

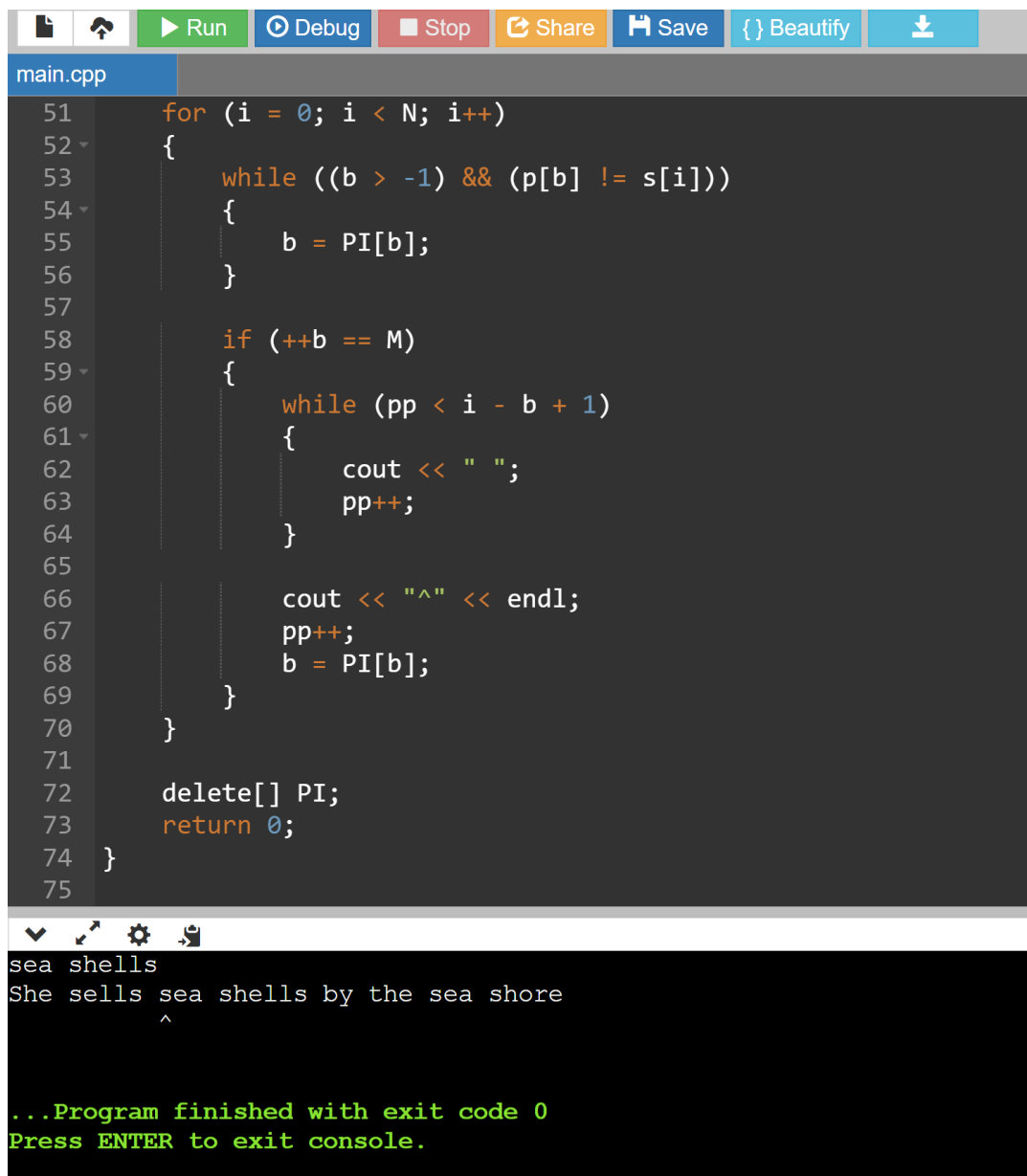
Przetwarzanie informacji multimedialnej

Sprawozdanie lab 1

Michał Dandyk

nr albumu: 125294

Zadanie 1.1



```
main.cpp
51  for (i = 0; i < N; i++)
52  {
53      while ((b > -1) && (p[b] != s[i]))
54      {
55          b = PI[b];
56      }
57
58      if (++b == M)
59      {
60          while (pp < i - b + 1)
61          {
62              cout << " ";
63              pp++;
64          }
65
66          cout << "^" << endl;
67          pp++;
68          b = PI[b];
69      }
70  }
71
72  delete[] PI;
73  return 0;
74 }
75
```

sea shells
She sells sea shells by the sea shore
^

...Program finished with exit code 0
Press ENTER to exit console.

Zadanie 1.2

Algorytm	Tekst	Wzorzec	Wynik z konsoli
Naiwny	She sells sea shells by the sea shore	sea shells	<pre> sea shells She sells sea shells by the sea shore ^ </pre>
	A quick brown fox jumped over a lazy dog	fox	<pre> fox A quick brown fox jumped over a lazy dog ^ </pre>
	Yesterday all my troubles seemed so far away	away	<pre> away Yesterday all my troubles seemed so far away ^ </pre>
	Abra cabra cadabra	cabra	<pre> cabra Abra cabra cadabra ^ </pre>
	Lorem ipsum dolor sit amet, consectetur adipiscing elit	consectetur	<pre> consectetur Lorem ipsum dolor sit amet, consectetur adipiscing elit ^ </pre>
Morrisa-Pratta	She sells sea shells by the sea shore	sea shells	<pre> sea shells She sells sea shells by the sea shore ^ </pre>
	A quick brown fox jumped over a lazy dog	fox	<pre> fox A quick brown fox jumped over a lazy dog ^ </pre>
	Yesterday all my troubles seemed so far away	away	<pre> away Yesterday all my troubles seemed so far away ^ </pre>
	Abra cabra cadabra	cabra	<pre> cabra Abra cabra cadabra ^ </pre>
	Lorem ipsum dolor sit amet, consectetur adipiscing elit	consectetur	<pre> consectetur Lorem ipsum dolor sit amet, consectetur adipiscing elit ^ </pre>
Knutha-Morrisa-Pratta	She sells sea shells by the sea shore	sea shells	<pre> sea shells She sells sea shells by the sea shore ^ </pre>
	A quick brown fox jumped over a lazy dog	fox	<pre> fox A quick brown fox jumped over a lazy dog ^ </pre>
	Yesterday all my troubles seemed so far away	away	<pre> away Yesterday all my troubles seemed so far away ^ </pre>
	Abra cabra cadabra	cabra	<pre> cabra Abra cabra cadabra ^ </pre>
	Lorem ipsum dolor sit amet, consectetur adipiscing elit	consectetur	<pre> consectetur Lorem ipsum dolor sit amet, consectetur adipiscing elit ^ </pre>

Boyera-Moora	She sells sea shells by the sea shore	sea shells	<pre> sea shells She sells sea shells by the sea shore ^ </pre>
	A quick brown fox jumped over a lazy dog	fox	<pre> fox A quick brown fox jumped over a lazy dog ^ </pre>
	Yesterday all my troubles seemed so far away	away	<pre> away Yesterday all my troubles seemed so far away ^ </pre>
	Abra cabra cadabra	cabra	<pre> cabra Abra cabra cadabra ^ </pre>
	Lorem ipsum dolor sit amet, consectetur adipiscing elit	consectetur	<pre> consectetur Lorem ipsum dolor sit amet, consectetur adipiscing elit ^ </pre>

Zadanie 1.3

Algorytm Morrisa-Pratta jest jednym z popularnych algorytmów wyszukiwania wzorca w tekście. Jest często wybierany ze względu na swoją ogólną wydajność i efektywność.

1. Złożoność obliczeniowa:

- **Najlepsza przypadkowa złożoność:** $O(m + n)$, gdzie m to długość wzorca, a n to długość tekstu. Najlepszy przypadek występuje, gdy wzorec i tekst są identyczne.
- **Przeciętna i najgorsza złożoność:** $O(m + n)$, co oznacza, że algorytm zachowuje się efektywnie w praktyce dla różnych przypadków.

2. Pozytywne cechy:

- **Efektywność:** Algorytm jest efektywny i szybki w praktyce, zwłaszcza dla średnich i dużych ilości danych.
- **Stabilność:** Jako algorytm deterministyczny zawsze zwraca poprawne wyniki.

3. Negatywne cechy:

- **Wrażliwość na powtarzające się podciągi:** Jeśli w tekście występują powtarzające się podciągi, algorytm może być mniej efektywny.
- **Wymagania pamięciowe:** Algorytm Morrisa-Pratta wymaga dodatkowej pamięci dla tablicy PI (prefix function), co może być problematyczne w przypadku bardzo dużych danych.

Zadanie 1.4

```
1 import random
2 import string
3
4 # generujemy tańcuch s
5 s = "Lorem ipsum dolor sit amet, consectetur adipiscing elit"
6 N = len(s)
7
8 # generujemy wzorzec
9 p = "consectetur"
10 M = len(p)
11
12 PI = [0] * (M + 1)
13
14 # dla wzorca obliczamy tablicę PI [ ]
15 PI[0] = b = -1
16 for i in range(1, M + 1):
17     while b > -1 and p[b] != p[i - 1]:
18         b = PI[b]
19     b += 1
20     PI[i] = b
21
22 # wypisujemy wzorzec
23 print(p)
24
25 # wypisujemy tańcuch s
26 print(s)
27
28 # poszukujemy pozycji wzorca w tańcuchu
29 pp = b = 0
30 for i in range(N):
31     while b > -1 and p[b] != s[i]:
32         b = PI[b]
33
34     if b + 1 == M:
35         while pp < i - b:
36             print(" ", end="")
37             pp += 1
38         print("^", end="")
39         pp += 1
40         b = PI[b]
41
42 print()
43
```

Zadanie 2.1

```
main.cpp
5
6 #include <iostream>
7 #include <string>
8
9 using namespace std;
10
11 // definicje elementów Enigmy
12
13 const string pierscien_szyfr [ 5 ] = {"EKMFLGDQVZNTOWYHXUSPAIBRCJ",
14                                       "AJDKSIRUXBLHWTMCQGZNPYFVOE",
15                                       "BDFHJLCPRTXVZNYEIWGAKMUSQO",
16                                       "ESOVPPZJAYQUIRHXLNFTGKDCMWB",
17                                       "VZBRGITYUPSDNHLXAWMJQOFECK"};
18 const string przeniesienie = "RFWKA";
19 const string beben_odwr = "YRUHQSLDPXNGOKMIEBFZCWVJAT";
20
21 int main( )
22 {
23     int pierscien [ 3 ], i, j, k, n, c;
24     bool ruch;
25     string szyfr, s, lacznica;
26
27     // odczytujemy konfigurację pierścieni szyfrujących
28
29     cin >> n;
30
31     for( i = 2; i >= 0; i-- )
32     {
33         pierscien [ i ] = ( n % 10 ) - 1; // numer pierścienia na i-tej pozycji
34         n /= 10;
35     }
36
37     // odczytujemy położenia początkowe pierścieni
38
39     123
40     abba
41     csd
42     ala ma kota
43
44     ...Program finished with exit code 0
45     Press ENTER to exit console.
```

Zadanie 2.2

Szyfr Cezara	Szyfr testowy	VCBIU WHVWRZB
	Ala ma kota	VCBIU FHCDUD
Szyfrowanie z pseudolosowym dostępem	123/Szyfr testowy	rzsSetofwtyy
	12345/Ala ma kota	Aaamka tol
Szyfr przestawieniowy	Szyfr testowy	zsfy rettswoy
	Ala ma kota	la aamk toa
Szyfr Enigmy	234/ABC/ABCDEF/Szyfr testowy	LYVTTPOIBDBVN
	123/ABCD/ABCD/Ala ma kota	WWXLURLTVWW
Szyfr RSA	3-5608/943-912/12345	474
	4-2345/353-912/1234567	81

Zadanie 2.3

Szyfr Enigmy miał swoje zalety w czasie, gdy został wprowadzony, ale z biegiem czasu stał się podatny na nowe metody kryptograficzne. To jedno z kluczowych wydarzeń w historii kryptografii, które pomogło w zrozumieniu potrzeby ciągłego doskonalenia metod szyfrowania.

1. Złożoność Obliczeniowa:

- **Pozytywna strona:** W momencie swojego powstania, szyfr Enigmy był uważany za niezwykle trudny do złamania. Miał potężną przestrzeń klucza, co utrudniało atak brutalny.
- **Negatywna strona:** W miarę postępu technologicznego, enkryptory Enigmy stały się coraz bardziej podatne na ataki kryptograficzne. Dzisiaj, ze względu na rozwój komputerów, zaszyfrowane komunikaty mogą być łatwiej złamane.

2. Pozytywne Cechy:

- **Mechanizm Rotorowy:** Zastosowanie rotujących wirników (rotorów) dodawało skomplikowania szyfrowania, co utrudniało atakującym odczytanie wiadomości.
- **Regularna Zmiana Ustawień:** Systematyczna zmiana ustawień Enigmy sprawiała, że rozkodowanie jednego komunikatu nie ułatwiała odszyfrowania kolejnych.

3. Negatywne Cechy:

- **Brak Centralnej Organizacji:** Brak centralnej jednostki kontrolującej wszystkie ustawienia maszyn Enigmy sprawiał, że atakującym wystarczyło

złamać jedno stanowisko, aby zdekodować wiadomości dla całego danego okresu czasu.

- **Zależność od Operatorów:** Szyfr Enigmy mógł być podatny na ludzki błąd, a także na fakt, że operatorzy mogli stosować niezalecane praktyki, co zwiększało ryzyko złamania.

Zadanie 2.4

```
1 def ceasar_cipher(text):
2     encrypted_text = ""
3
4     # zamiana małych liter na duże, oraz szyfrowanie kodem cezara
5     for char in text:
6         char = char.upper()
7         if 'A' <= char <= 'Z':
8             encrypted_text += chr((ord(char) - 62) % 26 + 65)
9         else:
10            encrypted_text += char
11
12    return encrypted_text
13
14
15 if __name__ == "__main__":
16     # wpisywanie przez użytkownika tekstu do zaszyfrowania
17     s = input("Podaj tekst do zaszyfrowania: ")
18
19     encrypted_text = ceasar_cipher(s)
20
21     # wyświetlenie szyfru
22     print(encrypted_text)
23
```

Podaj tekst do zaszyfrowania: ala ma kota
OOD PD NRWD