

Dokumentacja projektu
Parser PESEL
Języki skryptowe

Dawid Bitner, IA

25 stycznia 2019

Politechnika Śląska
Wydział Matematyki Stosowanej
Rok akademicki 2018/2019











Część I

Opis programu

Program ma za zadanie sprawdzanie poprawności podanego numeru **Powszechnego Elektronicznego System Ewidencji Ludności (PESEL)**. Program działa na zasadzie niewielkiej bazy danych przechowującej rekordy i potrafiącej tworzyć kopię zapasową wyników.

Instrukcja obsługi

Aby dodać kolejne numery PESEL do bazy, w celu sprawdzenia ich poprawności należy stworzyć kolejne pliki *inX.txt*, w folderze *in* gdzie *X* to kolejna liczba określająca numer pliku wejściowego. W celu uruchomienia programu należy uruchomić plik *skrypt.bat*, wyniki działania programu zostaną zapisane w folderze *out*, a raport wygenerowany do folderu *backup*, oraz jeżeli użytkownik ma poprawnie skojarzone otwieranie plików *HTML* z przeglądarką - otwarty w domyślnej przeglądarce.

 backup	24.12.2018 18:45	Folder plików	
 cpp	24.12.2018 18:42	Folder plików	
 css	24.12.2018 16:28	Folder plików	
 in	24.12.2018 17:00	Folder plików	
 out	24.12.2018 18:45	Folder plików	
 python	24.12.2018 16:50	Folder plików	
 info.txt	24.12.2018 18:54	Dokument tekstowy	1 KB
 parserPESEL.exe	24.12.2018 18:43	Aplikacja	1 686 KB
 skrypt.bat	24.12.2018 18:45	Plik wsadowy Win...	1 KB
 skrypt_parser.py	24.12.2018 17:09	JetBrains PyCharm ...	5 KB

Część II

Część techniczna

Program uruchamiamy poprzez normalne uruchomienie pliku *skrypt.bat*, lub wywołania go za pomocą linii komend. Skrypt konsolowy ma za zadanie wywoływanie kolejnych elementów programów tj.:

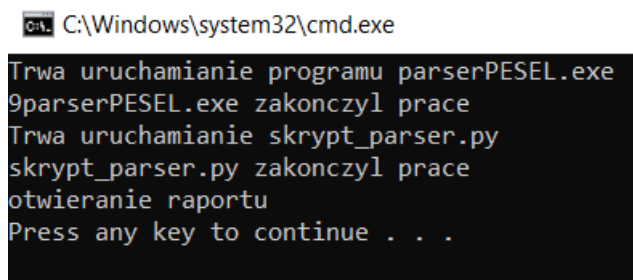
1. Stworzenie folderów: *in*, *out* i *backup* jeżeli nie istnieją.
2. Wywołanie *parserPESEL.exe* - algorytmu napisanego w języku *C++* sprawdzającego poprawność numerów PESEL.
3. Wywołanie skryptu *skrypt_parser.py* napisanego w języku *Python* generującego raport/kopię zapasową.

W celu uproszczenia działania programu powyższe pliki zostały umieszczone w folderze nadrzędnym do folderów *in*, *out* i *backup*.

Opis działania

skrypt.bat

Skrypt w pierwszej kolejności tworzy foldery: *in*, *out* i *backup* jeżeli nie istnieją. Następnie wywołuje kolejno *parserPESEL.exe* i *skrypt_python.py* informując o swoich działaniach użytkownika.



```
C:\Windows\system32\cmd.exe
Trwa uruchamianie programu parserPESEL.exe
9parserPESEL.exe zakonczyl prace
Trwa uruchamianie skrypt_parser.py
skrypt_parser.py zakonczyl prace
otwieranie raportu
Press any key to continue . . .
```

parserPESEL.exe

Odpowiada za część algorytmiczną programu. Każdy wpis w rejestrze PESEL jest określany unikatowym symbolem jednoznacznie identyfikującym osobę fizyczną. Numer PESEL jest to 11-cyfrowy stały symbol numeryczny jednoznacznie identyfikujący określoną osobę fizyczną. Zbudowany jest z następujących elementów: zakodowanej daty urodzenia, liczby porządkowej, zakodowanej płci, cyfry kontrolnej.

Przykładowa postać:

440514 0145 8

cyfry [1-6] – data urodzenia z określeniem stulecia urodzenia

cyfry [7-10] – numer serii z oznaczeniem płci

cyfra [10] – płeć

cyfra [11] – cyfra kontrolna

skrypt_python.py

Odpowiada za stworzenie kopii zapasowej/raportu w postaci tabeli w pliku *html*. Raport jest tworzony na zasadzie kopii przyrostowej tj. jeżeli istnieje kopia stworzona wcześniej, zostaje ona wczytana, zmienna sprawdzająca ilość plików wyjściowych zostaje ustawiona na liczbę plików + 1 i dopisywanie wyników rozpoczyna się od pierwszego nowego pliku który nie został zawarty w poprzednim raporcie. Jeżeli żadna kopia nie została wcześniej utworzona, skrypt wypisze do pliku *html* wszystkie dane w plikach wejściowych i wyjściowych. Nazwa pliku raportu jest w formacie *timestamp.html*, gdzie *timestamp* to dokładny czas stworzenia raportu w postaci: *ROK-MIESIAC-DZIEN_GODZINA-MINUTA-SEKUNDA*.

Implementacja

parserPESEL.exe

Schemat działania:

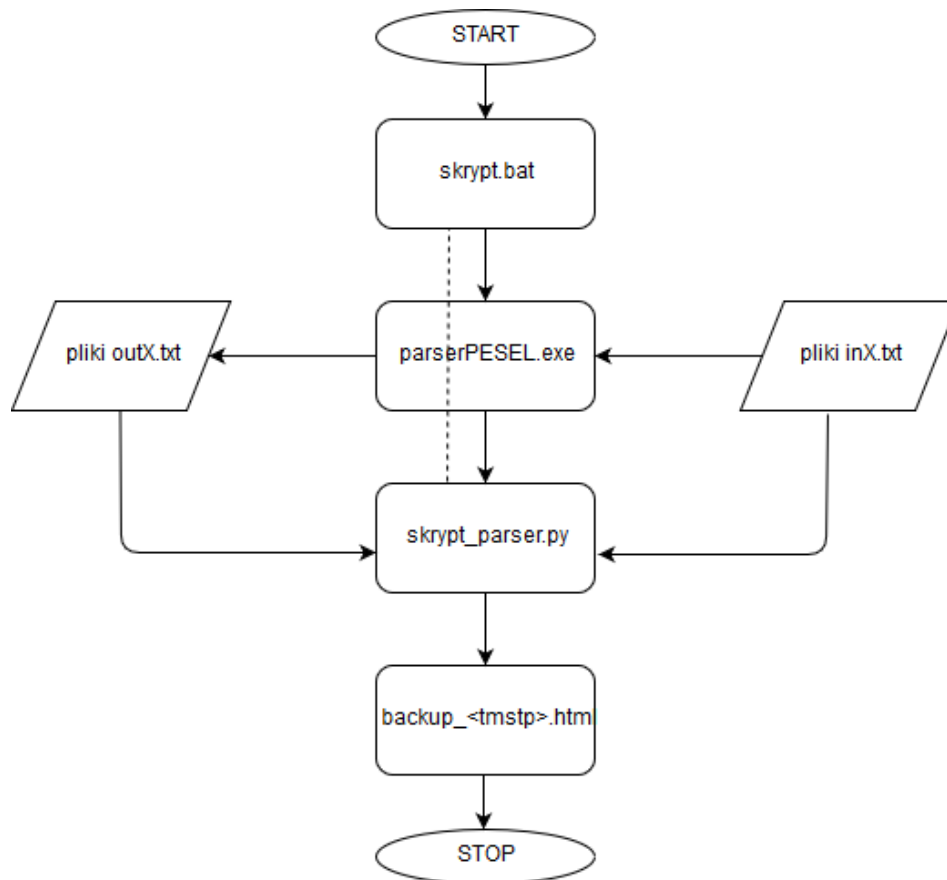
- wyszukaj pliki *out* z rozszerzeniem *.txt* w folderze *out*
- ostatni plik *outX.txt* zostanie zapisany do zmiennej, jeżeli nie istnieje to zmienna wynosi 0
- wyciągamy *X* z nazwy pliku, zapisujemy do zmiennej typu *int*
- wczytujemy odpowiednie pliki *inX.txt* z folderu *in*, zaczynamy od *X*, gdzie *X* to *X* z ostatniego pliku *out* + 1
- dla każdego pliku wejściowego wykonujemy poniższy algorytm, a jego wynik zapisujemy do odpowiedniego pliku wyjściowego:
- algorytm wywołuje kolejne metody sprawdzające poprawność numeru PESEL, jeżeli w którymś miejscu walidator znajdzie nieprawidłowość zwróci wartość zmiennej typu *bool* jako *false*, w przeciwnym wypadku jako *true*, operacja ta wykonywana jest dla każdego kolejnego pliku wejściowego
- przy wartości *true* do pliku wyjściowego zostaje wypisana informacja o poprawności numeru, dacie urodzenia tej osoby i jej płci, w przeciwnym wypadku w pliku wyjściowym zostaje wypisana informacja o nieprawidłowości walidacji
- jeżeli nie istnieje już żaden plik wejściowy, kończymy działanie programu

skrypt_python.py

Schemat działania:

- skrypt sprawdza czy w folderze *backup* istnieje jakaś kopia zapasowa, jeżeli tak zostanie skopiowana, a nowe dane będą dopisywane poniżej, jeżeli nie zostanie stworzony plik *html* od podstaw, zawierający nagłówki i dołączenie arkusza stylów
- z ostatniego pliku wyjściowego zostaje wyciągnięta liczba, mówiąca o ilości plików, zostaje zwiększona o 1
- w postaci kopii przyrostowej zostają dopisywane kolejne komórki tabeli zawierające treść plików wejściowych i odpowiadających im plików wyjściowych

Ogólny schemat blokowy



Pełen kod programu

Kod zawiera komentarze dokładnie tłumaczące działanie wybranych fragmentów programu.

skrypt.bat

```
@echo off
IF NOT EXIST in mkdir in
IF NOT EXIST out mkdir out
IF NOT EXIST backup mkdir backup
echo Trwa uruchamianie programu parserPESEL.exe
parserPESEL.exe
echo parserPESEL.exe zakonczyl prace
echo Trwa uruchamianie skrypt_parser.py
py -u skrypt_parser.py
echo skrypt_parser.py zakonczyl prace
echo otwieranie raportu
pause
```

parserPESEL.exe

main.cpp

```
#include "pesel.h"

int main(){
    pesel Pesel;

    char PESEL[12] = ""; // prawidlowy numer PESEL ma dokladnie 11 cyfr,
    wystarczy ze jest dluzszy o 1 cyfra zeby byl nieprawidlowy
    int i;

    /*rozwiazanie problemu za:
    https://stackoverflow.com/questions/26475540/c-having-problems-with-a-simple-example-of-findfirstfile */

    string DATA_DIR = "out/out*.*"; // wyszukaj tylko pliki out*.* w folderze out;
    string lastFile = ""; // tworzenie stringu, rownie dobrze moze byc samo
    string lastFile;

    HANDLE hFind; // uchwyt
    WIN32_FIND_DATA data; // https://docs.microsoft.com/en-us/windows/desktop/api/minwinbase/ns-minwinbase-\_win32\_find\_dataa

    hFind = FindFirstFileA(DATA_DIR.c_str(), &data); // https://docs.microsoft.com/en-us/windows/desktop/api/fileapi/nf-fileapi-findfirstfilea

    if(hFind != INVALID_HANDLE_VALUE){
        while(FindNextFile(hFind, &data)); // czytanie kazdego pliku, ostatni zostanie
        lastFile = data.cFileName; // lastFile to nazwa pliku
        FindClose(hFind); // zamknij handler
    }

    if(lastFile.length()){ // jesli to nie jest pusty string, czyli jesli istnieje juz
    jakis out
    lastFile = lastFile.substr(3, lastFile.length() - 4 - 3); // 4 = ".txt"; 3 = "out"
    i = atoi(lastFile.c_str()) + 1; // string -> char[] -> int
    }
    else // jesli jednak zaden out nie istnieje
    i = 1; // to zacznij od pierwszego

    cout << lastFile;
```

```

do{
stringstream ss; // z <sstream>, int to string
ss << i; // z <sstream>, int to string

//---OTWIERANIE PLIKU WEJSCIOWEGO---//
    string filenameInput = "in/in" + ss.str(); // inX.txt, gdzie X
    to kolejny iterator
filenameInput += ".txt"; // inX.txt, gdzie X to kolejny iterator

ifstream fileInput;
fileInput.open(filenameInput.c_str(), ios::in);

//---WARUNEK STOPU---//
if(!fileInput.is_open()){ // jeżeli plik nie istnieje to zakończ
    return 0;
}

//---CZYTANIE Z PLIKU---//
    fileInput>>PESEL;
    fileInput.close();

    Pesel.PeselValidator(PESEL); // funkcja uruchamiająca walidator -> pesel.cpp

//---TWORZENIE DANYCH WYJŚCIOWYCH---//
    string filenameOutput = "out/out" + ss.str();
filenameOutput += ".txt";
ofstream fileOutput;
fileOutput.open(filenameOutput.c_str(), ios::out);

    if (Pesel.valid == 1) {
        fileOutput<<"Numer PESEL jest prawidłowy\n";
        fileOutput<<"Rok urodzenia: "<<Pesel.getBirthYear()<<"\n";
        fileOutput<<"Miesiąc urodzenia: "<<Pesel.getBirthMonth()<<"\n";
        fileOutput<<"Dzień urodzenia: "<<Pesel.getBirthDay()<<"\n";
        fileOutput<<"Plec: "<<Pesel.getSex()<<"\n";
    }
    else {
        fileOutput<<"Numer PESEL jest nieprawidłowy\n";
    }
    i++; // zwiększa iterator, outX.txt
}while(true);

return 0;
}

```

pesel.h

```
#ifndef PESEL_H
#define PESEL_H

#include <iostream>
#include <cstdlib>
#include <string>
#include <fstream>
#include <sstream>
#include <windows.h>

using namespace std;

class pesel{
public:

    pesel();

    short PESEL[11];
    bool valid = 0;

    int getBirthYear();
    int getBirthMonth();
    int getBirthDay();
    const char* getSex();
    int checkSum();
    int checkMonth();
    int checkDay();
    int leapYear(int);
    void PeselValidator(char*);

    virtual ~pesel();
};

#endif // PESEL_H
```

pesel.cpp

```
#include "pesel.h"

pesel::pesel()
```



```

{
    //ctor
}

pesel::~pesel()
{
    //dtor
}

//https://pl.wikipedia.org/wiki/PESEL

int pesel::getBirthYear(){
/*Numeryczny zapis daty urodzenia przedstawiony jest w następującym porządku: dwie
ostatnie cyfry roku, miesiąc i dzień.
Dla odróżnienia poszczególnych stuleci przyjęto następującą metodę kodowania:

    dla osób urodzonych w latach 1900 do 1999 - miesiąc zapisywany jest w sposób
    naturalny, tzn. dwucyfrowo od 01 do 12
    dla osób urodzonych w innych latach niż 1900-1999 dodawane są do numeru miesiąca
    następujące wielkości:
        dla lat 1800-1899 - 80
        dla lat 2000-2099 - 20
        dla lat 2100-2199 - 40
        dla lat 2200-2299 - 60.*/

    int year;
    int month;

    year = 10 * PESEL[0];
    year += PESEL[1];
    month = 10 * PESEL[2];
    month += PESEL[3];

    if (month > 80 && month < 93){
        year += 1800;
    }
    else
        if (month > 0 && month < 13){
            year += 1900;
        }
    else
        if (month > 20 && month < 33){
            year += 2000;
        }
    else

```

```

        if (month > 40 && month < 53){
            year += 2100;
        }
    else
        if (month > 60 && month < 73){
            year += 2200;
        }
    return year;
}

int pesel::getBirthMonth(){
    int month;
    month = 10 * PESEL[2];
    month += PESEL[3];

    if (month > 80 && month < 93){
        month -= 80;
    }
    else
        if (month > 20 && month < 33){
            month -= 20;
        }
    else
        if (month > 40 && month < 53){
            month -= 40;
        }
    else
        if (month > 60 && month < 73){
            month -= 60;
        }
    return month;
}

int pesel::getBirthDay(){
    int day;

    day = 10 * PESEL[4];
    day += PESEL[5];

    return day;
}

const char* pesel::getSex(){
/*Informacja o płci osoby, której zestaw informacji jest identyfikowany, zawarta
jest na 10. (przedostatniej) pozycji numeru PESEL.

```

cyfry 0, 2, 4, 6, 8 - oznaczają płeć żeńską
cyfry 1, 3, 5, 7, 9 - oznaczają płeć męską

Po zmianie płci przydzielany jest nowy numer PESEL*/

```
    if (valid){
        if (PESEL[9] % 2 == 1) {
            return "Mezczynna";
        }
        else{
            return "Kobieta";
        }
    }
    else{
        return "-";
    }
}
```

int pesel::checkSum(){
/*Jedenasta cyfra jest cyfrą kontrolną, służącą do wychwytywania przekłamań numeru.
Jest ona generowana na podstawie pierwszych dziesięciu cyfr.
Aby sprawdzić czy dany numer PESEL jest prawidłowy, należy, zakładając, że litery a-j
to kolejne cyfry numeru od lewej, obliczyć wyrażenie:

$9 \times a + 7 \times b + 3 \times c + 1 \times d + 9 \times e + 7 \times f + 3 \times g + 1 \times h + 9 \times i + 7 \times j$

Jeżeli ostatnia cyfra otrzymanego wyniku nie jest równa cyfrze kontrolnej, to znaczy,
że numer zawiera błąd*/

```
    int sum = 9 * PESEL[0] +
    7 * PESEL[1] +
    3 * PESEL[2] +
    1 * PESEL[3] +
    9 * PESEL[4] +
    7 * PESEL[5] +
    3 * PESEL[6] +
    1 * PESEL[7] +
    9 * PESEL[8] +
    7 * PESEL[9];
    sum = sum % 10;

    if (sum == PESEL[10]) return 1;
    else return 0;
}
```

```

int pesel::checkMonth(){
    int month = getBirthMonth();

    if (month > 0 && month < 13){
        return 1;
    }
    else{
        return 0;
    }
}

int pesel::checkDay(){
    int year = getBirthYear();
    int month = getBirthMonth();
    int day = getBirthDay();
    if ((day > 0 && day < 32) &&
        (month == 1 || month == 3 || month == 5 ||
         month == 7 || month == 8 || month == 10 ||
         month == 12)) {
        return 1;
    }
    else
        if ((day > 0 && day < 31) &&
            (month == 4 || month == 6 || month == 9 ||
             month == 11)) {
                return 1;
            }
    else
        if ((day > 0 && day < 30 && leapYear(year)) || // warunk co do roku
            przestepnego
            (day > 0 && day < 29 && !leapYear(year))) {
                return 1;
            }
    else{
        return 0;
    }
}

int pesel::leapYear(int year){
    if (year % 4 == 0 && year % 100 != 0 || year % 400 == 0)
        return 1;
    else
        return 0;
}

```

```

void pesel::PeselValidator(char *PESELNumber){
    int i;

    if (strlen(PESELNumber) != 11){
        valid = 0;
    }
    else{
        for (i = 0; i < 11; i++){
            PESEL[i] = PESELNumber[i] - 48; //char to int, wg. tablicy ASCII
            aby tego dokonaoć odejmujemy 48
        }
        if (checkSum() && checkMonth() && checkDay()){
            valid = 1;
        }
        else{
            valid = 0;
        }
    }
}
}

```

skrypt__python.py

```

#!/usr/bin/python
# -*- coding: utf-8 -*-
import os, sys
import time # timestamp
import datetime # timestamp
import webbrowser # przeglądarka
import os # pliki
import shutil # foldery

for file in os.listdir("backup"): # wszystkie pliki w folderze backup
    if file.endswith(".html"): # jeśli plik jest rozszerzenia html
        lastBackup = file # to przypisz go do zmiennej lastBackup, ostatni plik
        to ostatni backup
        backupExists = True # ustaw flage, że backup istnieje

try: # sprawdź czy zmienna lastBackup istnieje
    lastBackup
except NameError: # jeśli nie
    backupExists = False # ustaw flage, że to będzie pierwszy backup

outFiles = [] # pusta lista przechowująca wszystkie indeksy

```

```

for file in os.listdir("out"): # wszystkie pliki w folderze out
    if file.startswith("out") and file.endswith(".txt"): # jesli plik jest o
        nazwie out*.txt
        outFiles.append(file) # to dopisz go do listy
        lastOut = True # ustaw flage, ze ostatni plik wyjscia istnieje

for i in range(len(outFiles)): # dla wszystkich plikow out nadpisz nazwe indeksami
    outFiles[i] = int(outFiles[i][3:len(outFiles[i])-4]) # outx.txt - usuwamy 3
    pierwsze i 4 ostatnie, rzutowanie int

if not len(outFiles): # jesli lista plikow wyjsciowych jest pusta
    lastOut = False # ustaw flage, ze zaden plik wyjsciowy nie istnieje

if lastOut: # jesli plik wyjsciowy istnieje
    index = max(outFiles) # to ustaw index jako maksymalny

ts = time.time() # timestamp
st = datetime.datetime.fromtimestamp(ts).strftime("%Y-%m-%d_%H-%M-%S") # konwersja
timestampu na date

fileOut = open("backup/" + st + ".html", "a+") # stworz nowy plik backup z data

if backupExists: # jesli jakis backup istnieje
    fileTemp = open("backup/" + lastBackup).read() # skopiuj zawartosc poprzedniego
    backupu do zmiennej tymczasowej
    shutil.copyfile("backup/" + lastBackup, "backup/" + st + ".html") # skopiuj
    ostatni backup do nowego pliku
    while index: # poki dzialamy na indeksach (od ostatniego indeksu wyjscia do 1)
        iTemp = "<tr id=" + str(index) # string oznaczajacy wpis wiersza w HTML
        o podanym indeksie (np. dla out3 to 3)
        if iTemp in fileTemp: # jesli znalazl
            indexTemp = fileTemp.index(iTemp) + 7 # pomin <tr id= i przejdz do liczby
            i = int(fileTemp[indexTemp:indexTemp + len(str(index))]) + 1 # index + 1,
            break
        index -= 1 # a jesli nie znalazl to zmniejsza index (czyli sprawdza out2,
        wedlug przykladu)
    del fileTemp # zwolnij pamiec, tej zmiennej nie bedziemy juz uzywac
    del iTemp # zwolnij pamiec, tej zmiennej nie bedziemy juz uzywac
    del indexTemp # zwolnij pamiec, tej zmiennej nie bedziemy juz uzywac
    del index # zwolnij pamiec, tej zmiennej nie bedziemy juz uzywac

else: # jesli jednak backup nie istnieje
    fileOut.write("<html><head><meta charset='utf-8' />"
    "<link rel='Stylesheet' href='../css/style.css'><script>var url = window.location.

```

```

pathname; "
    "var file = url.substring(url.lastIndexOf('/')+1);"
    "filename = file.substr(0, file.length - 5);"
    "document.title = 'Backup ' + filename;</script><meta charset='UTF-8'>"
    "</head><body><h1>"
    "<script>document.write(document.title)</script></h1>"
    "<h2>Dawid Bitner IA</h2>"
    "<h3>parser PESEL - jezyki skryptowe - projekt</h3>"
    "<table>"
    "    <tr><th>Input</th><th>Output</th></tr>)" # to stworz nowy plik HTML
    i = 1 # i ustaw zmienna indeksu na 1

while True:
    try: # sprawdz czy sie pliki otworza
        fileInput = open("in/in" + str(i) + ".txt", "r")
        fileOutput = open("out/out" + str(i) + ".txt", "r")
    except FileNotFoundError as e: # warunek stopu, jesli pliki sie nie otworza to
        konczy program
        break

    fileOut.write("<tr id=" + str(i) + "><td style='width:10%; text-align:center'>"
    + fileInput.readline() + "</td><td>"
    + fileOutput.read()+"</td></tr>)" # wpisz zawartosc plikow in i out

    i += 1 # powieksz iterator
    fileInput.close() # zamknij plik wejscowy
    fileOutput.close() # zamknij plik wyjsciowy

fileOut.close() # jesli warunek stopu zostanie spelniony (wyjdzie z petli) to zamknij
plik backup
webbrowser.open("file://" + os.path.realpath(fileOut.name)) # otworz przegladarke

```