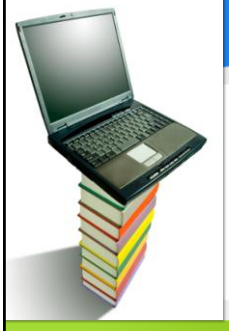


Jerzy Nawrocki
Wydział Informatyki
Politechnika Poznańska
jerzy.nawrocki@put.poznan.pl

# Przetwarzanie tekstów i AWK



## Problem konwersji plików

```
FName:Jurek   SName:Busz   Salary 585
FName:Alek SName:Gor   Salary 700
```

| FName | SName | Salary |
|---|---|---|
| Jurek | Busz | 585 |
| Alek | Gor | 700 |





## Problem konwersji plików

```
#include <stdio.h>
#include <stdlib.h>
FILE *fin;
char token[200];
char gettoken(void)
{int i=0;
char c;
do {c = getc(fin);
    if (c == EOF) return (EOF);
    } while (c < '!');
```

Rozwiązanie w C:

≈ 40 linii kodu





## Problem konwersji plików

Rozwiązanie w AWK

```
BEGIN   {FS=":| ";}
NR == 1  {print $1, "\t", $3, "\t", $5;}
         {gsub(/,/, "." , $6);  print $2, "\t", $4, "\t", $6;}
```





## Powstanie języka

Bell Labs, Murray Hill (New Jersey), Foto: http://en.wikipedia.org/wiki/Bell_Labs

**Bell Labs, New Jersey (USA), 1977**

**AWK: Aho, Weinberger, Kernighan**
**Platformy: Unix, MS DOS/Windows**
**Podobieństwo do C**

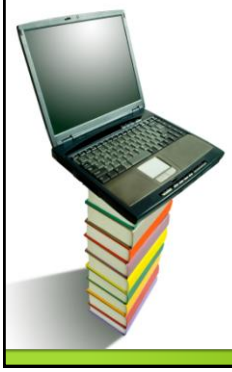


## Cel wykładu

**Przedstawić:**

- **Inny paradygmat programowania (programowanie regułowe)**
- **Wyrażenia regularne**
- **Podstawy programowania w języku AWK**

## Plan wykładu



- **Idea języka AWK**
- Najprostsze programy
- Wzorce wiersza
- Wyrażenia regularne
- Zmienne





## Idea języka AWK – Plik wejściowy

Pole

Wiersz

```
Jerzy Nawrocki    43089  I1
Jan   Kowalski    43780  I2
Adam  Malinowski  43990  I1
```

Pola: **$1**, **$2**, **$3**, ...





## Idea języka AWK – Schemat programu

Reguła przetwarzania

```
wzorzec1   {instrukcje1}
wzorzec2   {instrukcje2}
  ...          ...
```





## Zasada działania

```
Jerzy Nawrocki
Jan   Kowalski
Adam  Malinowski
```

```
wzorzec1 {instrukcje1}
wzorzec2 {instrukcje2}
  ...        ...
```





## Plan wykładu

- Idea języka AWK
- **Najprostsze programy**
- Wzorce wiersza
- Wyrażenia regularne
- Zmienne





## Najprostsze programy

Ile pól na wyjściu?

```
Jerzy Nawrocki    43089  I1
Jan   Kowalski    43780  I2
Adam  Malinowski  43990  I1
```

```
$4=="I1"     { print $2, $1; }
```

## Najprostsze programy


```
Jerzy Nawrocki    43089   I1
Jan   Kowalski    43780   I2
Adam  Malinowski  43990   I1
```

```
$4=="I1"
```





## Najprostsze programy


```
Jerzy Nawrocki    43089   I1
Jan   Kowalski    43780   I2
Adam  Malinowski  43990   I1
```

```
{ print $2, $1; }
```





## Plan wykładu

- Idea języka AWK
- Najprostsze programy
- **Wzorce wiersza**
- Wyrażenia regularne
- Zmienne





## Wzorce wiersza

- Początek i koniec tekstu
- Relacje
- Wzorce złożone
- ~~Wzorce zakresu~~
- *Wyrażenia regularne*





## Początek i koniec tekstu

```
Jerzy Nawrocki    43089   I1
Jan   Kowalski    43780   I2
Adam  Malinowski  43990   I1
```

```
BEGIN      { print "-----"; }
$4=="I2"   { print $2, $1; }
END        { print "*****"; }
```





## Początek i koniec tekstu

```
Jerzy Nawrocki    43089   I1
Jan   Kowalski    43780   I2
Adam  Malinowski  43990   I1
```

```
END        { print "*****"; }
$4=="I2"   { print $2, $1; }
BEGIN      { print "-----"; }
```

## Relacje

```
12      11
 2      11
```

$1 > $2





## Wzorce złożone

|| lub (alternatywa)
$1==1 || $2==1

&& i (koniunkcja)
$1==1 && $2==1

! nie (zaprzeczenie)
! $1==1





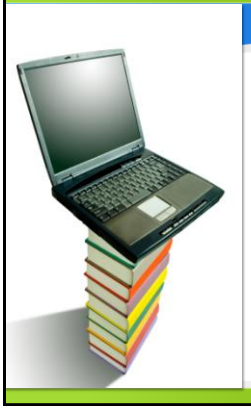
## Wzorce złożone

```
Jerzy Adam          43089   I1
Adam   Kowalski     43780   I2
Adam   Malinowski   43990   I1
```

```
$4=="I1" && $1=="Adam"    { print $2, $1; }
```





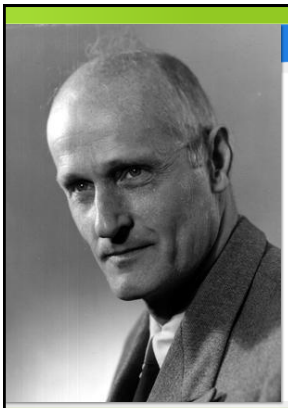


## Plan wykładu

- Idea języka AWK
- Najprostsze programy
- Wzorce wiersza
- **Wyrażenia regularne**
- Zmienne







## Stephen Kleene

1909-01-05, Connecticut, USA

1934: Dr, Princeton Univ., (Alonzo Church)

1935: Univ. of Wisconsin-Madison (USA)

1939-40: Inst. for Advanced Study, Princeton – teoria rekurencji

1990: National Medal of Sci.

1994-01-25, Madison

http://www.math.wisc.edu/~gpslogic/





## Wyrażenia regularne

### Wyrażenie arytmetyczne

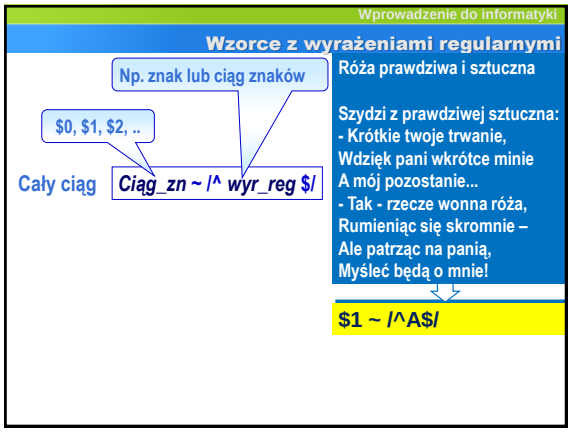
**Wartość: Tekst → Liczba**

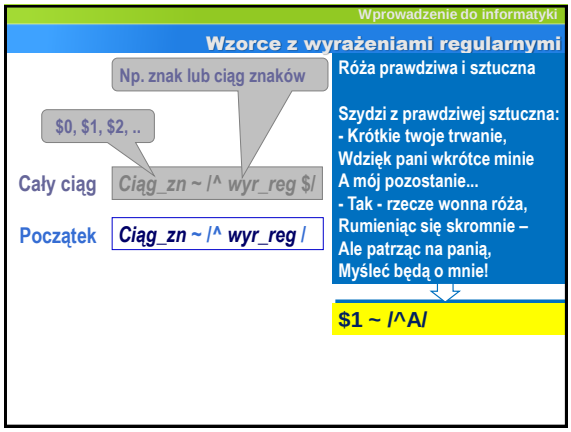
**Wartość(2·3 + 3) = 9**

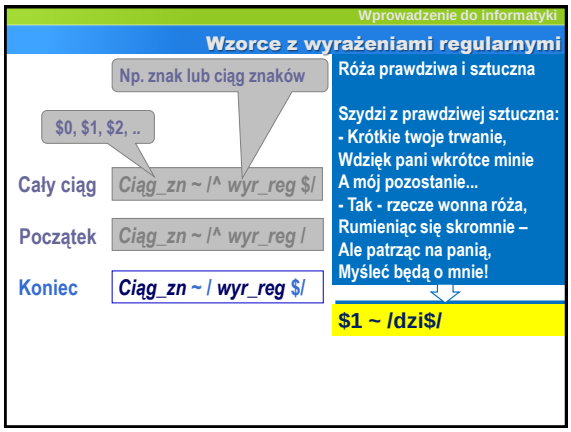
### Wyrażenie regularne

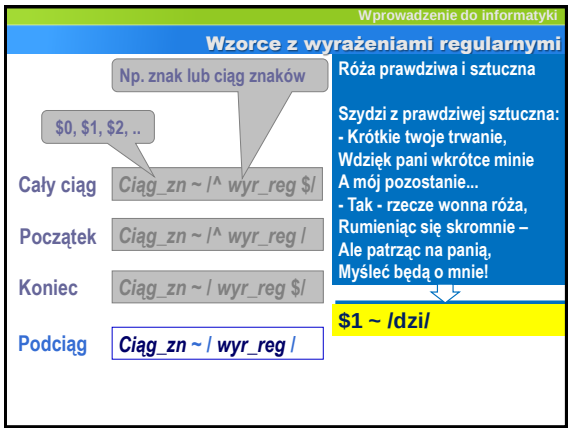
**Wartość: Tekst → ZbiórCiągówZnaków**

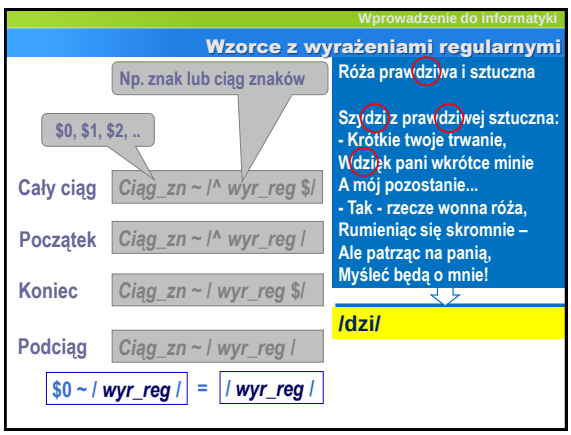
**Wartość(/Ala | Ola/) = {"Ala", "Ola"}**

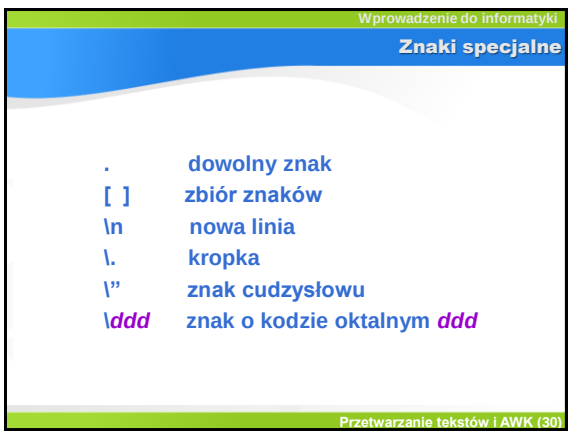

Wprowadzenie do informatyki
Wzorce z wyrażeniami regularnymi
Np. znak lub ciąg znaków
$0, $1, $2, ..
Cały ciąg
Ciąg_zn ~ /^ wyr_reg $/
Róża prawdziwa i sztuczna
Szydzi z prawdziwej sztuczna:
- Krótkie twoje trwanie,
Wdzięk pani wkrótce minie
A mój pozostanie...
- Tak - rzecze wonna róża,
Rumieniąc się skromnie –
Ale patrząc na panią,
Myśleć będą o mnie!
$1 ~ /^A$/


Wprowadzenie do informatyki
Wzorce z wyrażeniami regularnymi
Np. znak lub ciąg znaków
$0, $1, $2, ..
Cały ciąg
Ciąg_zn ~ /^ wyr_reg $/
Początek
Ciąg_zn ~ /^ wyr_reg /
Róża prawdziwa i sztuczna
Szydzi z prawdziwej sztuczna:
- Krótkie twoje trwanie,
Wdzięk pani wkrótce minie
A mój pozostanie...
- Tak - rzecze wonna róża,
Rumieniąc się skromnie –
Ale patrząc na panią,
Myśleć będą o mnie!
$1 ~ /^A/


Wprowadzenie do informatyki
Wzorce z wyrażeniami regularnymi
Np. znak lub ciąg znaków
$0, $1, $2, ..
Cały ciąg
Ciąg_zn ~ /^ wyr_reg $/
Początek
Ciąg_zn ~ /^ wyr_reg /
Koniec
Ciąg_zn ~ / wyr_reg $/
Róża prawdziwa i sztuczna
Szydzi z prawdziwej sztuczna:
- Krótkie twoje trwanie,
Wdzięk pani wkrótce minie
A mój pozostanie...
- Tak - rzecze wonna róża,
Rumieniąc się skromnie –
Ale patrząc na panią,
Myśleć będą o mnie!
$1 ~ /dzi$/


Wprowadzenie do informatyki
Wzorce z wyrażeniami regularnymi
Np. znak lub ciąg znaków
$0, $1, $2, ..
Cały ciąg
Ciąg_zn ~ /^ wyr_reg $/
Początek
Ciąg_zn ~ /^ wyr_reg /
Koniec
Ciąg_zn ~ / wyr_reg $/
Podciąg
Ciąg_zn ~ / wyr_reg /
Róża prawdziwa i sztuczna
Szydzi z prawdziwej sztuczna:
- Krótkie twoje trwanie,
Wdzięk pani wkrótce minie
A mój pozostanie...
- Tak - rzecze wonna róża,
Rumieniąc się skromnie –
Ale patrząc na panią,
Myśleć będą o mnie!
$1 ~ /dzi/


Wprowadzenie do informatyki
Wzorce z wyrażeniami regularnymi
Np. znak lub ciąg znaków
$0, $1, $2, ..
Cały ciąg
Ciąg_zn ~ /^ wyr_reg $/
Początek
Ciąg_zn ~ /^ wyr_reg /
Koniec
Ciąg_zn ~ / wyr_reg $/
Podciąg
Ciąg_zn ~ / wyr_reg /
$0 ~ / wyr_reg / = / wyr_reg /
Róża prawdziwa i sztuczna
Szydzi z prawdziwej sztuczna:
- Krótkie twoje trwanie,
Wdzięk pani wkrótce minie
A mój pozostanie...
- Tak - rzecze wonna róża,
Rumieniąc się skromnie –
Ale patrząc na panią,
Myśleć będą o mnie!
/dzi/


Wprowadzenie do informatyki
Znaki specjalne
. dowolny znak
[ ] zbiór znaków
\n nowa linia
\. kropka
\" znak cudzysłowu
\ddd znak o kodzie oktalnym ddd
Przetwarzanie tekstów i AWK (30)

## Znaki specjalne


```
/^.$/
/[0123456789]/
/[0-9]/
```





## Dopełnienie zbioru znaków


[^ ... ]

```
/[^0-9]/
/^[0-9]/
```





## Alternatywa (lub)

*wyr_reg* | *wyr_reg*


```
Ząbki listka

Nic tak nie potrafi gryźć
Jak cudzy laurowy liść.
```

```
/fi | ki/
```





## Nawiasy i alternatywa

Nawiasy podnoszą priorytet operatorów:
3*(4 + 5) = 3* 9 = 27

Rozdzielność mnożenia względem dodawania:
3*(4 + 5) = 3*4 + 3*5 = 12 + 15 = 27
(4 + 5)*3 = 4*3 + 5*3 = 12 +15 = 27

Rozdzielność konkatenacji względem alternatywy

```
/(f | k)i/      =  /fi | ki/
/Adam(e | i)k/  =  /Adamek | Adamik/
```





## Operator powtarzania

Wypisz wszystkie wiersze, w których pierwsze pole jest liczbą.

```
1 2 3
Rok 1984
4 damy
```

```
$1 ~ /[0-9]/
```





## Operator powtarzania

Wypisz wszystkie wiersze, w których pierwsze pole jest liczbą.

```
1 2 3
Rok 1984
4 damy
```

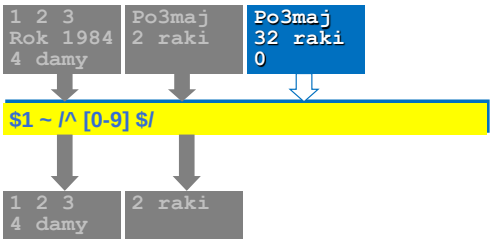
```
Po3maj
2 raki
```

```
$1 ~ /[0-9]/
```

```
1 2 3
4 damy
```

## Operator powtarzania

Wypisz wszystkie wiersze, w których pierwsze pole jest liczbą.

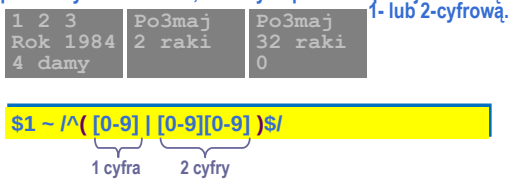







## Operator powtarzania

Wypisz wszystkie wiersze, w których pierwsze pole jest liczbą 1- lub 2-cyfrową.

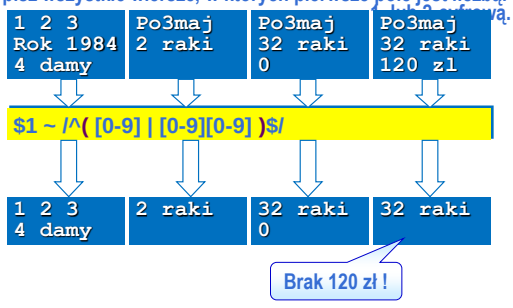







## Operator powtarzania

Wypisz wszystkie wiersze, w których pierwsze pole jest liczbą.

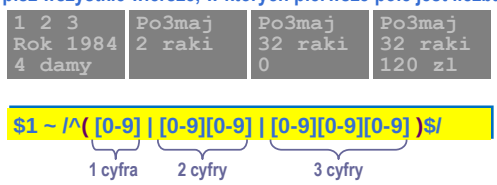







## Operator powtarzania

Wypisz wszystkie wiersze, w których pierwsze pole jest liczbą.








## Operator powtarzania

$[0\text{-}9] = [0\text{-}9]^1$

$[0\text{-}9][0\text{-}9] = [0\text{-}9]^2$

$[0\text{-}9][0\text{-}9][0\text{-}9] = [0\text{-}9]^3$

. . . . . .

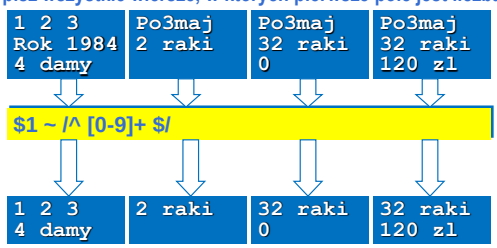
$[0\text{-}9]^1 \mid [0\text{-}9]^2 \mid [0\text{-}9]^3 \mid \ldots = [0\text{-}9]+$





## Operator powtarzania

Wypisz wszystkie wiersze, w których pierwsze pole jest liczbą.



## Operator powtarzania *

Może wystąpić sekwencja w.

Musi przynajmniej raz wystąpić w.

w+ = w | ww | www | . . .

w* = ε | w | ww | www | . . .

Ciąg pusty

w(ε | w | ww | www |.. )= w|ww|www|wwww|..

x ε = x

ε x = x

w+ = w w* = w* w

x w* = x | x w+

w*x = x | w+x





## Zagadka

Co robi ten program?

```
Nadgodziny  2001/02
================
Nawrocki 60
Complak 359
```

```
$2 ~ /^[0-9]+$/
```





## Plan wykładu

- Idea języka AWK
- Najprostsze programy
- Wzorce wiersza
- Wyrażenia regularne
- **Zmienne**





## Zmienne

- Zmienne wprowadzone przez programistę
  (*typ*: ciąg znaków;
  *wartość początkowa*: ciąg pusty / zero)
- Zmienne wbudowane
  (mają standardowe znaczenie)
- Zmienne polowe $1, $(i+j-1), ..





## Przykładowe zmienne wbudowane

NF - liczba pól w wierszu
NR - numer wiersza
FILENAME - nazwa pliku



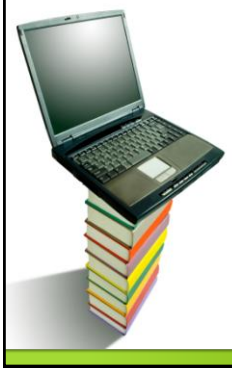


## Zmienne

| NR | NF | total |
|---|---|---|
| 1 | 2 | 0 |
| | | 2 |

```
Reguly sitwy
------------
Zero do zera
A bedzie kariera.
```

```
{total= total + NF;}
END {print "Pol: ", total;
     print "Wierszy: ", NR;}
```



**Podsumowanie**





## Cel wykładu

**Przedstawić:**

- **Inny paradygmat programowania (programowanie regułowe)**





## Idea języka AWK – Plik wejściowy

Pole

Wiersz

```
Jerzy Nawrocki   43089  I1
Jan   Kowalski   43780  I2
Adam  Malinowski 43990  I1
```

Pola: **$1**, **$2**, **$3**, ...





## Idea języka AWK – Schemat programu

Reguła przetwarzania

```
wzorzec1   {instrukcje1}
wzorzec2   {instrukcje2}
  ...        ...
```





## Zasada działania

```
Jerzy Nawrocki
Jan   Kowalski
Adam  Malinowski
```

```
wzorzec1 {instrukcje1}
wzorzec2 {instrukcje2}
  ...        ...
```





## Cel wykładu

**Przedstawić:**

- **Inny paradygmat programowania (programowanie regułowe)**
- **Wyrażenia regularne**

## Wyrażenia regularne

**Wyrażenie arytmetyczne**

**Wartość: Tekst → Liczba**

**Wartość(2·3 + 3) = 9**

**Wyrażenie regularne**

**Wartość: Tekst → ZbiórCiągówZnaków**

**Wartość(/Ala | Ola/) = {"Ala", "Ola"}**





## Wyrażenia regularne

$\varepsilon\, x = x\, \varepsilon$

$L_1 \bullet L_2 = \{xy: x \in L_1 \wedge y \in L_2\}$

$L^0(r) = \{\varepsilon\}$

$L^{n+1}(r) = L(r) \bullet L^{n+1}(r)$

$L^n(r) = \{\underbrace{xx\ldots x}_{n}: \ x \in L(r)\}$

$L(r^*) = \bigcup L^n(r)$

$L(r^*) = \{\varepsilon, x, xx, \ldots x^n: \ x \in L(r)\}$ $(01)^* = \{\varepsilon, 01, 0101, 010101, \ldots\}$





## Cel wykładu

**Przedstawić:**

- **Inny paradygmat programowania (programowanie regułowe)**
- **Wyrażenia regularne**
- **Podstawy programowania w języku AWK**





## Podsumowanie


**gawk -f prog.awk <in.txt >out.txt**

**Inne mechanizmy AWK:**

- **Instrukcje złożone (if, while, ..)**
- **Tablice dynamiczne**
- **Funkcje wbudowane (gsub, ..)**





## Literatura


- A. Aho, B. Kernighan, P. Weinberger, *The AWK Programming Language*, Addison-Wesley, Reading, 1988.
- J. Nawrocki, W. Complak, Wprowadzenie do przetwarzania tekstów w języku AWK, *Pro Dialog* 2 (1994), 23-46.