

## Projekt z podstaw programowania

Wykonały: Natalia Starzyk (096951) i Julia Wesołowska (096604)

Wydział: Zarządzania i Modelowania Komputerowego

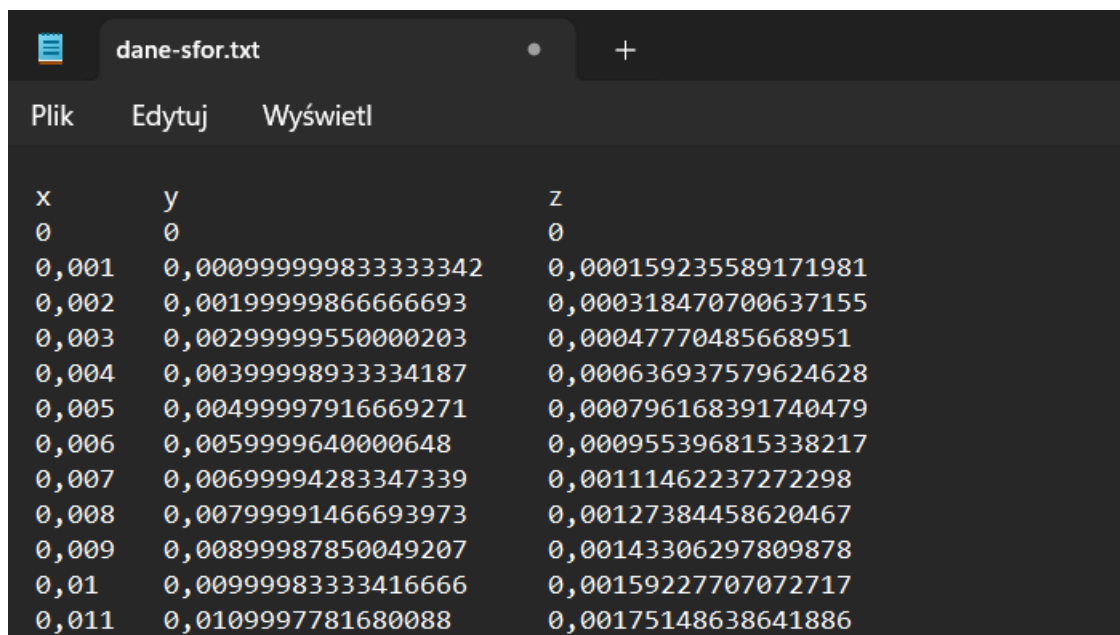
Kierunek: Inżynieria danych, I rok, stacjonarne

### Zadanie 3. „Niesforne dane”

Aby wykonać zadanie trzeba użyć następujących komend w podanej kolejności:

- `unzip dane.zip`
- `dos2unix dane.txt`
- `paste -d "\t" - - - <dane.txt >dane-sfor.txt` (z posortowanych danych tworzy nowy plik)
- `sed -i '1i x\ty\tz' dane-sfor.txt` (dodaje nagłówki x y z)
- `unix2dos dane-sfor.txt`

Rezultat:



x	y	z
0	0	0
0,001	0,00099999983333342	0,000159235589171981
0,002	0,00199999866666693	0,000318470700637155
0,003	0,00299999550000203	0,00047770485668951
0,004	0,00399998933334187	0,000636937579624628
0,005	0,00499997916669271	0,000796168391740479
0,006	0,0059999640000648	0,000955396815338217
0,007	0,00699994283347339	0,00111462237272298
0,008	0,00799991466693973	0,00127384458620467
0,009	0,00899987850049207	0,00143306297809878
0,01	0,00999983333416666	0,00159227707072717
0,011	0,0109997781680088	0,00175148638641886

### Zadanie 4. „Dodawanie poprawek”

Aby wykonać zadanie trzeba użyć następujących komend w podanej kolejności:

- `unzip lista.zip`
- `dos2unix lista.txt lista-pop.txt`
- `diff -u lista.txt lista-pop.txt > poprawki.patch` (porównuje oba pliki i zapisuje różnice w pliku patch)
- `patch lista.txt < poprawki.patch` (aplikowanie łatki)
- `md5sum lista-pop.txt lista.txt` (porównanie sumy kontrolnej)

Rezultat:

Obliczone sumy wyszły takie same, a więc poprawki zostały poprawnie dodane.

```
natal@Naskolot MSYS ~  
$ md5sum lista-pop.txt lista.txt  
683c1c85343c7337adfb13acb7598237 *lista-pop.txt  
683c1c85343c7337adfb13acb7598237 *lista.txt
```

Zadanie 5. „Z CSV do SQL i z powrotem”

Aby wykonać zadanie trzeba użyć następujący komend w podanej kolejności:

- unzip csv.zip
- awk -F ";" 'NR > 1 { printf "INSERT INTO stepsData (time, intensity, steps) VALUES (%s, %s, %s);\n", \$1, \$2, \$3 }' steps-2sql.csv > steps-2sql.sql
- cat steps-2csv.sql | grep 'VALUES' | awk -F'[(,)]' '{if (\$6 ~ /000\$/) {sub(/000\$/, "", \$6)}; gsub(/^ +/, "", \$7); gsub(/^ +/, "", \$8); print \$6";"\$7";"\$8}' > steps-2csv.csv
- sed -i '1i dateTime;steps;synced' steps-2csv.csv

Rezultat:

Po odpowiednim sformatowaniu otrzymaliśmy pliki, których wygląd jest przedstawiony poniżej.

```
natal@Naskolot MSYS ~  
$ head steps-2sql.sql  
INSERT INTO stepsData (time, intensity, steps) VALUES (1562001120, 19, 0);  
INSERT INTO stepsData (time, intensity, steps) VALUES (1562001180, 23, 0);  
INSERT INTO stepsData (time, intensity, steps) VALUES (1562001240, 13, 0);
```

```
natal@Naskolot MSYS ~  
$ head steps-2csv.csv  
dateTime;steps;synced  
1562004600;41;0  
1562005200;65;0  
1562005800;86;0  
1562006400;163;0  
1562007000;234;0  
1562007600;258;0  
1562008800;385;0  
1562009400;757;0  
1562010000;829;0
```

## Zadanie 6. “Marudny tłumacz”

Aby wykonać zadanie trzeba użyć następujących komend w podanej kolejności:

- unzip tłumacz.zip
- awk ‘{ if (\$0 ~ /^ \*"/) { print "// " \$0}; print \$0 } else {print \$0 } }’ en-7.2.json5 > pl-7.2.json5
- sort en-7.2.json5 > sorted-7.2.txt
- sort en-7.4.json5 > sorted-7.4.txt
- comm -13 sorted-7.2.txt sorted-7.4.txt > new-entries.txt
- awk ‘{ if (\$0 ~ /^ \*"/) { print "// " \$0; print \$0 } }’ new-entries.txt > pl-7.4.json5

Rezultat:

```
natal@Naskolot MSYS ~
$ head pl-7.2.json5
{
// "401.help": "You're not authorized to access this page. You can use the button below to get back to the home page.",
"401.help": "You're not authorized to access this page. You can use the button below to get back to the home page.",
// "401.link.home-page": "Take me to the home page",
"401.link.home-page": "Take me to the home page",
// "401.unauthorized": "unauthorized",
"401.unauthorized": "unauthorized",
```

```
natal@Naskolot MSYS ~
$ head pl-7.4.json5
// "access-status.embargo.listelement.badge": "Embargo",
"access-status.embargo.listelement.badge": "Embargo",
// "access-status.metadata.only.listelement.badge": "Metadata only",
"access-status.metadata.only.listelement.badge": "Metadata only",
// "access-status.open.access.listelement.badge": "Open Access",
"access-status.open.access.listelement.badge": "Open Access",
// "access-status.restricted.listelement.badge": "Restricted",
"access-status.restricted.listelement.badge": "Restricted",
// "access-status.unknown.listelement.badge": "Unknown",
"access-status.unknown.listelement.badge": "Unknown",
```

## Zadanie 7. “Fotografik gamoń”

Aby wykonać zadanie trzeba wykonać następujące kroki:

- Przejdź do katalogu zawierającego archiwa ze zdjęciami: cd zad7
- Sprawdź, czy znajdują się tam jedynie dwa pliki .zip  
ls \*.zip
- Stwórz katalog o nazwie obrazy: mkdir obrazy
- Otwórz katalog: cd obrazy
- Stwórz katalog o nazwie wypakowane: mkdir wypakowane
- Otwórz katalog: cd wypakowane

- Rozpakuj wszystkie archiwa zip do folderu obrazy/wypakowane:  

```
find . -name "*.zip" -exec unzip -d obrazy/wypakowane {} \;
```

```
cd wypakowane
```

```
rm *.zip
```

```
for f in *.png; do convert "$f" "${f%}.jpg"; done
```
- Wyszukaj pliki .png:  

```
ls *.png
```
- Przekonwertuj pliki PNG do formatu JPG:  

```
for f in .png; do
```

```
  convert "$f" "${f%}.jpg"
```

```
done
```
- Usuń oryginalne pliki .png, które już zostały przekonwertowane:  

```
rm *.png
```
- Dostosuj rozmiar i parametry JPG-ów zgodnie z wymaganiami:  











```
for f in *.jpg; do magick "$f" -resize x720 -density 96 -units PixelsPerInch "$f"
```

```
done
```
- Spakuj zmodyfikowane obrazy do archiwum zip razem z katalogiem  
Wypakowane:  

```
zip -r obrazy.zip wypakowane/
```

Rezultat:

W ramach zadania 7 rozpakowano archiwa ZIP ze zdjęciami, przekonwertowano pliki PNG do formatu JPG, przeskalowano je do wysokości 720 px i zapisano w folderze „Wypakowane”, a następnie całość spakowano jako „obrazy.zip”.

Nazwa	Typ	Rozmiar po skompr...	Chronione ...	Rozmiar	Stopień	Data modyfikacji
 adrien-olichon-3137064	Plik JPG	111 KB	Nie	113 KB	2%	15.05.2025 20:27
 aleksandr-slobodanyk-989941	Plik JPG	1 015 KB	Nie	1 026 KB	2%	15.05.2025 20:27
 aleksandr-slobodanyk-989946	Plik JPG	1 222 KB	Nie	1 232 KB	1%	15.05.2025 20:27
 alexander-dummer-37646-1356304	Plik JPG	70 KB	Nie	71 KB	1%	15.05.2025 20:27
 alexazabache-5117913	Plik JPG	39 KB	Nie	40 KB	2%	15.05.2025 20:27
 alex-montes-892479-1820563	Plik JPG	42 KB	Nie	43 KB	2%	15.05.2025 20:27
 anni-roenkae-3435272	Plik JPG	74 KB	Nie	75 KB	2%	15.05.2025 20:27
 anntarazevich-5620861	Plik JPG	45 KB	Nie	46 KB	2%	15.05.2025 20:27
 archwall_dark_blue.png	Plik JPG	15 KB	Nie	22 KB	33%	15.05.2025 20:27
 archwall_dark_orange.png	Plik JPG	15 KB	Nie	22 KB	33%	15.05.2025 20:27

Zadanie 8. „Wszędzie te PDF-y”

Aby wykonać zadanie trzeba wykonać następujące kroki:

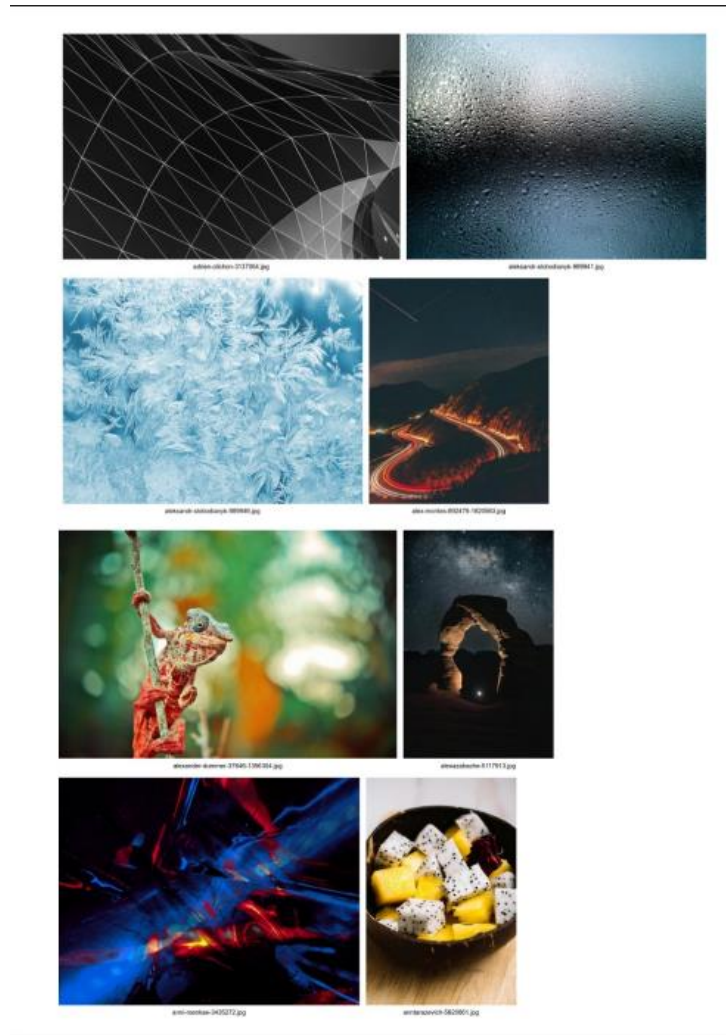
- Otwórz katalog zawierający wcześniej wypakowane i przekonwertowane na format .jpg pliki graficzne.
- Zweryfikuj, czy praca odbywa się w środowisku MSYS2 MINGW, które jest wymagane do poprawnego działania używanych narzędzi.
- Za pomocą pętli oraz narzędzia magick dodaj podpisy do zdjęć:

```
for f in *.jpg; do
    magick "$f" -gravity south -background white -splice 0x40 -fill black -
    pointsize 20 -annotate +0+5 "$f" "$f"
done
```

- Wszystkie obrazy zostały opatrzone podpisami umieszczonymi w dolnej części.
- Wykorzystaj polecenie montage (z pakietu ImageMagick) w celu wygenerowania pliku PDF zawierającego wszystkie zdjęcia:  
montage \*.jpg -tile 2x4 -geometry +10+10 portfolio.pdf
- W folderze roboczym pojawił się nowy dokument PDF o nazwie portfolio.pdf.
- W celu uporządkowania plików, utwórz katalog o nazwie Portfolio i przenieś do niego gotowy dokument:  
mkdir Portfolio  
mv portfolio.pdf Portfolio/
- Otwórz katalog zawierający wcześniej wypakowane i przekonwertowane na format .jpg pliki graficzne.
- Zweryfikuj, czy praca odbywa się w środowisku MSYS2 MINGW, które jest wymagane do poprawnego działania używanych narzędzi.
- Za pomocą pętli oraz narzędzia magick dodaj podpisy do zdjęć:  
for f in \*.jpg; do  
 magick "\$f" -gravity south -background white -splice 0x40 -fill black -  
 pointsize 20 -annotate +0+5 "\$f" "\$f"  
done
- Wszystkie obrazy zostały opatrzone podpisami umieszczonymi w dolnej części.
- Wykorzystaj polecenie montage (z pakietu ImageMagick) w celu wygenerowania pliku PDF zawierającego wszystkie zdjęcia:  
montage \*.jpg -tile 2x4 -geometry +10+10 portfolio.pdf
- W folderze roboczym pojawił się nowy dokument PDF o nazwie portfolio.pdf.
- W celu uporządkowania plików, utwórz katalog o nazwie Portfolio i przenieś do niego gotowy dokument:  
mkdir Portfolio  
mv portfolio.pdf Portfolio/

Rezultat:

W zadaniu 8 do zdjęć dodano podpisy z nazwami plików i wygenerowano plik PDF „portfolio.pdf” zawierający siatkę obrazów, który przeniesiono do folderu „Portfolio”.



## Zadanie 9. „Porządki w kopiach zapasowych”

Aby wykonać zadanie trzeba wykonać następujące kroki:

- Otwórz katalog zawierający pliki kopie-1.zip oraz kopie-2.zip (komenda: `cd Zadanie_9`).
- Sprawdź zawartość katalogu za pomocą polecenia:  
`ls *.zip`
- Rozpakuj oba pliki archiwów ZIP:  
`unzip kopie-1.zip`  
`unzip kopie-2.zip`
- W katalogu z wypakowanymi plikami utwórz plik skryptu:  
`touch data_sort.sh`
- Skrypt otwórz w edytorze tekstowym nano:  
`nano data_sort.sh`
- W edytorze wprowadź następującą zawartość skryptu:  
`#!/bin/bash`  
`for file in *.zip; do`  
`rok=${file:0:4}`

- ```

mies=${file:5:2}
mkdir -p Posortowane/$rok/$mies
mv "$file" Posortowane/$rok/$mies/
done

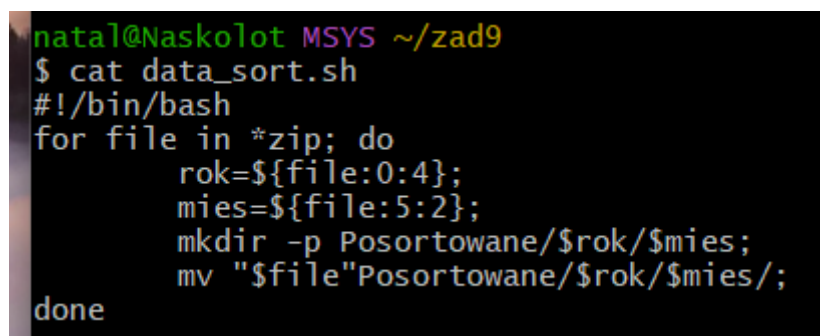
```
- Po zapisaniu skryptu kombinacją CTRL + O, po wciśnięciu ENTER oraz wyjściu z edytora (CTRL + X), nadano plikowi prawo do uruchamiania:

```
chmod u+x data_sort.sh
```
  - Skrypt został uruchomiony:

```
./data_sort.sh
```
  - W wyniku działania skryptu pliki .zip zostały uporządkowane i przeniesione do folderu Posortowane, w którym zostały pogrupowane według roku i miesiąca (liczbowo).

Rezultat:

W zadaniu 9 rozpakowano pliki kopii zapasowych i uruchomiono skrypt, który uporządkował je automatycznie według roku i miesiąca w strukturze katalogów „Posortowane”.



```

natal@Naskolot MSYS ~/zad9
$ cat data_sort.sh
#!/bin/bash
for file in *.zip; do
    rok=${file:0:4};
    mies=${file:5:2};
    mkdir -p Posortowane/$rok/$mies;
    mv "$file" Posortowane/$rok/$mies/;
done

```

Zadanie 10. „Galeria dla grafika”

Aby wykonać zadanie trzeba wykonać następujące kroki:

- Otwórz katalog zawierający obrazy (wypakowane i wcześniej przekonwertowane na format .jpg).
- Utwórz nowy plik skryptu o nazwie obrazy\_html.sh za pomocą polecenia:

```
touch obrazy_html.sh
```
- Skrypt otwórz w edytorze tekstowym nano:

```
nano obrazy_html.sh
```
- Do pliku wprowadź następujący kod:

```
echo '<div class="responsive">' > galeria.html
for file in *.jpg; do
    echo " <div class=\"gallery\">" >> galeria.html
    echo "  <a target=\"_blank\" href=\"$file\">" >> galeria.html
    echo "    <img src=\"$file\">" >> galeria.html
    echo "  </a>" >> galeria.html
    echo " <div class=\"desc\">$file</div>" >> galeria.html
done

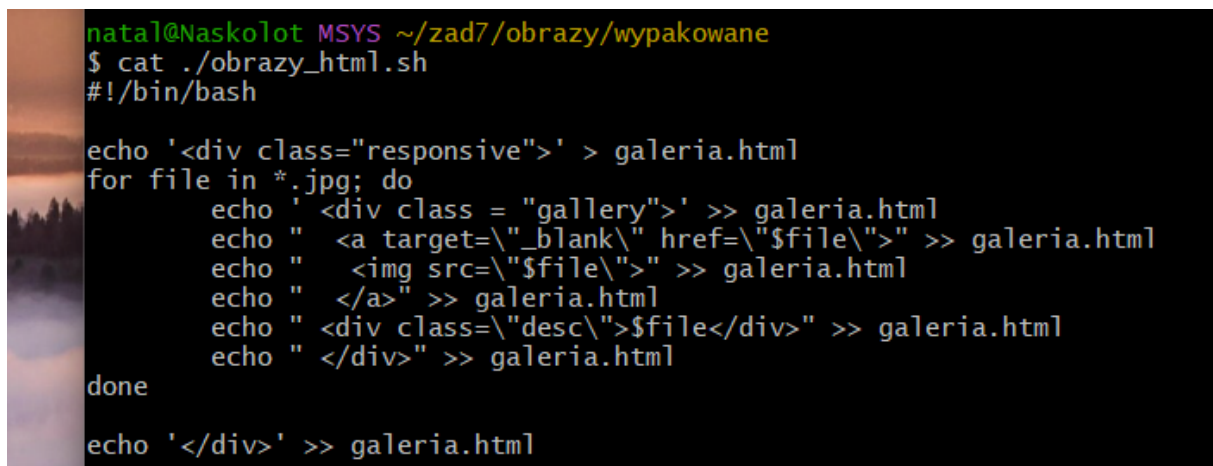
```

```
echo " </div>" >> galeria.html
done
echo " </div>" >> galeria.html
```

- Zmiany zapisz przy pomocy kombinacji klawiszy CTRL + O, zatwierdź klawiszem ENTER, a następnie zamknij edytor (CTRL + X).
- Nadaj plikowi skryptowemu uprawnienia do uruchamiania:  
chmod u+x obrazy\_html.sh
- Skrypt uruchom poleceniem:  
./obrazy\_html.sh
- W wyniku działania skryptu, w folderze z obrazami wygenerowano plik galeria.html, zawierający uporządkowaną strukturę galerii – z nagłówkami i podpisami odpowiadającymi nazwom plików graficznych.

Rezultat:

Zadanie 10 zakończono utworzeniem pliku „galeria.html” prezentującego wszystkie obrazy JPG w postaci prostej, opisaniej galerii gotowej do wyświetlenia w przeglądarce.

A terminal window with a dark background and a vertical image strip on the left. The terminal shows the execution of a shell script named obrazy\_html.sh. The script uses echo and cat commands to create an HTML file named galeria.html. It iterates over files in the current directory with a .jpg extension, generating HTML code for each image, including a link and a description. The script ends with a final echo statement to close the main div.

```
natal@Nasko1ot MSYS ~/zad7/obrazy/wypakowane
$ cat ./obrazy_html.sh
#!/bin/bash

echo '<div class="responsive">' > galeria.html
for file in *.jpg; do
    echo '  <div class = "gallery">' >> galeria.html
    echo "    <a target=\"_blank\" href=\"$file\">" >> galeria.html
    echo "      <img src=\"$file\">" >> galeria.html
    echo "    </a>" >> galeria.html
    echo "  <div class=\"desc\">$file</div>" >> galeria.html
    echo "  </div>" >> galeria.html
done

echo '</div>' >> galeria.html
```