Jan Rusinek

PROGRAMOWANIE HYBRYDOWE z PRZYKŁADAMI ZASTOSOWANIA w DYDAKTYCE

[**słowa kluczowe.** Algorytmy, informatyzacja procesu dydaktycznego, programowanie hybrydowe, pascal, TeX, PHP, arkusz kalkulacyjny]

Streszczenie

W pracy omawiane jest zastosowanie programowania hybrydowego do tworzenia zadań domowych lub egzaminacyjnych, jak również do tworzenia i opracowywania ankiet. Pokazane są możliwości „współpracy" takich języków programowania jak Pascal, PHP, TeX. arkusz kalkulacyjny czy „język" używany przez pliki wsadowe.

1. Wstęp

Przez programowanie hybrydowe rozumiemy przygotowanie programu albo algorytmu z użyciem dwóch lub więcej języków programowania.

Najbardziej oczywista sytuacja, kiedy trzeba użyć programowania hybrydowego to taka, kiedy jeden fragment algorytmu jest dostępny tylko w jednym języku, a drugi tylko w innym. z taką sytuacją będziemy mieli do czynienia np. kiedy odpowiedni program graficzny (np. TeX) potrafi złożyć poprawnie wzór matematyczny, ale nie potrafi dokonać odpowiednich obliczeń, natomiast typowy język programowania bez problemu dokonuje obliczeń matematycznych bądź numerycznych, a ma kłopoty ze składem.

Bywa też tak, że w każdym z języków można wykonać dany projekt w całości, ale rozdzielenie go na różne języki daje lepszy efekt, lub oszczędza pracę. Tak może się zdarzyć kiedy na przykład mamy już gotowe algorytmy lub ich fragmenty napisane w różnych językach i tłumaczenie ich na jeden wspólny język zabrałoby sporo czasu (nie mówiąc już o możliwości zrobienia w trakcie tego tłumaczenia błędów).

Projekt może być przygotowywany przez kilku programistów i każdy specjalizuje się w innym języku.

W pracy zaprezentowano kilka sytuacji, kiedy zastosowanie programowania hybrydowego pozwoliło w prosty sposób rozwiązać postawiony problem. W przykładach zawierających zadania matematyczne, zadania te są stosunkowo proste i pomijana jest część teoretyczna, aby treści matematyczne nie przesłaniały głównego celu pracy - zademonstrowania użyteczności programowania hybrydowego.

2. Pascal i TeX

Przypuśćmy, że mamy utworzyć następujące zadanie

Wyznacz z dokładnością dwóch miejsc po przecinku punkt z przedziału [1;100]*, w którym funkcja*

$$f(x) = ae^{bx} - cx^2$$

osiąga najmniejszą wartość w tym przedziale.

Chcemy, aby parametry a, b, c były losowane z odpowiednich przedziałów dając wiele różnych zestawów tego zadania, i aby rozwiązania wszystkich zestawów drukowały się w odpowiednim miejscu (np. na osobnej stronie).

Aby rozwiązać zadanie trzeba najpierw pokazać, że pochodna funkcji f, czyli funkcja

$$f'(x) = abe^{bx} - 2cx$$

ma tylko jedno miejsce zerowe w tym przedziale (można to zrobić np. wykorzystując drugą pochodną). To jedyne miejsce zerowe będzie szukanym rozwiązaniem.

Do wydruku treści zadania użyjemy TeX-a, który jest niezastąpiony przy składzie matematycznych wzorów, ale nie za bardzo radzi sobie z obliczeniami. Wyposażenie go w takie możliwości (patrz np. [10]) wymaga sporo dodatkowego wysiłku, podczas kiedy w typowych językach programowania mamy to już gotowe. Dlatego proponowane jest następujące rozwiązanie tego zadania.

Programik napisany w pascalu losuje daną a jako liczbę całkowitą z przedziału [2;7], b jako liczbę dziesiętną z dokładnością do jednego miejsca po przecinku z przedziału [0,1; 0,5], i c jako liczbę całkowitą z przedziału [11; 21], oblicza miejsce zerowe powstałej w ten sposób funkcji (wykorzystując najprostszy algorytm) i zapisuje to wszystko w odpowiednim pliku `dane.ttt`. Oto ten programik:

```
function g(x:real):real;
begin
g:=a*b*exp(b*x)-2*c*x;
```

```
end;

procedure oblicz;
begin
l:=0;r:=200;while r-l>0.0001 do begin
d:=(l+r)/2;
if g(d)=0 then begin l:=d;r:=d;end;
if g(d)<0 then l:=d; if g(d)>0 then r:=d;
end;end;

begin
randomize;
assign(fi,'dane.ttt');rewrite(fi);
for n:=1 to 100 do begin
a:=random(5)+2;a1:=(random(4)+1);b:=a1/10;
c:=random(10)+11;oblicz;
writeln(fi,'\dane{',a,'}{','0{,}',a1,'}{',c,'}{',d:4:2,'}');
end;close(fi);end.
```

Przy trzech zestawach plik `dane.ttt` wygląda zatem następująco:

```
\dane{5}{0{,}4}{13}{12.78}
\dane{3}{0{,}3}{17}{22.48}
\dane{2}{0{,}3}{15}{23.57}
```

Teraz wystarczy napisać algorytm w TeX-u, który odpowiednio zinterpretuje zapisane dane. Oto on:

```
\def\dane#1#2#3#4{\advance\nn1{\bf Zestaw \the\nn.}
Wyznacz z~dokładnością dwóch miejsc po
przecinku punkt z~przedziału $[1{;}100]$, w~którym
funkcja $f(x)=#1 e^{#2 x}- #3x^2$ osiąga
najmniejszą wartość w~tym przedziale?\newpage}
\input{dane.ttt}
\def\dane#1#2#3#4{\advance\nn1 {\bf \the\nn.}  $x=#4$.\\}
\nn=0{\large\bf ODPOWIEDZI}
\input{dane.ttt}
```

Jak widać polecenie `\dane` jest definiowane dwukrotnie - za pierwszym razem pobiera ono dane a, b, c i drukuje zestawy zadań, a za drugim pobiera wynik.

3. TeX z arkuszem kalkulacyjnym

Następny problem jest podobny, tylko zadanie matematyczne, które chcemy otrzymać dotyczy statystyki matematycznej i potrzebne są do tego wartości kwantyli rozkładu **t**-studenta ([6]). Istnieją profesjonalne programy, które takie obliczenia wykonują. W tym artykule opieramy się wyłącznie na programach niekomercyjnych. Takim programem jest `Openoffice`, a konkretnie jego składnik **calc** - odpowiednik Excela.

Chcemy mianowicie w wielu wersjach uzyskać zadanie następujące:

Chcąc sprawdzić wagę pewnych owoców wylosowano 6 sztuk i otrzymano wagę w dekagramach: a_1, a_2, a_3, a_4, a_5, a_6. *Oblicz średnią oraz odchylenie standardowe z próby. Zakładając, że dane pochodzą z rozkładu normalnego wyznacz przedział ufności na poziomie ufności* $1-\alpha = x$.

Parametry a_i oraz x są losowane. Podobnie jak w poprzednim problemie chcemy, aby na początku wydrukowały się treści zadań, a potem odpowiedzi.

Proponowane rozwiązanie jest następujące. Losowanie parametrów a_i oraz x przeprowadzamy TeXem zapisując wyniki do odpowiedniego pliku `csv`. Możemy to zrobić na przykład następującym algorytmem (wykorzystujemy pakiet `random.tex`).

```
\newwrite\zzzttt
\newcount\xa\newcount\xb\newcount\xc
\newcount\xd\newcount\xe\newcount\xf
\newcount\ufnosc\newcount\lewy\newcount\prawy
\lewy=21\prawy=99
\def\losuj{
\setrannum{\xa}{\lewy}{\prawy}
\setrannum{\xb}{\lewy}{\prawy}
\ifnum\xa=\xb\advance\xb1\else\fi
\setrannum{\xc}{\lewy}{\prawy}
\setrannum{\xd}{\lewy}{\prawy}
\setrannum{\xe}{\lewy}{\prawy}
\setrannum{\xf}{\lewy}{\prawy}
\setrannum{\ufnosc}{91}{99}}
\newcount\ile
\immediate\openout\zzzttt=aaa.csv
\ile=0
\whiledo{\ile<30}{\advance\ile1
\losuj
\immediate\write\zzzttt{\string\aa [\the\xa [\the\xb
```

```
[\the\xc [\the\xd[\the\xe[\the\xf
[0,\the\ufnosc
[\string\bb
[=\string średnia(b\the\ile:g\the\ile)
[=wariancja(b\the\ile:g\the\ile)
[=pierwiastek(k\the\ile)
[=rozkład.t.odw(2*(1-h\the\ile);5)
[=M\the\ile*L\the\ile/pierwiastek(5)
[=j\the\ile-n\the\ile
[=j\the\ile+n\the\ile
[]}}
\closeout\zzztttt
```

Otrzymujemy plik tekstowy `aaa.csv`, który np. przy trzech wersjach wygląda następująco:

Pierwsza linia:

```
\aa[89[87[34[61[60[43[0,98[\bb[=Średnia(b1:g1) [=wariancja(b1:g1)
[=pierwiastek(k1) [=rozkład.t.odw(2*(1-h1);5)
[=M1*L1/pierwiastek(5) [=j1-n1[=j1+n1[]
```

Druga linia:

```
\aa[57[31[88[43[24[97[0,92[\bb[=średnia(b2:g2) [=wariancja(b2:g2)
[=pierwiastek(k2) [=rozkład.t.odw(2*(1-h2);5)
[=M2*L2/pierwiastek(5) [=j2-n2[=j2+n2[]
```

Trzecia linia:

```
\aa[73[39[39[45[65[88[0,93[\bb[=Średnia(b3:g3) [=wariancja(b3:g3)
[=pierwiastek(k3) [=rozkład.t.odw(2*(1-h3);5)
[=M3*L3/pierwiastek(5) [=j3-n3[=j3+n3[]
```

Otwieramy go programem `calc` wybierając znak „[” jako separator pola i rezygnujemy z separacji tekstu. Wtedy w kolumnach od `J` do `P` pojawią się kolejno wyliczone dla danych z kolumn `B - H`: średnia, wariancja, odchylenie standardowe, kwantyl rozkładu Studenta dla parametru z kolumny `H`, l równe połowie długości przedziału ufności, lewy koniec przedziału ufności, prawy koniec przedziału ufności.

Zapisujemy to w nowym pliku tekstowym `csv` zachowując wybrane parametry i otrzymujemy plik następujący (nazwijmy go np. `bbb.csv`):

```
\aa[89[87[34[61[60[43[0,98[\bb[62,33[500,67[22,38[2,76[27,58[34,75[89,92[]
\aa[57[31[88[43[24[97[0,92[\bb[56,67[904,27[30,07[1,65[22,18[34,49[78,85[]
\aa[73[39[39[45[65[88[0,93[\bb[58,17[412,97[20,32[1,75[15,93[42,24[74,1[]
```

Teraz wystarczy zastosować następujący algorytm TeX-a:

```
\ile=0
\def\aa [#1[#2[#3[#4[#5[#6[#7[]{
\advance\ile1
ZESTAW \the\ile. Chcąc sprawdzić wagę pewnych owoców wylosowano
6 sztuk i otrzymano wagę w~dekagramach: #1, #2, #3, #4, #5, #6.
Oblicz średnią oraz odchylenie standardowe z~próby. Zakładając, że
dane pochodzą z rozkładu normalnego wyznacz przedział ufności na
poziomie ufności $1-\alpha=#7$.
\newpage}
\def\bb[#1[#2[#3[#4[#5[#6[#7[]{}
\input{bb.csv}
\def\aa[#1[#2[#3[#4[#5[#6[#7[{}
\ile=0\parindent=0em
\def\bb[#1[#2[#3[#4[#5[#6[#7[]{\advance\ile1{\bf \the\ile.}
średnia=#1, odchylenie standardowe=#3, $P=\langle#6;#7\rangle$\par}
\input{bb.csv}
```

Przy pierwszym odczycie pliku są wczytywanie tylko dane pomiędzy `\aa` i `\bb` - dane po siódmym znaku [(czyli od `\bb` do `[]` są „połykane", a przy drugim odczycie tylko dane pomiędzy `\bb` i `[]`.

4. Polecenie `\write18` w TeX-u

Od kilku lat dostępne są wersje TeX-a z możliwościami uruchamiania "programów zewnętrznych". Do tego celu służy plecenie `write18`, które działa następująco:

Jeśli chcemy aby w danym momencie TeX, zanim przejdzie do składania następnej linii, wywołał jakiś program zewnętrzny uruchamiany z linii poleceń - nazwijmy go na przykład `program.exe` wypisujemy następujące polecenie:

`\write18{program.exe}`

Żeby to zadziałało trzeba TeX-a uruchomić z parametrem `--shell-escape`. Dobrze jest zatem mieć wersję TeX-a uruchamianą z linii poleceń a nie tylko „przy pomocy myszy"[1]. Najwygodniej napisać sobie odpowiedni plik wsado-

[1]Prawdopodobnie wszystkie wersje TeX-a taką możliwośc mają, choć nie wszyscy użytkownicy o tym wiedzą i z tego korzystają.

wy, który uruchamia TeX-a z tym parametrem np.
`call pdflatex --shell-escape %1.tex`
i umieścić go w folderze zmiennej środowiskowej `PATH` (np. w katalogu `WINDOWS`).

Program, który będziemy wywoływać powinien być też widoczny z dowolnego miejsca, albo umieszczony w tym samym folderze co plik TeX-a, który będziemy kompilować.

Zademonstrujemy to na przykładzie języka `PHP`.

Instrukcję jak skonfigurować komputer, aby skrypty `php` uruchamiać z linii poleceń z dowolnego miejsca można znaleźć np. w [12].

Przypuśćmy, że chcemy w 100 wersjach przygotować zadanie podobne do już rozpatrywanego:

Wyznacz z dokładnością 0,01 *największą wartość funkcji*

$$f(x) = ax^2 - e^x$$

w przedziale $[0;6]$ *oraz z tą samą dokładnością punkt, w którym ta wartość jest przyjmowana,*

gdzie a jest liczbą dziesiętną (niecałkowitą!) z przedziału $[3{,}11; 19{,}99]$.

Najpierw tworzymy następujący algorytm TeX-a.

```
\ile=0
\newwrite\zzzttt
\whiledo{\ile<100}{\advance\ile1
\setrannum{\aa}{3}{19}\setrannum{\bb}{1}{99}
\newpage
{\bf Zestaw \the\ile.} Wyznacz z~dokładnością $0{,}01$
największą wartość funkcji
$$f(x)=\the\aa{,}\ifnum\bb<\edef\cc{\the\aa.\0\the\bb}\else
\edef\cc{\the\aa.\the\bb}\fi \cc x^2-e^x$$
w przedziale $[0;6]$ oraz
punkt, w~którym ta wartość jest przyjmowana.
\immediate\openout\zzzttt=mu.ttt
\immediate\write\zzzttt{\cc}
\immediate\closeout\zzzttt
\immediate\write18{php yyy.php}}
\newpage
\ile=0
{\Large\bf ODPOWIEDZI}
\input{max.ttt}
```

Mamy tu 100 krotny przebieg. w każdym przebiegu jest losowana część całkowita liczby a oraz jej część po przecinku. Jest to zapisywane jako liczba dziesiętna do pliku `mu.ttt`. Potem jest uruchamiany plik `yyy.php`, który pobiera daną z pliku `mu.ttt` i rozwiązuje nasze zadanie dopisując kolejne wyniki do pliku `max.ttt`.

A oto cały plik `yyy.php`

```
<?
$fp=fopen("mu.ttt","r");
$aa=fgets($fp, 255);
fclose($fp);
function ff($a,$x)
{return $a*$x*$x-exp($x);}
$h=0.0001;
$xx=0;$xxx=0;$max=-100;
while ($xx<=6){if (ff($aa,$xx)>$max)
{$max=ff($aa,$xx);$xxx=$xx;} else{}$xx=$xx+$h;}
$gp=fopen("max.ttt","a");
fputs($gp, "\pp");
fputs($gp, number_format($xxx, 2, ',',''));
fputs($gp, "qqq");
fputs($gp, number_format($max, 2, ',',''));
fputs($gp, "bbb\n");
fclose($gp);
?>
```

Algorytm działa w ten sposób, że oblicza wartości funkcji w przedziale $[0;6]$ przesuwając argument o $h = 0{,}0001$. z wzoru Lagrange'a ([7]) można łatwo wykazać, że jeśli szukane maksimum znajdzie się w jakimś przedziale długości h, to będzie się ono różnić od wartości na końcach o znacznie mniej niż 0,01.

Można też nie pisać skryptu `php` osobno, ale tworzyć go w czasie kompilacji TeX-a.

Może to być dogodne z kilku powodów. Pierwszy to taki, że cały projekt uruchamiamy „jednym poleceniem".

Drugi powód może być jeszcze bardziej przekonujący. Otóż możemy napisać makro TeX-owe robiące potrzebne obliczenia na liczbach dziesiętnych, które potem można wielokrotnie wykorzystywać. Oto propozycja takiego makra:

```
\newwrite\zzzttt
\gdef\oblicz#1\koniec{\xdef\wynik{#1}}
\def\dzialanie#1#2#3{
\immediate\openout\zzzttt=zzz.php\relax
\immediate\write\zzzttt{\string<?}
\immediate\write\zzzttt{$fp=fopen("zzz.ttt","w");}
\immediate\write\zzzttt{$x=#1;}
\immediate\write\zzzttt{$xx=#3;}
\immediate\write\zzzttt{$y=$x#2$xx;}
\immediate\write\zzzttt{fputs(\string$fp, "\string\oblicz");}
\immediate\write\zzzttt{fputs($fp, $y);}
\immediate\write\zzzttt{fputs($fp, "\string\koniec");}
\immediate\write\zzzttt{fclose($fp);}
\immediate\write\zzzttt{?\string>}
\immediate\closeout\zzzttt
\immediate\write18{php zzz.php}
\input{zzz.ttt}}
```

Teraz wywołanie np.
`\dzialanie3.18*4.35`
spowoduje, że pod poleceniem `\wynik` będzie się krył wynik mnożenia.

Kolejny makro jest jeszcze bardziej interesujące. Otóż możemy pod TeX-em (odwołując się po drodze do skryptu `php`) robić różne obliczenia używając funkcji i algorytmów, które są w `php` dostępne.

Oto przykładowe makro, które wywołuje funkcje:

```
\newwrite\xxxttt
\def\funkcja#1(#2){
\immediate\openout\xxxttt=xxx.php\relax
\immediate\write\xxxttt{\string<?}
\immediate\write\xxxttt{$fp=fopen("zzz.ttt","w");}
\immediate\write\xxxttt{$x=#2;}
\immediate\write\xxxttt{$y=#1;}
\immediate\write\xxxttt{fputs(\string$fp, "\string\oblicz");}
\immediate\write\xxxttt{fputs($fp, $y);}
\immediate\write\xxxttt{fputs($fp, "\string\koniec");}
\immediate\write\xxxttt{fclose($fp);}
\immediate\write\xxxttt{?\string>}
\immediate\closeout\xxxttt
\immediate\write18{php xxx.php}
```

```
\input{zzz.ttt}
}
```

Pod #1 kryje się wzór na obliczenie, a pod #2 argument. Obliczony rezultat jest liczbą kryjącą się pod poleceniem \wynik. Na przykład wykonanie polecenia \funkcja $x*$x*$x(2) spowoduje, że pod zmienną \wynik kryje się napis (nie liczba!) „2”,

A oto przykład ciekawego zastosowania w dydaktyce wykorzystującego gotowy już program napisany w TeX-u do produkcji zadań testowych. Jego działanie opisane jest w [8].

Opiszemy krótko składnię takiego pytania testowego.

```
\question
Treść pytania
\answers
{odpowiedź pierwsza\true x}
...
{odpowiedź ostatnia\true x}
\endquestion
```

gdzie x jest jedynką, gdy odpowiedź jest prawdziwa i 0, gdy nie jest.

Przypuśćmy, że chcemy przygotować pytanie:

Niech M będzie punktem, w którym funkcja $f(x) = e^{ax} - bx$ przyjmuje najmniejszą wartość w przedziale $[0;\infty]$,

gdzie a jest liczbą dziesiętną losowaną z przedziału $[0{,}1; 0{,}54]$, a b jest liczbą całkowitą losowaną z przedziału $[6; 21]$.

Pokażmy rozwiązanie: obliczamy pochodną: mamy $f'(x) = ae^{ax} - b$. Przyrównując ją do zera otrzymujemy: $x = \frac{\ln(b/a)}{a}$)

Teraz możemy zredagować zadanie.

```
\question
\setrannum{\xa}{1}{4}
\setrannum{\xb}{6}{21}
Niech $M$ będzie punktem, w~którym  funkcja
$f(x) = e^{0{,}\the\xa x} - \the\xb x$
przyjmuje najmniejszą wartość w~przedziale
$[0;\infty]$. Wtedy \funkcja{log(\the\xb/$x)/$x}[0.\the\xa]
\answers
{\setrannum{\xc}{5}{20}
```

```
$M>\the\xc$
\ifdim\wynik pt>\the\xc pt\xd=1\else\xd=0\true\xd}
{\setrannum{\xc}{21}{42}
$M<\the\xc$
\ifdim\wynik pt<\the\xc pt\xd=1\else\xd=0\fi\true\xd}
{\setrannum{\xc}{42}{55}
$M<\the\xc$
\ifdim\wynik pt<\the\xc pt\xd=1\else\xd=0\ftrue\xd}
\endquestion}
```

Liczba a jest równa 0,1, 0,2, 0,3 lub 0,4. Liczba b jest liczbą całkowitą z przedziału $[6; 21]$. Stąd M należy do przedziału $\left[\frac{\ln(6/0{,}4)}{0{,}4}; \frac{\ln(21/0{,}1)}{0{,}1}\right] \approx [6{,}7; 53{,}5]$. Dlatego w pierwszej odpowiedzi losuje się parametr całkowity z przedziału $[5; 20]$, w drugiej z przedziału $[21; 42]$, w trzeciej z przedziału $[43; 55]$. Następnie porównuje się wylosowane wielkości kryjące się pod zmienną `\xc` z liczbą M. Ponieważ TeX nie potrafi działać na liczbach niecałkowitych zastosowane jest przypisanie zarówno wielkości M kryjącej się pod zmienną `\wynik` jak i zmiennej `\xc` odpowiednim długościom, a długości niecałkowite TeX potrafi porównywać!

Można też zmieniając nieco makro `\funkcja` w miejsce wzoru na funkcję w pliku `php` wpisywać całe algorytmy, choć przy zadaniach egzaminacyjnych byłoby to niezbyt przydatne - zadania egzaminacyjne nie powinny być skomplikowane rachunkowo!

5. Wykorzystanie więcej niż dwóch języków

W ostatniej części pracy pokażemy problem rozwiązany dzięki zastosowaniu dwóch gotowych wcześniej programów, napisanych w różnych językach oraz algorytmu pliku wsadowego.

Trzy lata temu został napisany przez autora w TeX-u program do tworzenia i obróbki ankiet studenckich oceniających wykładowców. Jest on opisany w [11]. Składa się on z dwóch składników. Pierwszy to program o nazwie `ankieta.tex`, który odczytując plik pod nazwą `grupa.txt` zawierający spis wykładowców prowadzących zajęcia w danej grupie, tworzy tekst ankiet. Studenci zapisują swoje oceny pracowników na odpowiednich kwestionariuszach. Kwestionariusze te są następnie przepuszczane przez skaner i odczyt kwestionariuszy jest zapisywany w pliku `ankieta.csv`. Drugi składnik to program o nazwie `wynik.tex`, który odpowiednio łączy pliki `ankieta.csv` i `grupa.txt`, przeprowadza obliczenia (np.liczy średnie oceny każdego wykładowcy i średnie oceny całej grupy) i drukuje gotowe wyniki.

Początkowo był on stosowany w Wydziale Zarzadzania WSM w Ciechanowie, gdzie grup było co najwyżej sześć i w związku z tym nie było potrzeby większej jeszcze automatyzacji pracy. Tworzono folder do każdej grupy i w każdym z nich przygotowywano plik `grupa.txt`, przeprowadzano najpierw procedurę przygotowania ankiety, a potem jej obróbki i nie zajmowało to dużo czasu.

Rok później wykorzystaliśmy ten program w jednostce macierzystej w Warszawie, gdzie grup było ponad 80. Przygotowanie 80 plików `grupa.txt`, 80 różnych ankiet w różnych katalogach[2], potem obróbka ankiet zajęły wiele godzin.

Dlatego naturalna wydała się idea, żeby zautomatyzować przynajmniej częściowo pracę.

Pliki zawierające dane o wykładowcach danej grupy okazały się być prawie gotowe w odpowiedniej jednostce organizacyjnej uczelni. Były to pliki w formacie `csv` mające w nazwach informacje jakiej grupy dotyczą. Na przykład `1_SEMESTR_ADMINISTRACJI_II_STOPNIA_NIESTACJONARNE_TOK1.csv` itp.
Trzeba było w takim razie znaleźć sposób na automatyczne przetwarzanie ich do struktury identycznej jak struktura pliku `grupa.txt`. i tu na pomoc przyszedł gotowy już program napisany wiele lat temu w pascalu do innych wprawdzie celów, ale praktycznie co do joty realizujący żądane zadanie, oraz możliwości algorytmiczne plików wsadowych. Po drobnej korekcie powstał program napisany w pascalu `zamien.exe` przetwarzający plik pod nazwą `grupa.ccc` mający strukturę taką jak we wspomnianej jednostce organizacyjnej na plik `grupa.txt` mający strukturę i nazwę już gotową to przetwarzania przez program `ankieta.tex`. Teraz wystarczyło umieścić wszystkie 80 plików `csv`, pliki `ankieta.tex`, `wynik.tex` we wspólnym katalogu (nazwaliśmy go `ankieta`[3]). Tworzymy następnie dwa pliki wsadowe.

Oto pierwszy z nich:

```
md ankietapdfy
cd..
md wynikiankiety
cd ankieta
FOR %%A IN (*.csv) DO call wykonaj.bat %%A
```

[2] Już ręczne utworzenie 80 katalogów - ich nazwy nie mogą być zbyt krótkie, muszą zawierać dane o grupie - zajęło trochę czasu

[3] Całą pracę wykonywaliśmy na dysku `D`, oczywiście można to zmienić

Tworzy on dwa katalogi, w jednym będą gotowe do druku ankiety pod odpowiednimi nazwami, w drugim będziemy potem tworzyć podkatalogi, w których będziemy w przyszłości przetwarzać wyniki ankiet. Następnie wywołuje odpowiednią pętlę przetwarzając wszystkie pliki `csv`.

Drugi `wykonaj.bat` jest następujący:

```
copy %1 grupa.ccc
call zamien.exe
call pdflatex ankieta.tex
copy ankieta.pdf d:\ankieta\ankietapdfy\%1.pdf
cd..
cd wynikiankiety
md %1
cd..
cd ankieta
copy grupa.txt d:\wynikiankiety\grupa.txt
copy wynik.tex d:\wynikiankiety\wynik.tex
```

Plik `robankiety.bat` dla każdego pliku `csv` uruchamia plik `wykonaj.bat` z nazwą pliku csv, czyli nazwą odpowiedniej grupy jako parametrem. Przetwarza ten plik na plik `grupa.txt`, kompiluje plik `ankieta.tex` tworząc plik `ankieta.pdf`, będący ankietą dla odpowiedniej grupy. Następnie kopiuje ten plik pod nazwą grupy do podkatalogu `ankietapdfy`. Wreszcie tworzy w katalogu `wynikiankiety` podkatalog mający nazwę grupy i kopiuje do tych podkatalogów pliki potrzebne w przyszłości do przetwarzania wyników ankiet.

W ten sposób praktycznie jednym naciśnięciem klawisza mamy wszystkie 80 ankiet w formacie `pdf` gotowych do druku.

Przetwarzania ankiet nie da się już tak dobrze zautomatyzować, ale można to zrobić choć częściowo.

Formalarze wypełnione przez studentów w każdej grupie przepuszczamy przez skaner i wyniki odczytu zapisujemy pod nazwą `wynik.csv` w odpowiednim podkatalogu związanym z tą grupą, gdzie już „czekają na niego" pliki `grupa.txt` oraz `wynik.tex`. To jest ten fragment pracy, którego zautomatyzować nie można, ale do ostatniej części można przygotować odpowiedni plik wsadowy, który końcową pracę zrobi za nas.

Oto on (nazwijmy go `robwyniki.bat`):

```
md wynikiankietypdfy
FOR %%A IN (*.csv) DO call wykonajwyniki.bat %%A
```

Z kolei plik `wykonajwyniki.bat` jest następujący:

```
cd..
cd wynikiankiety
cd %1
call pdflatex wynik.tex
copy wynik.pdf d:\ankieta\wynikiankietypdfy\%1wyniki.pdf
cd..
cd ankieta
```

Wszystkie wyniki ankiet zapiszą się w formacie `pdf` we wspólnym katalogu.

Bibliografia

[1] Abrahams P., Hargreaves K., Berry K., (1990); *TeX for the Impatient*, Addison-Wesley

[2] Borde A., *TeX w przykładach*, `ftp://ftp.gust.org.pl/pub/GUST/contrib/TBE/tbe.pdf`

[3] Eijkhout V., *TeX by Topics. A TeXnician's Reference*, `http://www.gust.org.pl/doc/documentation`

[4] Kierzkowski A. (2004), PHP 5 Ćwiczenia praktyczne, Helion

[5] Knuth D. E. (2005); *TeX Przewodnik użytkownika*, WNT

[6] Krysicki W., Bartos J., Dyczka W., Królikowski K, Wasilewski W, (2000); *Rachunek prawdopodobieństawa i statystyka matematyczna w zadaniach*, Wydawnictwo Naukowe PWN

[7] Kuratowski K. (1961), Rachunek różniczkowy i całkowy, PWN

[8] Rusinek J., (2007); *Algorytm permutowania w TeX-u zastosowany do informatyzacji procesu egzaminacyjnego*, „Rocznik Naukowy Wydziału Zarządzania w Ciechanowie", 1-4 (I), (153-174)

[9] Rusinek J., (2008); *Pliki do odczytu i zapisu w TeX-u – zastosowanie do przetwarzania wyników egzaminu* , „Rocznik Naukowy Wydziłu Zarządzania w Ciechanowie", 1-2 (II), (107-124)

[10] Rusinek J., (2009); *Zmienne liczbowe w TeX-u. Zastosowanie w dydaktyce*, Rocznik Naukowy Wydziału Zarządzania w Ciechanowie, 1-2 (III), (113-124)

[11] Rusinek J., (2010); *Obliczeniowe możliwości TeX-a. Zastosowanie wspomagające dydaktykę*, „Rocznik Naukowy Wydziału Zarządzania w Ciechanowie", 1-4 (IV), (119-131)

[12] `http://blog.netbiel.pl/uruchamianie-skryptu-php-z-linii-polecen.html`