Marta Załuska

INFORMATYKA PRAKTYCZNA, III SEMESTR, I STOPIEŃ, STACJONARNE

PARADOKS URODZIN

Algorytmion (rok 2013, zad. 3)

Wstęp

Opis programu

W klasycznym ujęciu, rozwiązanie problemu urodzin polega na znalezieniu najmniejszej liczby osób takiej, żeby prawdopodobieństwo znalezienia dwóch osób obchodzących urodziny tego samego dnia było równe co najmniej 50%. Paradoks polega na tym, że ta liczba jest zaskakująco mała – potrzeba jedynie 23 osób, aby spełniony był ten warunek.

Program oblicza problem urodzin eksperymentalnie, metodą Monte Carlo – generuje daty urodzin, sprawdza ich zgodność z warunkami i znajduje najmniejszą zgodną liczbę osób.

Instrukcja obsługi

Interakcja z użytkownikiem odbywa się przez plik `menu.bat`. Po uruchomieniu skryptu, na ekranie użytkownika pojawia się proste menu z możliwością wyboru jednej z czterech opcji, wybieranej poprzez naciśnięcie cyfry jej odpowiadającej, a następnie klawisza Enter:

1. `Uruchom program`
2. `Wyświetl opis`
3. `Utwórz backup`
4. `Wyjście`

Po wybraniu jednej z opcji `1-3`, użytkownik może wrócić do głównego menu, naciskając dowolny klawisz. Wybranie opcji `Wyjście` zamyka okienko terminala.

Użytkownik może zmienić ilość testów, które program wykonuje dla każdej testowanej liczby osób, poprzez zmianę treści pliku `in.txt` w folderze `input`.

Analiza programu

Opis teoretyczny

Szukamy najmniejszej liczby osób takiej, że prawdopodobieństwo istnienia w tej grupie dwóch osób świętujących urodziny w ten sam dzień wynosi co najmniej 50%. Generujemy losowo grupę n osób, a raczej daty ich urodzin – liczbę w przedziale $[1; 365]$. Daty urodzin wybierane są z rozkładem jednostajnym, nie uwzględnia się również lat przestępnych – urodzin 29 lutego.

Dla każdej osoby sprawdzamy, czy istnieje inna osoba z tą samą datą urodzin. Jeśli istnieje ktoś taki, przerywamy test, bo nie potrzebujemy informacji o kolejnej dopasowanej parze. Zapisujemy wynik (para znaleziona lub nie), generujemy na nowo pulę n osób i przeprowadzamy kolejny test.

Po ustalonej w pliku `input\in.txt` liczbie prób zliczamy, ile z nich było pomyślnych (para została znaleziona), a ile nie. Na tej podstawie obliczamy szanse:

$$p_n = \frac{ilość\ prób\ pomyślnych}{ilość\ testów}$$

Jeśli $p_n \geq \frac{1}{2}$, to warunek problemu został spełniony, a n jest naszym ostatecznym wynikiem – szukaną najmniejszą liczbą osób. Jeśli $p_n < \frac{1}{2}$, to szukamy dalej – zwiększamy grupę o jedną osobę (zwiększamy n o 1) i powtarzamy operacje.

Program nie wpadnie nigdy w nieskończoną pętlę, ponieważ dla 366 osób zasada szufladkowa Dirichleta gwarantuje, że dwie z nich będą dzielić urodziny.

Omówienie funkcji

→ `generate_random_birthdays`

Funkcja `generate_random_birthdays` przyjmuje jako argument liczbę osób n i generuje próbkę danych – n dat urodzin w postaci numeru dnia w roku. Wykorzystywana jest do tego funkcja `random.randint` z biblioteki `random`. Jak wspomniane wyżej, daty są wybierane z rozkładem jednostajnym, co nie odbiega mocno od rzeczywistości. Pomijany jest także dzień 29 lutego.

Daty (numery dni w roku) są zapisywane w tablicy `birthdays`. Po wygenerowaniu n dat, funkcja zwraca tablicę `birthdays`.

```
def generate_random_birthdays(n):
    birthdays = []
    for i in range(n):
        date = random.randint( a: 1, b: 365)
        birthdays.append(date)
    return birthdays
```

→ `same_birthday_chances`

Funkcja `same_birthday_chances` przyjmuje jako argument liczbę osób n i zwraca obliczone szanse spełnienia warunków problemu urodzin dla tej liczby osób. Przyjmujemy liczbę testów 20000, dla większej liczby uzyskane prawdopodobieństwa będą oczywiście bardziej dokładne i zbliżone do rozwiązania dokładnego.

Na początku ustawiamy licznik `are_same` na 0, będzie on przechowywał informację o liczbie testów, które spełniają warunki.

Każdy test polega na wygenerowaniu próbki n dat urodzin, za pomocą funkcji `generate_random_birthdays`. Następnie dla każdej wygenerowanej osoby sprawdzamy funkcją `count`, czy ktoś współdzieli z nią datę urodzin. Jeśli istnieje taka osoba, to zwiększamy licznik `are_same` i przechodzimy do kolejnego testu.

Po przeprowadzeniu wszystkich testów liczymy prawdopodobieństwo spełnienia warunków problemu urodzin dla n osób.

```
def same_birthday_chances(n):
    are_same = 0
    tests = 20000
    for i in range(tests):
        birthdays = generate_random_birthdays(n)
        for birthday in birthdays:
            if birthdays.count(birthday) > 1:
                are_same += 1
                break
    return are_same/tests
```

→ `validate_tests_number`

Ta funkcja przyjmuje jako argument nazwę pliku, w którym powinny się zawierać dane wejściowe – liczba testów, które ma przeprowadzić program dla danej liczby osób n.

Funkcja czyta zadany plik – jeśli plik z danymi nie istnieje, lub dane zapisane w nim są niepoprawne, to funkcja zwraca domyślną wartość 20000. W przeciwnym wypadku zwraca wartość zapisaną w pliku (jego pierwszej linijce).

```
def validate_tests_number(input_file):
    try:
        with open(input_file, 'r') as f_in:
            tests = f_in.readline()
            tests = int(tests)
            if tests < 1:
                raise ValueError
    except (IOError, ValueError):
        print("Niepoprawne dane wejściowe. Przyjmuję domyślną liczbę testów 20000.")
        tests = 20000
    return tests
```

→ `solve_birthday_problem`

Funkcja `solve_birthday_problem` przyjmuje jako argumenty ścieżkę względną pliku, w którym powinny znajdować się dane wejściowe (`input_file`) oraz pliku, do którego ma zostać zapisany wynik obliczeń (`output_file`). Zwraca `True` – jeśli wszystko się powiodło, `False` – jeśli program napotkał problem i nie dokończył obliczeń.

Na początku program zapisuje w zmiennej `tests` wynik funkcji `validate_tests_number(input_file)`. Następnie funkcja próbuje otworzyć `output_file` – jeśli się to nie uda, zwraca `False`, kończąc działanie.

Jeśli udało się otworzyć plik, tworzona jest zmienna `start_time` z aktualnym czasem. Następnie program zadaje najmniejszą liczbę osób n, dla której warunek problemu mógłby być spełniony, czyli 2. Zapisuje prawdopodobieństwo rozwiązania problemu urodzin dla tej liczby osób n i zadanej liczby testów, obliczone przez funkcję `same_birthday_chances(n, tests)` w zmiennej `chances`. Jeśli szanse są większe, niż 50%, to problem urodzin został rozwiązany, w przeciwnym razie liczba osób n jest inkrementowana.

Kiedy funkcja znajdzie szukaną liczbę osób n, tworzona jest zmienna `end_time` z aktualnym czasem. Obliczona liczba osób, prawdopodobieństwo dla tej liczby, liczba przeprowadzonych testów, a także łączny czas obliczeń (różnica między `end_time` i `start_time`) są zapisywane do pliku `output_file`. Funkcja zwraca `True` i kończy swoje działanie.

```
def solve_birthday_problem(input_file, output_file):
    tests = validate_tests_number(input_file)

    try:
        with open(output_file, 'w') as f_out:
            start_time = time.time()
            n = 2
            chances = same_birthday_chances(n, tests)

            while chances < 0.5:
                n += 1
                chances = same_birthday_chances(n, tests)

            end_time = time.time()
            runtime = end_time - start_time

            f_out.write(f"Najmniejsza liczba osób, które spełniają warunek: {n}\n")
            f_out.write(f"z przybliżonymi szansami: {(chances*100):.2f}%\n")
            f_out.write(f"Liczba wykonanych testów dla danej liczby osób: {tests}\n")
            f_out.write(f"Czas trwania obliczeń: {runtime:.4f} s")
            return True
    except FileNotFoundError:
        print(f"Nastąpił błąd przy otwarciu pliku {output_file}")
        return False
```

→ `main`

Główna funkcja wywoływana przez program tworzy ścieżki plików wejściowych i wyjściowych. Następnie wywołuje funkcję `solve_birthday_problem` z utworzonymi ścieżkami jako argumenty. W zależności od wyniku wywołanej funkcji informuje użytkownika o przebiegu operacji.

```
if __name__ == '__main__':
    input_folder = "input"
    input_file = os.path.join(input_folder, "in.txt")
    output_folder = "output"
    os.makedirs(output_folder, exist_ok=True)
    output_file = os.path.join(output_folder, "out.txt")

    print(f"Przetwarzanie pliku: {input_file}")
    if solve_birthday_problem(input_file, output_file):
        print(f"Wynik zapisany w pliku: {output_file}")
    else:
        print(f"Błąd przetwarzania pliku: {output_file}")
```

Działanie programu

Program uruchamiamy, otwierając skrypt `menu.bat`. Ten plik wyświetla menu i informuje użytkownika o możliwych opcjach do wyboru, ponumerowanych od 1 do 4. Jeśli użytkownik wybierze nieistniejącą opcję, zostanie wyświetlony komunikat.

Po wybraniu opcji `1. Uruchom program`, skrypt kasuje poprzedni `raport.html` (o ile istnieje), tworzy folder `output` oraz wywołuje program `paradoks-urodzin.py`.

Plik `paradoks-urodzin.py` ustala pliki wejścia i wyjścia. Następnie ustala liczbę testów do wykonania – jeśli w pliku wejściowym znajduje się odpowiednia wartość, to ją przyjmuje, w przeciwnym wypadku przyjmuje wartość domyślną 20000. Uruchamia mierzenie czasu wykonywania obliczeń. Następnie, aż do znalezienia najmniejszej liczby spełniającej warunek, wykonuje poniższe operacje.

Dla danej liczby osób n, generuje losowo daty ich urodzin. Następnie dla każdej osoby sprawdza, czy spełnia warunek. Powtarza generowanie i sprawdzanie tyle razy, ile zostało ustalone jako liczba testów. Na tej podstawie oblicza szanse spełnienia warunku dla n osób i zadanej liczby testów. Jeśli te szanse są większe niż 50%, przechodzi dalej, w przeciwnym wypadku zwiększa liczbę osób o jedną i wykonuje daną liczbę testów.

Po znalezieniu odpowiedniej liczby, program oblicza jak długo zajęły mu operacje. Następnie do pliku wyjściowego zapisuje informacje o obliczonej liczbie osób, szansach, liczbie wykonanych testów oraz czasie obliczeń. Jeśli coś się nie powiodło, zwraca informację o błędzie.

Program `paradoks-urodzin.py` kończy się, więc skrypt `menu.bat` wywołuje program `raport.py`. Ma on wygenerować raport działania programu do pliku `raport.html`. Najpierw wpisuje do niego szkielet pliku `.html`, określający style i generujący nagłówki

z tytułem projektu i datą wygenerowania. Następnie otwiera plik wynikowy, zwrócony przez `paradoks-urodzin.py`. Przepisuje kolejno linijki z tego pliku, do pliku `raport.html`. Następnie dopisuje końcówkę pliku, załączając obrazek węża świętującego swoje urodziny i kończy działanie, wracając do skryptu `menu.bat`.

Skrypt otwiera nowoutworzony `raport.html` i informuje o możliwości powrotu do menu, po naciśnięciu dowolnego klawisza.

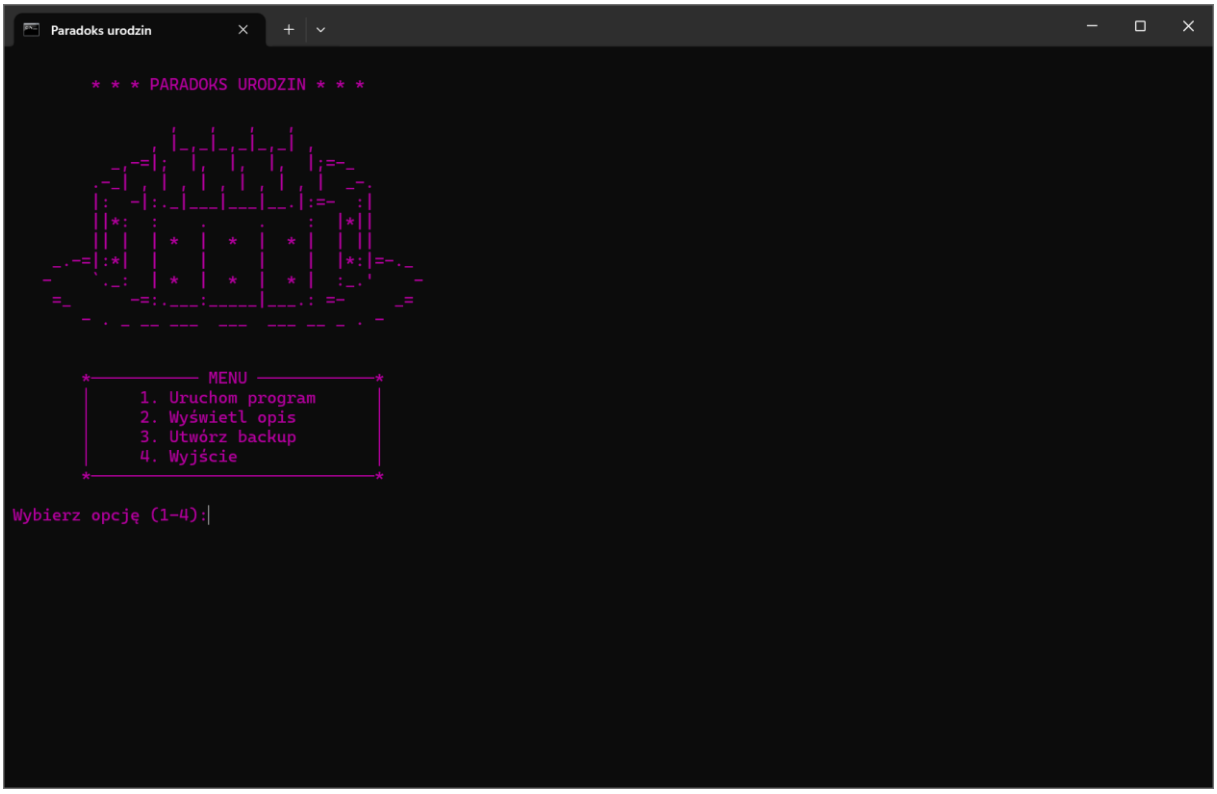
Wybranie opcji `2. Wyświetl opis`, wyświetla na ekran konsoli krótki opis działania programu.

Wybranie opcji `3. Utwórz backup`, powoduje utworzenie folderu `backups` (jeśli jeszcze nie istnieje), a w nim folderu nazwanego datą jego wygenerowania. Do środka folderu program kopiuje ostatnie pliki wynikowe, a także ostatni wygenerowany raport.

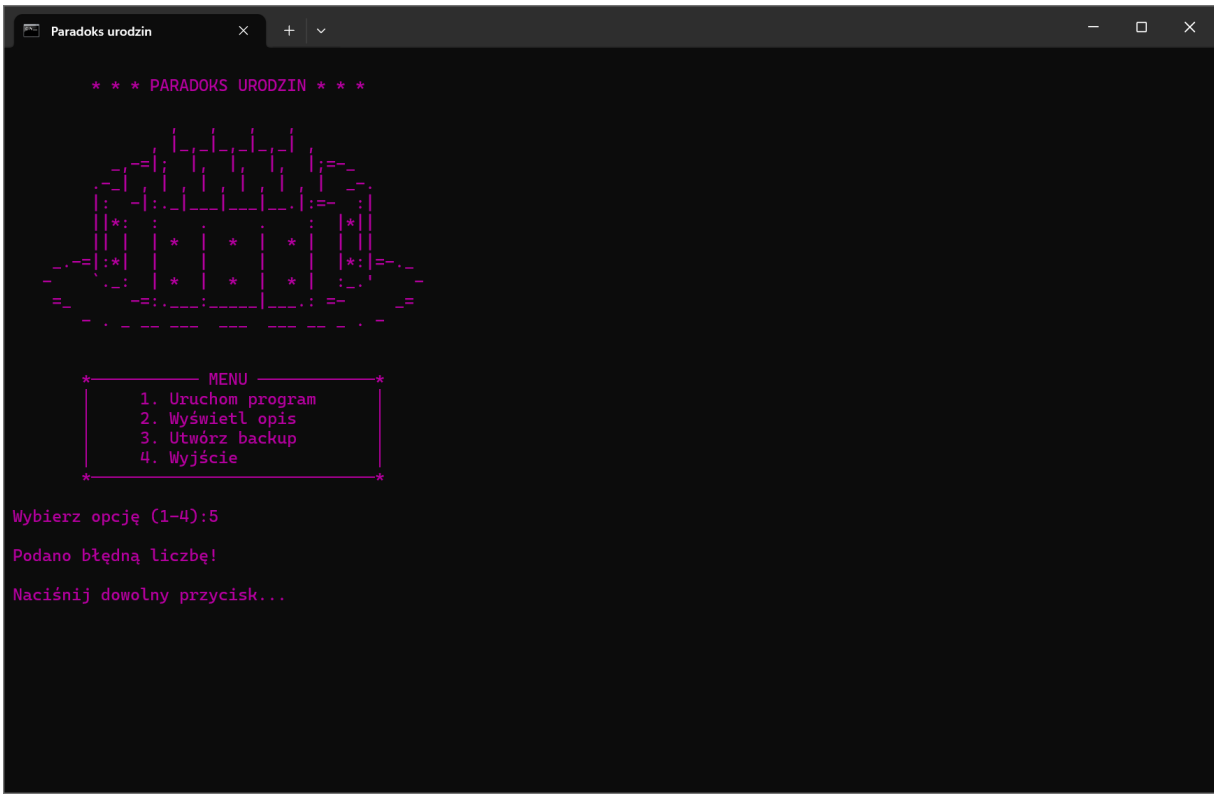
Wybranie opcji `4. Wyjście`, powoduje zakończenie działania programu `menu.bat`.

Zrzuty ekranu

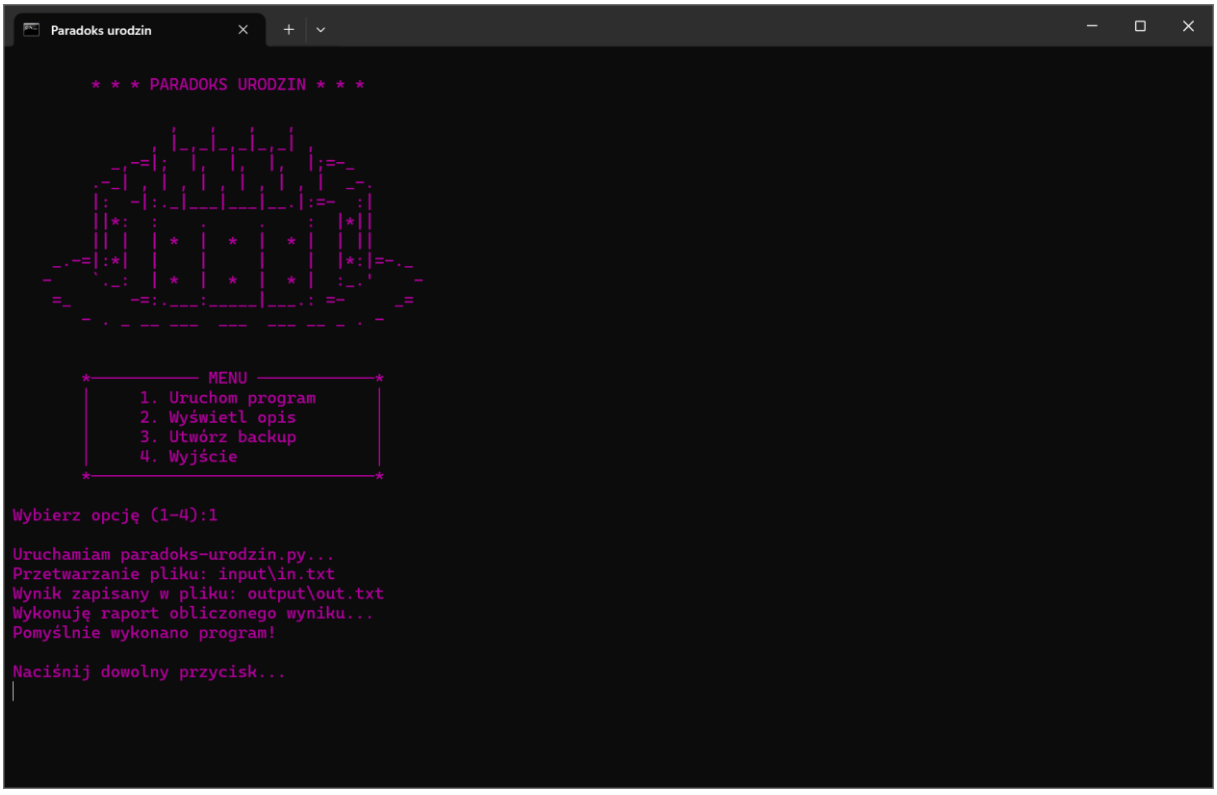
Działanie programu



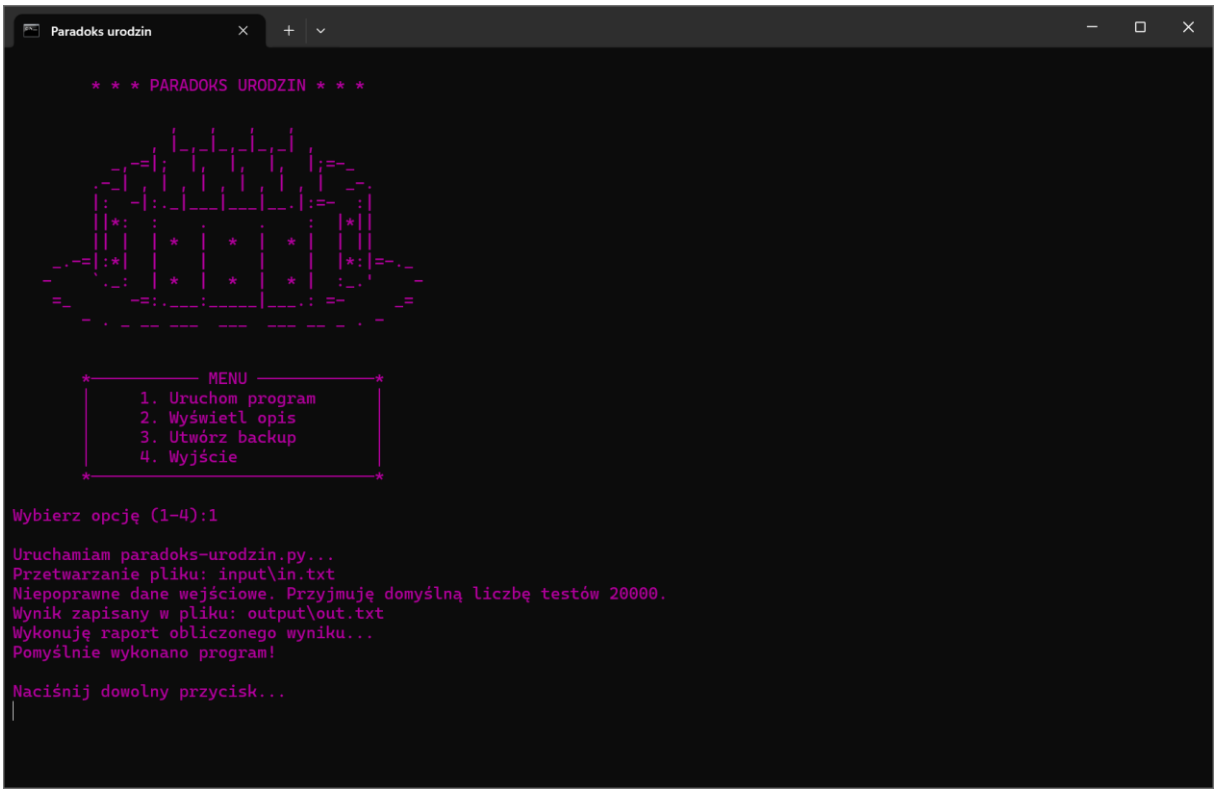
Zrzut ekranu 1



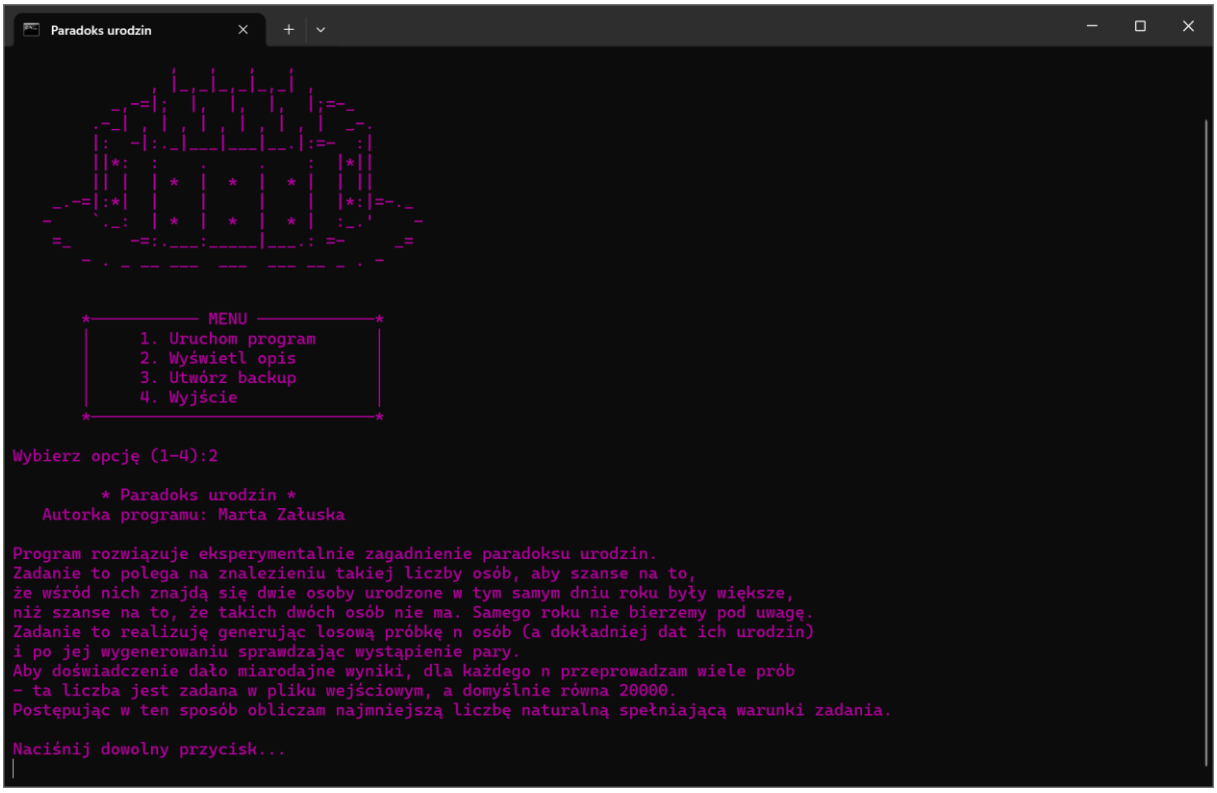
Zrzut ekranu 2



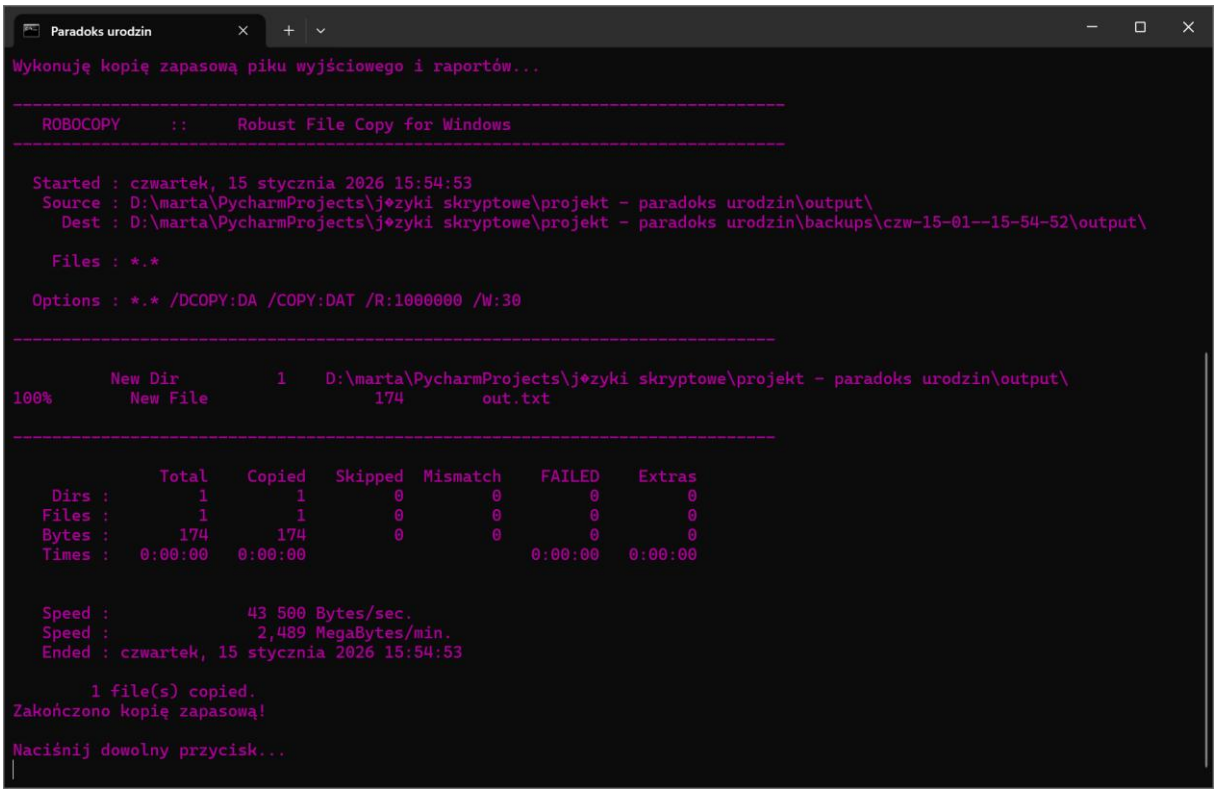
Zrzut ekranu 3



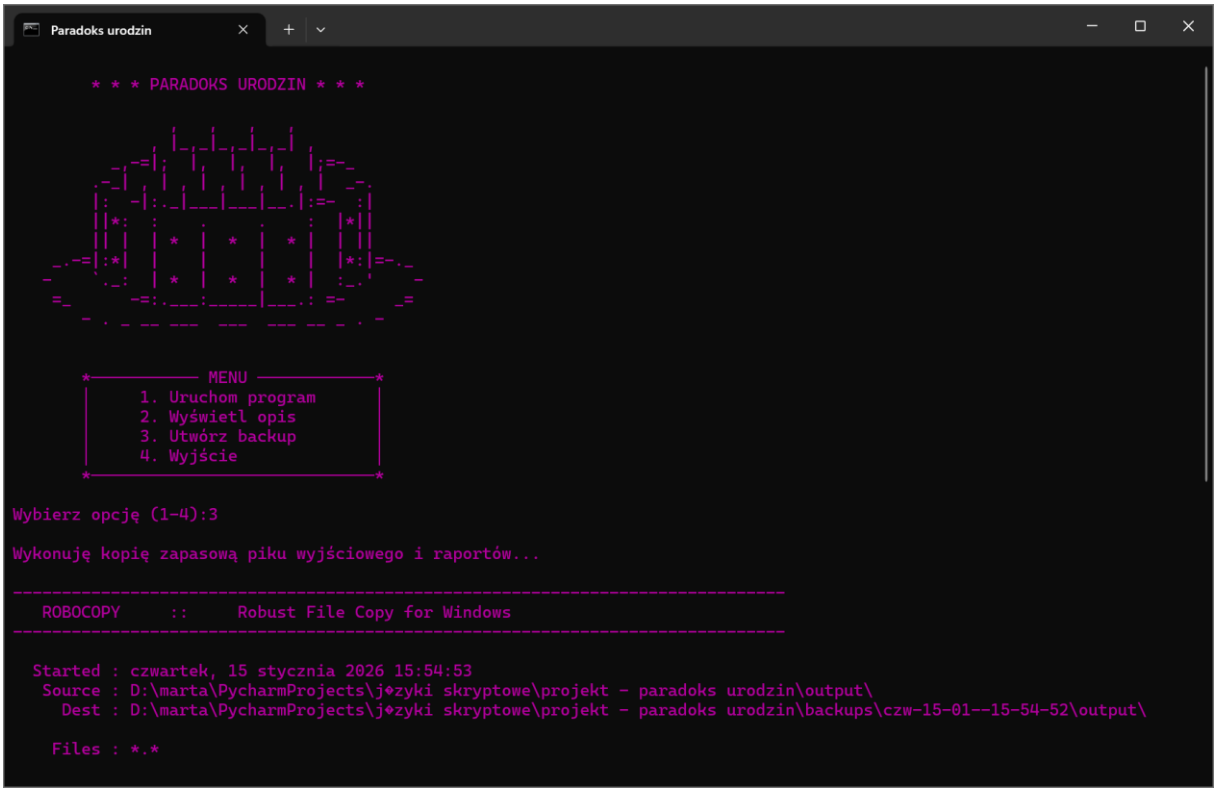
Zrzut ekranu 4



Zrzut ekranu 5



Zrzut ekranu 6



Zrzut ekranu 7

Raport

Zrzut ekranu 8

Kod źródłowy

paradoks_urodzin.py

```python
import random
import os
import time

def generate_random_birthdays(n):
    birthdays = []
    for i in range(n):
        date = random.randint( a: 1, b: 365)
        birthdays.append(date)
    return birthdays

def same_birthday_chances(n, t = 20000):
    are_same = 0
    tests = t
    for i in range(tests):
        birthdays = generate_random_birthdays(n)
        for birthday in birthdays:
            if birthdays.count(birthday) > 1:
                are_same += 1
                break
    return are_same/tests

def validate_tests_number(input_file):
    try:
        with open(input_file, 'r') as f_in:
```

```
            tests = f_in.readline()
            tests = int(tests)
            if tests < 1:
                raise ValueError
    except (IOError, ValueError):
        print("Niepoprawne dane wejściowe. Przyjmuję domyślną liczbę testów 20000.")
        tests = 20000
    return tests

def solve_birthday_problem(input_file, output_file):
    tests = validate_tests_number(input_file)

    try:
        with open(output_file, 'w') as f_out:
            start_time = time.time()
            n = 2
            chances = same_birthday_chances(n, tests)

            while chances < 0.5:
                n += 1
                chances = same_birthday_chances(n, tests)

            end_time = time.time()
            runtime = end_time - start_time

            f_out.write(f"Najmniejsza liczba osób, które spełniają warunek: {n}\n")
            f_out.write(f"z przybliżonymi szansami: {(chances*100):.2f}%\n")
            f_out.write(f"Liczba wykonanych testów dla danej liczby osób: {tests}\n")
            f_out.write(f"Czas trwania obliczeń: {runtime:.4f} s")
            return True
    except FileNotFoundError:
        print(f"Nastąpił błąd przy otwarciu pliku {output_file}")
        return False


if __name__ == '__main__':
    input_folder = "input"
    input_file = os.path.join(input_folder, "in.txt")
    output_folder = "output"
    os.makedirs(output_folder, exist_ok=True)
    output_file = os.path.join(output_folder, "out.txt")

    print(f"Przetwarzanie pliku: {input_file}")
    if solve_birthday_problem(input_file, output_file):
        print(f"Wynik zapisany w pliku: {output_file}")
    else:
        print(f"Błąd przetwarzania pliku: {output_file}")
```

raport.py

```python
from datetime import datetime

date_display = datetime.now().strftime("%d.%m.%Y %H:%M:%S")
output_file = open(f"raport.html", "w+", encoding="utf-8")

output_file.write(f"""
<html>
    <head>
        <meta charset="utf-8">
        <style>
            body {{
                text-align: center;
                background-color: #FBF2FF;
                color: #620E5E;
                font-family: Trebuchet MS, sans-serif;
            }}
            h1 {{
                padding-top: 50px
            }}
            h3 {{
                padding-top: 25px
            }}
        </style>
        <title>Paradoks urodzin</title>
    </head>
    <body>
        <h1>Rozwiązanie problemu urodzin</h1>
        <h3>Raport {date_display}</h3>
        <p>""")

with open(f"output/out.txt", "r") as file:
    for line in file:
        output_file.write(line)
        output_file.write("<br>")
output_file.write("""
        </p>
        <img src="birthday-snake.png", height=100px, style="padding: 25px">
    </body>
</html>""")

output_file.close()
```

menu.bat

```
@echo off
chcp 65001 >nul

Title Paradoks urodzin
color 0D

:menu
cls

echo.
echo         * * * PARADOKS URODZIN * * *
echo.
echo                  ,   ,   ,   ,
echo               , ^|_,_^|_,_^|_,_^| ,
echo           _,-=^|;  ^|,  ^|,  ^|,  ^|;=-_
echo         .-_^| , ^| , ^| , ^| , ^| , ^|  _-.
echo         ^|:  -^|:._^|___^|___^|__.^|:=-  :^|
echo         ^|^|*:  :    .     .    :  ^|*^|^|
echo         ^|^| ^|  ^| *  ^|  *  ^|  * ^|  ^| ^|^|
echo     _.-=^|:*^|  ^|    ^|     ^|    ^|  ^|*:^|=-._
echo    -    `._:  ^| *  ^|  *  ^|  * ^|  :_.'    -
echo     =_      -=:._____:_____^|___.: =-     _=
echo        - . _ __ ___  ___  ___ __ _ . -
echo.
echo.
echo        *———————— MENU ————————*
echo        |     1. Uruchom program      |
echo        |     2. Wyświetl opis        |
echo        |     3. Utwórz backup        |
echo        |     4. Wyjście              |
echo        *—————————————————————————*
echo.

set choice=0
set /p choice=Wybierz opcję (1-4):

if %choice%==1 goto opcja1
if %choice%==2 goto opcja2
if %choice%==3 goto opcja3
if %choice%==4 goto exit

echo.
echo Podano błędną liczbę!
echo.
echo Naciśnij dowolny przycisk...
pause >nul
goto menu

:opcja1

```

```
IF EXIST raport.html DEL raport.html
IF NOT EXIST output mkdir output
DEL /Q output
echo.
echo Uruchamiam paradoks-urodzin.py...
python paradoks-urodzin.py
echo Wykonuję raport obliczonego wyniku...
python raport.py
echo Pomyślnie wykonano program!
echo.
call raport.html
echo Naciśnij dowolny przycisk...
pause >nul
goto menu

:opcja2
echo.
echo          * Paradoks urodzin *
echo    Autorka programu: Marta Załuska
echo.
echo Program rozwiązuje eksperymentalnie zagadnienie paradoksu urodzin.
echo Zadanie to polega na znalezieniu takiej liczby osób, aby szanse na to,
echo że wśród nich znajdą się dwie osoby urodzone w tym samym dniu roku były większe,
echo niż szanse na to, że takich dwóch osób nie ma. Samego roku nie bierzemy pod uwagę.
echo Zadanie to realizuję generując losową próbkę n osób (a dokładniej dat ich urodzin)
echo i po jej wygenerowaniu sprawdzając wystąpienie pary.
echo Aby doświadczenie dało miarodajne wyniki, dla każdego n przeprowadzam wiele prób
echo - ta liczba jest zadana w pliku wejściowym, a domyślnie równa 20000.
echo Postępując w ten sposób obliczam najmniejszą liczbę naturalną spełniającą warunki zadania.
echo.
echo Naciśnij dowolny przycisk...
pause >nul
goto menu

:opcja3
echo.
echo Wykonuję kopię zapasową piku wyjściowego i raportów...
IF NOT EXIST backups mkdir backups

for /f "tokens=1-3 delims=-/. " %%a in ("%date%") do (
    set day=%%a
    set month=%%b
    set year=%%c
)

set hour=%time:~0,2%
set minute=%time:~3,2%
set second=%time:~6,2%

if "%hour:~0,1%"==" " set hour=0%hour:~1,1%
```

```
101
102 set date=%day%-%month%-%year%--%hour%-%minute%-%second%
103
104 IF EXIST raport.html mkdir backups\%date%
105 robocopy output backups\%date%\output
106 copy raport.html backups\%date%\raport.html
107
108 echo Zakończono kopię zapasową!
109 echo.
110 echo Naciśnij dowolny przycisk...
111 pause >nul
112 goto menu
113
114 :exit
```