

Sprawozdanie Ćwiczenie 1 – Przetwarzanie informacji multimedialnej

Jakub Wasik – 122047

Zadanie 1.1:

The screenshot shows the OnlineGDB web interface. The browser address bar displays `https://www.onlinegdb.com/online_c++_compiler`. The left sidebar contains navigation links: IDE, My Projects, Classroom (marked 'new'), Learn Programming, Programming Questions, Sign Up, and Login. The main editor area shows a C++ file named `main.cpp` with the following code:

```
47 cout << text << endl;
48
49 int pp = 0;
50 int i = 0;
51
52 while (i <= N - M) {
53     int j = M - 1;
54     while ((j > -1) && (pattern[j] == text[i + j])) j--;
55     if (j == -1) {
56         while (pp < i) {
57             cout << " ";
58             pp++;
59         }
60         cout << "^";
61         pp++;
62         i += BMNext[0];
63     } else {
64         i += max(BMNext[j + 1], j - Last[text[i + j]]);
65     }
66 }
67
68 cout << endl;
69 }
70
71 int main() {
72     string s = "A big black bug bit a big black bear";
73     string p = "big black";
74
75     searchPattern(s, p);
76
77     return 0;
78 }
79
```

Below the code editor is a console window titled "input" showing the program's output:

```
big black
A big black bug bit a big black bear
^          ^
...Program finished with exit code 0
Press ENTER to exit console.
```

The footer of the interface includes links for About, FAQ, Blog, Terms of Use, Contact Us, GDB Tutorial, Credits, and Privacy, along with the copyright notice © 2016 - 2024 GDB Online.

Zadanie 1.2:

Algorytm	Tekst	Wzorzec	Wynik z konsoli
Naiwny	A big black bug bit a big black bear	big black	big black A big black bug bit a big black bear ^ ^
Naiwny	A flea and a fly flew up in a flue	fly flew	fly flew A flea and a fly flew up in a flue ^
Naiwny	A happy hippo hopped and hiccupped	hippo	hippo A happy hippo hopped and hiccupped ^
Naiwny	A noisy noise annoys an oyster	noise	noise A noisy noise annoys an oyster ^
Naiwny	A proper copper coffee pot	copper	copper A proper copper coffee pot ^
Morrisa-Pratta	A big black bug bit a big black bear	big black	big black A big black bug bit a big black bear ^ ^
Morrisa-Pratta	A flea and a fly flew up in a flue	fly flew	fly flew A flea and a fly flew up in a flue ^
Morrisa-Pratta	A happy hippo hopped and hiccupped	hippo	hippo A happy hippo hopped and hiccupped ^
Morrisa-Pratta	A noisy noise annoys an oyster	noise	noise A noisy noise annoys an oyster ^
Morrisa-Pratta	A proper copper coffee pot	copper	copper A proper copper coffee pot ^
Knutha-Morrisa-Pratta	A big black bug bit a big black bear	big black	big black A big black bug bit a big black bear ^ ^
Knutha-Morrisa-Pratta	A flea and a fly flew up in a flue	fly flew	fly flew A flea and a fly flew up in a flue ^
Knutha-Morrisa-Pratta	A happy hippo hopped and hiccupped	hippo	hippo A happy hippo hopped and hiccupped ^
Knutha-Morrisa-Pratta	A noisy noise annoys an oyster	noise	noise A noisy noise annoys an oyster ^
Knutha-Morrisa-Pratta	A proper copper coffee pot	copper	copper A proper copper coffee pot ^
Boyera-Moora	A big black bug bit a big black bear	big black	big black A big black bug bit a big black bear ^ ^
Boyera-Moora	A flea and a fly flew up in a flue	fly flew	fly flew A flea and a fly flew up in a flue ^

Boyera-Moora	A happy hippo hopped and hiccupped	hippo	hippo A happy hippo hopped and hiccupped ^
Boyera-Moora	A noisy noise annoys an oyster	noise	noise A noisy noise annoys an oyster ^
Boyera-Moora	A proper copper coffee pot	copper	copper A proper copper coffee pot ^

Zadanie 1.3:

Algorytm naiwny jest prosty, uniwersalny i łatwy do zrozumienia, co sprawia, że jest dobrym wyborem dla małych danych i celów edukacyjnych. Jednak jego główną wadą jest niska wydajność w przypadku dużych zbiorów danych, brak mechanizmu pomijania i brak optymalizacji dla specyficznych struktur danych, co sprawia, że w praktyce bardziej zaawansowane algorytmy są częściej wybierane. Jest odpowiedni dla prostych zastosowań, małych danych i celów edukacyjnych dzięki swojej prostocie. W praktyce, zwłaszcza przy dużych zbiorach danych, bardziej zaawansowane algorytmy, takie jak Boyer-Moore czy Knuth-Morris-Pratt, są preferowane ze względu na swoją lepszą wydajność i specjalizowane heurystyki. Algorytm naiwny stanowi jednak punkt wyjścia do zrozumienia podstaw algorytmów wyszukiwania wzorca.

Zadanie 1.4:

```

main.py  [Icons] [Save] [Run]  Shell  [Clear]
1  tekst = "A flea and a fly flew up in a
    flue"
2  wzorzec = "fly flew"
3
4  print(wzorzec)
5  print(tekst)
6
7  for i in range(len(tekst) - len(wzorzec) +
    1):
8      podciag = tekst[i:i + len(wzorzec)]
9      print("^" if podciag == wzorzec else "
    ", end=" ")
10
11 print("\n\n")
  
```

```

fly flew
A flea and a fly flew up in a flue
    ^
>
  
```

Zadanie 2.1:

OnlineGDB beta
online compiler and debugger for c/c++

code. compile. run. debug. share.

IDE

My Projects

Classroom **new**

Learn Programming

Programming Questions

Sign Up

Login

main.cpp

```

1 #include <iostream>
2 #include <string>
3
4 using namespace std;
5
6 int main( )
7 {
8     string s;
9     int i;
10
11     // odczytujemy wiersz znaków
12
13     getline ( cin, s );
14
15     // zamieniamy małe litery na duże
16     // i kodujemy szyfrem cezara
17
18     for( i = 0; i < s.length( ); i++ )
19     {
20         s [ i ] = toupper ( s [ i ] );
21         if( ( s [ i ] >= 'A' ) && ( s [ i ] <= 'Z' ) ) s [ i ] = char ( 65 + ( s
22     }
23
24     // wypisujemy zaszyfrowany tekst
25
26     cout << s << endl << endl;
27     return 0;
28 }

```

input

Testowa fraza
WHVWRZD IUDCD

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

About • FAQ • Blog • Terms of Use • Contact Us • GDB Tutorial • Credits • Privacy
© 2016 - 2024 GDB Online

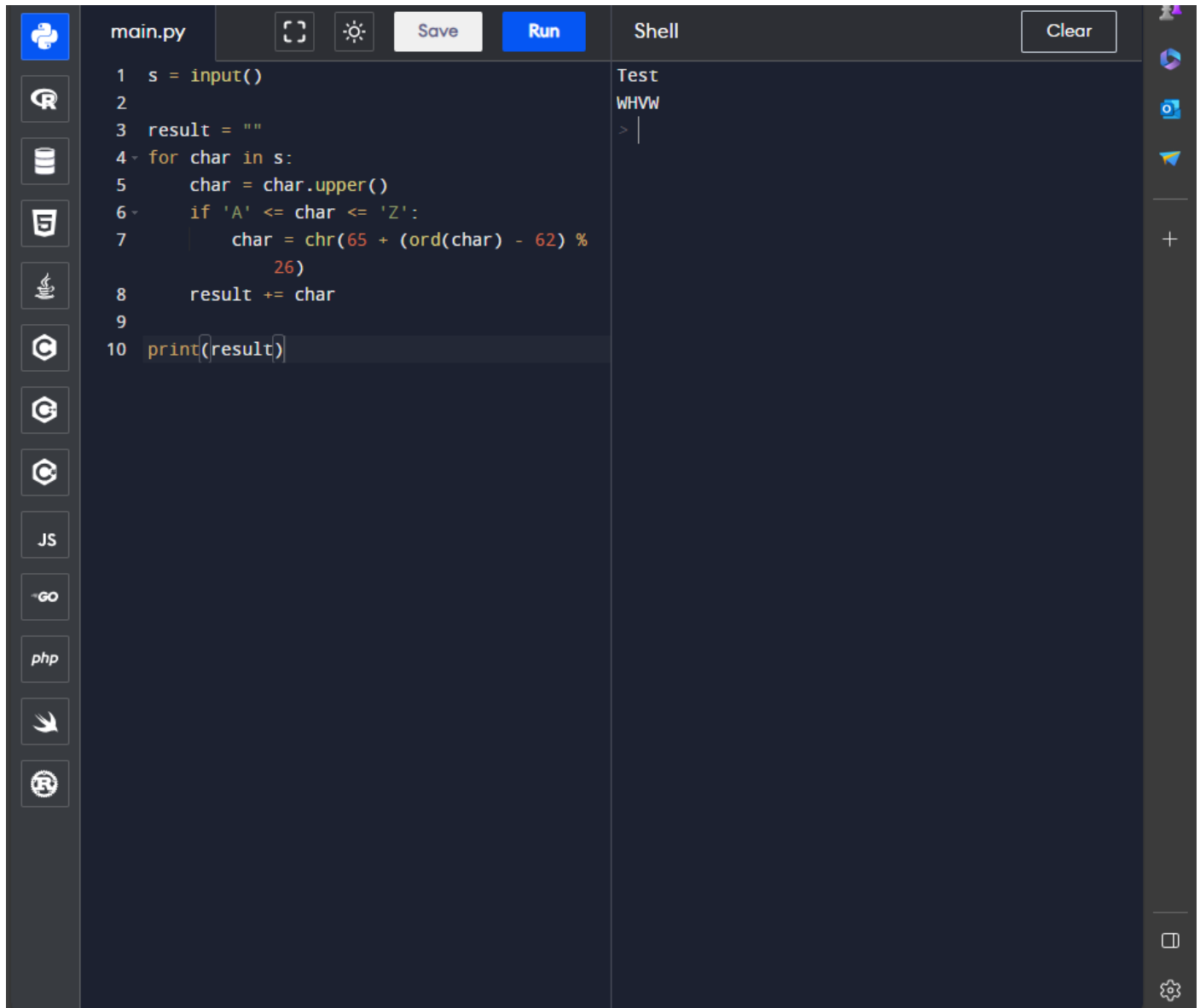
Zadanie 2.2:

Szyfr Cezara	Black Leopard	EODFN OHRSDUG
Szyfr Cezara	Silent Hill	VLOHQW KLOO
Szyfrowanie z pseudolosowym odstępem	1001 / Black Leopard	XCENW ZDAGUQZ
Szyfrowanie z pseudolosowym odstępem	1002 / Silent Hill	XSSKIN BHNI
Szyfr przestawieniowy	Black Leopard	lBca keLpora
Szyfr przestawieniowy	Silent Hill	iSeltH li
Szyfr Enigmy	123 / ABC / ABCD / Black Leopard	DWXXKAHRZVVWLP
Szyfr Enigmy	123 / ABC / ABCD / Silent Hill	FSMYHGLGBYD
Szyfr RSA	3 – 587 / 943 – 943 / 123456789	791
Szyfr RSA	3 – 587 / 943 – 943 / 987654321	878

Zadanie 2.3:

Szyfr Cezara to prosty szyfr przesunięcia, w którym każda litera tekstu jawnego jest zamieniana na literę przesuniętą o stałą wartość w alfabecie. Juliusz Cezar używał tego szyfru do komunikacji z dowódcami. Klucz szyfrujący to liczba określająca przesunięcie liter. Szyfr Cezara jest łatwy do zrozumienia, ale ze względu na swoją podatność na ataki brute force, jego użycie jest ograniczone, częściej wykorzystywany w celach edukacyjnych niż praktycznych.

Zadanie 2.4:



The image shows a code editor interface with a dark theme. On the left, there is a sidebar with various icons for different programming languages and tools. The main editor area displays a Python script named `main.py`. The script implements a Caesar cipher encryption function. It takes an input string `s`, converts it to uppercase, and then shifts each character by 26 positions in the alphabet. The output is printed to the console. To the right of the code editor, there is a 'Shell' window showing the execution of the script. The shell prompt is `>`, and the user has entered `Test` and `WHVW`. The output of the script is `Test` and `WHVW`.

```
main.py
1 s = input()
2
3 result = ""
4 for char in s:
5     char = char.upper()
6     if 'A' <= char <= 'Z':
7         char = chr(65 + (ord(char) - 62) %
8                     26)
9     result += char
10 print(result)
```

Shell

```
Test
WHVW
>
```