfru .....	282
Tworzenie mapowania szyfrogramu .....	283
Tworzenie pustego mapowania .....	283
Dodawanie liter do mapowania .....	283
Łączenie dwóch mapowań .....	285
W jaki sposób działają funkcje pomocnicze mapowania liter? .....	286
Wyszukiwanie zdeszyfrowanych liter w mapowaniu .....	290
Testowanie funkcji removeSolvedLettersFromMapping() .....	292
Funkcja hackSimpleSub() .....	292
Metoda replace() ciągu tekstowego .....	294
Deszyfrowanie wiadomości .....	295
Deszyfrowanie w powłoce interaktywnej .....	296
Wywoływanie funkcji main() .....	297
Podsumowanie .....	298
<b>18</b>	
<b>PROGRAMOWANIE SZYFRU VIGENÈRE'A .....</b>	<b>299</b>
Stosowanie wielu liter kluczy w szyfrze Vigenère'a .....	300
Dłuższe klucze szyfru Vigenère'a są znacznie bezpieczniejsze .....	302
Wybór klucza uniemożliwiającego atak słownikowy .....	303
Kod źródłowy programu wykorzystującego szyfr Vigenère'a .....	303
Przykładowe uruchomienie programu .....	305
Importowanie modułów i stałych oraz definiowanie funkcji main() .....	305
Tworzenie ciągu tekstowego za pomocą procesu dołączania do listy .....	306

Szyfrowanie i deszyfrowanie wiadomości .....	307
Wywoływanie funkcji main() .....	310
Podsumowanie .....	310

## 19

<b>ANALIZA CZĘSTOTLIWOŚCI .....</b>	<b>313</b>
Analiza częstotliwości występowania liter w tekście .....	314
Dopasowywanie częstotliwości występowania liter .....	316
Obliczanie wyniku dopasowania częstotliwości dla prostego szyfru podstawieniowego .....	316
Obliczanie wyniku dopasowania częstotliwości dla prostego szyfru przestawieniowego .....	317
Stosowanie analizy częstotliwości do złamania szyfru Vigenère'a .....	318
Kod źródłowy programu obliczającego wynik dopasowania częstotliwości .....	319
Przechowywanie liter w kolejności ETAOIN .....	321
Zliczanie liter w wiadomości .....	321
Pobieranie pierwszego elementu składowego krotki .....	323
Układanie liter według częstotliwości ich występowania w wiadomości .....	323
Zliczanie liter za pomocą funkcji getLetterCount() .....	324
Tworzenie słownika częstotliwości wystąpień i listy liter .....	324
Sortowanie liter w odwrotnej kolejności ETAOIN .....	325
Sortowanie list słownika według częstotliwości występowania .....	330
Tworzenie listy sortowanych liter .....	332
Obliczanie wyniku dopasowania częstotliwości dla wiadomości .....	332
Podsumowanie .....	334

## 20

<b>ŁAMANIE SZYFRU VIGENÈRE'A .....</b>	<b>335</b>
Atak słownikowy w celu złamania szyfru Vigenère'a metodą brute force .....	336
Kod źródłowy programu umożliwiającego złamanie szyfru Vigenère'a za pomocą ataku słownikowego .....	336
Przykładowe uruchomienie programu .....	337
Informacje o programie do łamania szyfru Vigenère'a za pomocą ataku słownikowego .....	337
Stosowanie metody Kasiskiego do ustalenia długości klucza .....	338
Odszukanie powtarzających się sekwencji .....	338
Pobieranie dzielników liczb określających odstęp .....	339
Pobieranie każdej n-tej litery ciągu tekstowego .....	341
Stosowanie analizy częstotliwości do złamania poszczególnych podkluczy .....	342
Przeprowadzanie ataku brute force na możliwe klucze .....	344
Kod źródłowy programu umożliwiającego złamanie szyfru Vigenère'a .....	344
Przykładowe uruchomienie programu .....	349
Importowanie modułów i definiowanie funkcji main() .....	350
Wyszukiwanie powtarzających się sekwencji .....	351
Obliczanie dzielników odstępów .....	354
Usuwanie duplikatów za pomocą funkcji set() .....	355
Usuwanie powtarzających się dzielników i sortowanie listy .....	355
Wyszukiwanie najczęściej występujących dzielników .....	356
Określanie prawdopodobnej długości klucza .....	358
Metoda listy extend() .....	358
Rozszerzanie słownika repeatedSeqSpacings .....	359
Pobieranie dzielników z factorsByCount .....	360
Pobieranie liter szyfrowanych za pomocą tego samego podklucza .....	360

Próba deszyfrowania z użyciem potencjalnych długości klucza .....	362
Argument w postaci słowa kluczowego key funkcji print() .....	364
Uruchamianie programu w trybie cichym lub wyświetlania informacji użytkownikowi .....	365
Wyszukiwanie możliwych kombinacji podkluczy .....	365
Wyświetlanie deszyfrowanego tekstu z użyciem właściwej wielkości liter .....	369
Zwrot deszyfrowanej wiadomości .....	370
Opuszczanie pętli po znalezieniu potencjalnego klucza .....	371
Atak brute force na wszystkie długości klucza .....	371
Wywoływanie funkcji main() .....	372
Modyfikowanie stałych programu .....	373
Podsumowanie .....	373

<b>21</b>	
<b>SZYFR Z KLUCZEM JEDNORAZOWYM .....</b>	<b>375</b>
Niemożliwy do złamania szyfr z kluczem jednorazowym .....	376
Tworzenie klucza o długości odpowiadającej długości wiadomości .....	376
Zapewnianie prawdziwej losowości klucza .....	378
Dlaczego klucza jednorazowego można użyć tylko raz? .....	379
Dlaczego dwukrotnie użyty klucz jednorazowy to szyfr Vigenère'a? .....	379
Podsumowanie .....	380

<b>22</b>	
<b>WYSZUKIWANIE I GENEROWANIE LICZB PIERWSZYCH .....</b>	<b>381</b>
Co to jest liczba pierwsza? .....	382
Kod źródłowy modułu liczb pierwszych .....	384
Przykładowe uruchomienie modułu .....	386
Sposób działania algorytmu próbnego dzielenia .....	386
Implementacja algorytmu próbnego dzielenia .....	388
Sito Eratostenesa .....	389
Generowanie liczb pierwszych za pomocą sita Eratostenesa .....	391
Algorytm pierwszości Rabina-Millera .....	392
Wyszukiwanie ogromnych liczb pierwszych .....	393
Generowanie ogromnych liczb pierwszych .....	395
Podsumowanie .....	395

<b>23</b>	
<b>GENEROWANIE KLUCZY DLA SZYFRU KLUCZA PUBLICZNEGO .....</b>	<b>397</b>
Kryptografia klucza publicznego .....	398
Problem z uwierzytelnieniem .....	400
Podpis cyfrowy .....	400
Uważaj na atak MITM .....	401
Etapy generowania kluczy publicznego i prywatnego .....	402
Kod źródłowy programu generującego klucze kryptografii klucza publicznego .....	403
Przykładowe uruchomienie programu .....	404
Tworzenie funkcji main() .....	406
Generowanie kluczy za pomocą funkcji generateKey() .....	406
Obliczanie wartości e .....	407
Obliczanie wartości d .....	407
Zwracanie kluczy .....	408
Tworzenie plików kluczy za pomocą funkcji makeKeyFiles() .....	408

Wywoływanie funkcji main() .....	410
Hybrydowe systemy kryptograficzne .....	411
Podsumowanie .....	411

## 24

<b>PROGRAMOWANIE SZYFRU KLUCZA PUBLICZNEGO .....</b>	<b>413</b>
Jak działa kryptografia klucza publicznego? .....	414
Tworzenie bloku .....	414
Konwersja ciągu tekstowego na blok .....	415
Matematyka szyfrowania i deszyfrowania za pomocą kryptografii klucza publicznego .....	416
Konwersja bloku na ciąg tekstowy .....	418
Dlaczego nie można złamać szyfru wykorzystującego kryptografię klucza publicznego? .....	420
Kod źródłowy programu wykorzystującego kryptografię klucza publicznego .....	421
Przykładowe uruchomienie programu .....	425
Konfiguracja programu .....	426
Wybór trybu pracy programu .....	426
Konwersja ciągu tekstowego na bloki za pomocą funkcji getBlocksFromText() .....	428
Funkcje min() i max() .....	428
Przechowywanie bloków w blockInt .....	429
Stosowanie funkcji getTextFromBlocks() do deszyfrowania wiadomości .....	431
Stosowanie metody insert() listy .....	432
łączenie listy message i tworzenie na jej podstawie jednego ciągu tekstowego .....	432
Tworzenie funkcji encryptMessage() .....	433
Tworzenie funkcji decryptMessage() .....	433
Odczytywanie kluczy publicznego i prywatnego z ich plików .....	434
Zapisywanie szyfrogramu do pliku .....	435
Deszyfrowanie danych z pliku .....	437
Wywoływanie funkcji main() .....	439
Podsumowanie .....	439

## A

<b>DEBUGOWANIE KODU PYTHONA .....</b>	<b>441</b>
Na czym polega działanie debugera? .....	441
Usuwanie błędów z programu wykorzystującego szyfr odwrotny .....	443
Definiowanie punktu przerwania .....	445
Podsumowanie .....	447

## B

<b>ODPOWIEDZI DO ĆWICZEŃ .....</b>	<b>449</b>
Rozdział 1. ....	449
Rozdział 2. ....	450
Rozdział 3. ....	451
Rozdział 4. ....	452
Rozdział 5. ....	453
Rozdział 6. ....	454
Rozdział 7. ....	455
Rozdział 8. ....	457
Rozdział 9. ....	459
Rozdział 10. ....	459
Rozdział 11. ....	460

Rozdział 12. ....	461
Rozdział 13. ....	462
Rozdział 14. ....	462
Rozdział 15. ....	463
Rozdział 16. ....	463
Rozdział 17. ....	464
Rozdział 18. ....	464
Rozdział 19. ....	465
Rozdział 20. ....	465
Rozdział 21. ....	466
Rozdział 22. ....	466
Rozdział 23. ....	466