

Ćwiczenie 4

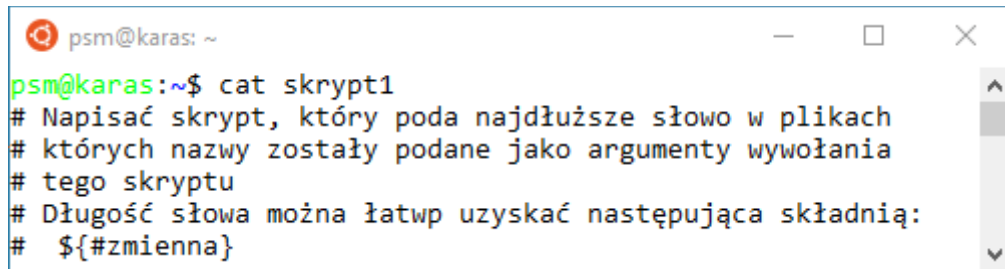
Skrypty dla bash-a

Cel ćwiczenia:

Przedmiotem ćwiczenia jest przedstawienie możliwości napisania skryptów w środowisku bash-a.

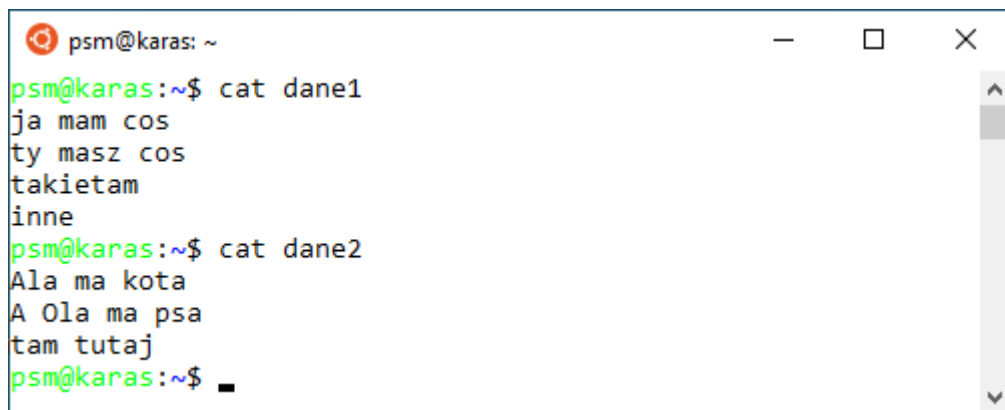
Przebieg ćwiczenia - zagadnienia:

1. Napisz skrypt jak poniżej:
 - a. treść



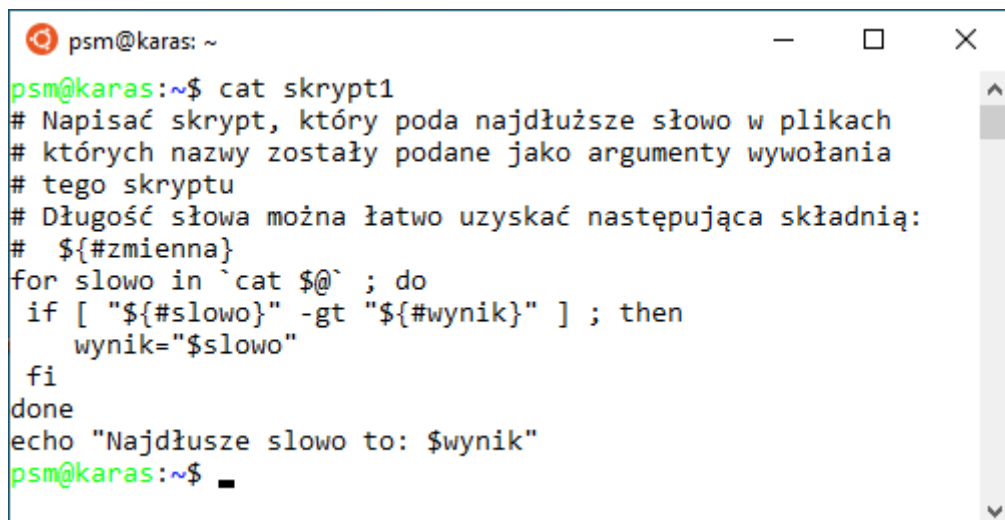
```
psm@karas: ~  
psm@karas:~$ cat skrypt1  
# Napisać skrypt, który poda najdłuższe słowo w plikach  
# których nazwy zostały podane jako argumenty wywołania  
# tego skryptu  
# Długość słowa można łatwo uzyskać następującą składnią:  
# ${#zmienna}
```

- b. dane - pliki wejściowe i zawartość wybranego pliku:



```
psm@karas: ~  
psm@karas:~$ cat dane1  
ja mam cos  
ty masz cos  
takietam  
inne  
psm@karas:~$ cat dane2  
Ala ma kota  
A Ola ma psa  
tam tutaj  
psm@karas:~$
```

- c. treść skryptu:



```
psm@karas: ~  
psm@karas:~$ cat skrypt1  
# Napisać skrypt, który poda najdłuższe słowo w plikach  
# których nazwy zostały podane jako argumenty wywołania  
# tego skryptu  
# Długość słowa można łatwo uzyskać następującą składnią:  
# ${#zmienna}  
for slowo in `cat $@` ; do  
    if [ "${#slowo}" -gt "${#wynik}" ] ; then  
        wynik="$slowo"  
    fi  
done  
echo "Najdłuższe słowo to: $wynik"  
psm@karas:~$
```

- d. wykonanie

```
psm@karas: ~  
psm@karas:~$ ./skrypt1 dane1 dane2  
Najdłuższe słowo to: takietam  
psm@karas:~$
```

2. Napisz skrypt jak poniżej
 - a. treść

```
psm@karas: ~  
psm@karas:~$ cat skrypt2  
# Napisz skrypt, który wypisuje swoje argumenty  
# w odwrotnej kolejności  
#
```

- b. treść skryptu:

```
psm@karas: ~  
psm@karas:~$ cat skrypt2  
# Napisz skrypt, który wypisuje swoje argumenty  
# w odwrotnej kolejności  
#  
# for zmienna ; do jest równoważne for zmienna in $@ ; do  
#  
for k in $@ ; do  
    WYNIK="$k $WYNIK"  
done  
echo $WYNIK  
psm@karas:~$
```

- c. wynik działania

```
psm@karas: ~  
psm@karas:~$ ./skrypt2 a b c d e f g h  
h g f e d c b a  
psm@karas:~$
```

- d. identycznie działa skrypt jak poniżej:

```
psm@karas: ~  
psm@karas:~$ cat skrypt2  
# Napisz skrypt, który wypisuje swoje argumenty  
# w odwrotnej kolejności  
#  
for k ; do  
    WYNIK="$k $WYNIK"  
done  
echo $WYNIK  
psm@karas:~$
```

3. Przykład skryptu z odwołaniem do tabeli:

a. treść

```
psm@karas: ~  
psm@karas:~$ cat skrypt3  
# Definicja tabeli  
#  
pora_dnia=(swit poranek południe popołudnie wieczór noc)  
for ((z=1 ; $z <= 3 ; z++ )) ; do  
    echo ${pora_dnia[$z]}  
done  
psm@karas:~$
```

b. wykonanie tego skryptu (indeks tabeli od zera):

```
psm@karas: ~  
psm@karas:~$ ./skrypt3  
poranek  
południe  
popołudnie  
psm@karas:~$
```

c. treść skryptu

```
psm@karas: ~  
psm@karas:~$ cat skrypt3.1  
# Definicja tabeli  
#  
pora_dnia=(swit poranek południe popołudnie wieczór noc)  
for ((y=6 ; $y >= 2 ; y-- )) ; do  
    echo ${pora_dnia[$y]}  
done  
psm@karas:~$
```

d. wynik działania – indeks 6 pokazuje wartość pustą:

```
psm@karas: ~  
psm@karas:~$ ./skrypt3.1  
noc  
wieczór  
popołudnie  
południe  
psm@karas:~$
```

e. treść skryptu

```
psm@karas: ~  
psm@karas:~$ cat skrypt3.2  
# Definicja tabeli  
#  
pora_dnia=(swit poranek południe popołudnie wieczór noc)  
for (( x=0 ; $x <= 3 ; x++ )) ; do  
    echo ${pora_dnia[$x]}  
done  
psm@karas:~$
```

f. wynik działania:

```
psm@karas: ~  
psm@karas:~$ ./skrypt3.2  
swit  
poranek  
południe  
popołudnie  
psm@karas:~$
```

4. Napisać skrypt o treści jak poniżej:

a. treść zadania:

```
psm@karas: ~  
psm@karas:~$ cat skrypt4  
# Napisz skrypt, który na podstawie skrótu lub numeru miesiąca  
# podaje jego pełną nazwę angielską i polską  
#
```

b. treść skryptu:

```
psm@karas: ~  
psm@karas:~$ cat skrypt4  
# Napisz skrypt, który na podstawie skrótu lub numeru miesiąca  
# podaje jego pełną nazwę angielską i polską  
#  
echo -n "Podaj numer lub skrót nazwy miesiąca: "  
read MIESIAC  
case $MIESIAC in  
    1|01|Jan*|st*) echo "$MIESIAC to January (styczeń)." ;;  
    2|02|Feb*|lu*) echo "$MIESIAC to February (luty)." ;;  
    3|03|Mar*|mar*) echo "$MIESIAC to March (marzec)." ;;  
    4|04|Apr*|kw*) echo "$MIESIAC to April (kwiecień)." ;;  
    *) echo "$MIESIAC to nieznany miesiac." ;;  
esac  
psm@karas:~$
```

c. wynik działania:

```
psm@karas: ~  
psm@karas:~$ ./skrypt4  
Podaj numer lub skrót nazwy miesiąca: 4  
4 to April (kwiecień).  
psm@karas:~$ ./skrypt4  
Podaj numer lub skrót nazwy miesiąca: 02  
02 to February (luty).  
psm@karas:~$ ./skrypt4  
Podaj numer lub skrót nazwy miesiąca: Feb  
Feb to February (luty).  
psm@karas:~$ ./skrypt4  
Podaj numer lub skrót nazwy miesiąca: kw  
kw to April (kwiecień).  
psm@karas:~$ ./skrypt4  
Podaj numer lub skrót nazwy miesiąca: 08  
08 to nieznany miesiac.  
psm@karas:~$
```

5. Napisać skrypt o treści jak poniżej:
a. treść zadania:

```
psm@karas: ~  
psm@karas:~$ cat skrypt5  
# Napisać skrypt, który wyrysowuje z gwiazdek kwadrat  
# o boku zadany argumentem wywołania tego skryptu.  
# Kwadrat ma być wypełniony gwiazdkami. Skrypt zawiera  
# sprawdzenie poprawności argumentu wywołanego skryptu:  
# argument musi być większy od zera i mniejszy od 40  
#
```

- b. treść skryptu:

```
Wybierzpsm@karas: ~  
psm@karas:~$ cat skrypt5  
# Napisać skrypt, który wyrysowuje z gwiazdek kwadrat  
# o boku zadany argumentem wywołania tego skryptu.  
# Kwadrat ma być wypełniony gwiazdkami. Skrypt zawiera  
# sprawdzenie poprawności argumentu wywołanego skryptu:  
# argument musi być większy od zera i mniejszy od 40  
#  
if [ $# -ne 1 ] ; then  
    echo "Skrypt musi być wywołany z argumentem w postaci wielkości boku kwadratu"  
.  
else  
    if [ $1 -lt 1 -o $1 -gt 39 ] ; then  
        echo "Argument musi być większy niż zero lub mniejszy niż 40"  
    else  
        for (( y=1 ; $y <= $1 ; y++ )) ; do  
            for (( x=1 ; $x <= $1 ; x++ )) ; do  
                echo -n "*"   
            done  
            echo ""  
        done  
    fi  
fi  
psm@karas:~$
```

c. wynik działania:

```
psm@karas: ~  
psm@karas:~$ ./skrypt5 -1  
Argument musi byc więszy niż zero lub mniejszy oniż 40  
psm@karas:~$ ./skrypt5 8  
*****  
*****  
*****  
*****  
*****  
*****  
*****  
*****  
psm@karas:~$ █
```

6. Napisać skrypt o treści jak poniżej:

a. treść zadania:

```
psm@karas: ~  
psm@karas:~$ cat skrypt6  
# Napisać skrypt, który wyrysowuje z gwiazdek kwadrat  
# o boku zadany argumentem wywołania tego skryptu.  
# Kwadrat ma być PUSTY - ma mieć obramowanie z gwiazdek.  
# Skrypt zawiera sprawdzenie poprawności argumentu wywołanego skryptu:  
# argument musi byc większy od dwóch i mniejszy od 40  
#
```

b. treść skryptu:

```
psm@karas: ~  
psm@karas:~$ cat skrypt6  
# Napisać skrypt, który wyrysowuje z gwiazdek kwadrat  
# o boku danym argumentem wywołania tego skryptu.  
# Kwadrat ma być PUSTY - ma mieć obramowanie z gwiazdek.  
# Skrypt zawiera sprawdzenie poprawności argumentu wywołanego skryptu:  
# argument musi być większy od dwóch i mniejszy od 40  
#  
if [ $# -ne 1 ] ; then  
    echo "Skrypt musi być wywołany z argumentem w postaci wielkości boku kwadratu"  
.  
else  
    if [ $1 -lt 3 -o $1 -gt 39 ] ; then  
        echo "Argument musi być większy niż dwa lub mniejszy niż 40"  
    else  
        for (( y=1 ; $y <= $1 ; y++ )) ; do  
            if [ $y -eq 1 -o $y -eq $1 ] ; then  
                for (( x=1 ; $x <= $1 ; x++ )) ; do  
                    echo -n "*"   
                done  
                echo ""  
            else  
                echo -n " "   
                for (( z=2 ; $z < $1 ; z++ )) ; do  
                    echo -n " "   
                done  
                echo ""  
            fi  
        done  
    fi  
fi  
psm@karas:~$
```

c. wynik działania:

```
psm@karas: ~  
psm@karas:~$ ./skrypt6 -1  
Argument musi być większy niż dwa lub mniejszy niż 40  
psm@karas:~$ ./skrypt6 8  
*****  
*      *  
*      *  
*      *  
*      *  
*      *  
*      *  
*****  
psm@karas:~$
```

7. Napisz skrypt jak poniżej:

a. treść

```
psm@karas: ~  
psm@karas:~$ cat skrypt1  
# Napisz skrypt, który wylicza sumę liczb całkowitych wpisywanych z klawiatury.  
# Zakładamy, że wszystkie liczby są poprawne
```

b. treść skryptu:

```
psm@karas: ~  
psm@karas:~$ cat skrypt1  
# Napisz skrypt, który wylicza sumę liczb całkowitych wpisywanych z klawiatury.  
# Zakładamy, że wszystkie liczby są poprawne  
echo "Ten skrypt sumuje podane poniżej liczby całkowite: "  
echo -n "Podaj sekwencję liczb: "  
read CIAG  
set $CIAG  
echo "Zadania lista liczb całkowitych: $@"  
echo "Ilość zadanych liczb: $#"  
SUMA=0  
ILOSC=$#  
for (( licz=1; $licz <= $ILOSC; licz++ )); do  
    echo "Krok: $licz"  
    echo "Argument: $1"  
    SUMA=`expr $SUMA + $1`  
    echo "Suma: $SUMA"  
    shift  
    echo "Pozostałe argumenty: $@"  
done  
echo "SUMA KOŃCOWA: $SUMA"  
psm@karas:~$
```

c. wykonanie

```
psm@karas: ~  
psm@karas:~$ ./skrypt1  
Ten skrypt sumuje podane poniżej liczby całkowite:  
Podaj sekwencję liczb: 2 8 4 6  
Zadania lista liczb całkowitych: 2 8 4 6  
Ilość zadanych liczb: 4  
Krok: 1  
Argument: 2  
Suma: 2  
Pozostałe argumenty: 8 4 6  
Krok: 2  
Argument: 8  
Suma: 10  
Pozostałe argumenty: 4 6  
Krok: 3  
Argument: 4  
Suma: 14  
Pozostałe argumenty: 6  
Krok: 4  
Argument: 6  
Suma: 20  
Pozostałe argumenty:  
SUMA KOŃCOWA: 20  
psm@karas:~$
```

8. Napisz skrypt jak poniżej

a. treść

```
psm@karas: ~  
psm@karas:~$ cat skrypt2  
# Napisz skrypt, który odczytuje linia po linii zawartość zadanego pliku tekstowego.  
# Wczytywanie powinno wykorzystywać konstrukcję pipelinu czyli znak |
```

b. treść skryptu:


```
psm@karas: ~  
psm@karas:~$ cat skrypt2  
# Napisz skrypt, który odczytuje linia po linii zawartość zadanego pliku tekstowego.  
# Wczytywanie powinno wykorzystywać konstrukcję pipeline czyli znak |  
PLIK="ala.txt"  
cat "$PLIK" | while read LINIA ; do  
    echo $LINIA  
done  
psm@karas:~$
```

c. plik ala.txt z danymi:

```
Wybierzpsm@karas: ~  
psm@karas:~$ cat ala.txt  
Ala ma kota.  
A Ola ma psa.  
Tak to bywa.  
psm@karas:~$
```

d. wynik działania

```
psm@karas: ~  
psm@karas:~$ ./skrypt2  
Ala ma kota.  
A Ola ma psa.  
Tak to bywa.  
psm@karas:~$
```

9. Przykład skryptu rysującego tabelkę:

a. treść zadania:

```
psm@karas: ~  
psm@karas:~$ cat skrypt3  
# Napisz skrypt, który generuje automatycznie tabelkę w postaci jak poniżej.  
# Przykładowa tabelka o wymiarach 2 na 5 (wiersze i kolumny) wygląda następująco:  
# +---+---+  
# | | | | |  
# +---+---+  
# | | | | |  
# +---+---+  
# Parametry tej tabelki podajemy na zachętę ze strony uruchomionego skryptu.  
# Należy sprawdzić poprawność tych parametrów, tzn. czy są większe od zera.
```

b. treść skryptu

```
psm@karas: ~  
psm@karas:~$ cat skrypt3  
# Napisz skrypt, który generuje automatycznie tabelkę w postaci jak poniżej.  
# Przykładowa tabelka o wymiarach 2 na 5 (wiersze i kolumny) wygląda następująco:  
# +-+---+---+  
# | | | | |  
# +-+---+---+  
# | | | | |  
# +-+---+---+  
# Parametry tej tabelki podajemy na zachęte ze strony uruchomionego skryptu.  
# Należy sprawdzić poprawność tych parametrów, tzn. czy są większe od zera.  
#  
echo -n "Podaj wysokość tabeli (ilość wiersz): "  
read WYSOKOSC  
echo -n "Podaj szerokość tabeli (ilość kolumn): "  
read SZEROKOSC  
if [ $WYSOKOSC -le 0 -o $SZEROKOSC -le 0 ] ; then  
    echo "Wymiary tabelki powinny być większe od zera."  
else  
    for (( y=1 ; $y <= $WYSOKOSC ; y++ )) ; do  
        for (( x=1 ; $x <= $SZEROKOSC ; x++ )) ; do  
            echo -n "+-"  
        done  
        echo "+"  
        for (( x=1 ; $x <= $SZEROKOSC ; x++ )) ; do  
            echo -n "| "  
        done  
        echo "| "  
    done  
    for (( x=1 ; $x <= $SZEROKOSC ; x++ )) ; do  
        echo -n "+-"  
    done  
    echo "+"  
fi  
psm@karas:~$
```

c. wynik działania:

```
psm@karas: ~  
psm@karas:~$ ./skrypt3  
Podaj wysokość tabeli: 2  
Podaj szerokość tabeli: 5  
+-+---+---+  
| | | | |  
+-+---+---+  
| | | | |  
+-+---+---+  
psm@karas:~$ ./skrypt3  
Podaj wysokość tabeli: 5  
Podaj szerokość tabeli: 2  
+-+  
| |  
+-+  
| |  
+-+  
| |  
+-+  
| |  
+-+  
| |  
+-+  
| |  
+-+  
psm@karas:~$
```

10. Napisać skrypt o treści jak poniżej:

d. treść zadania:

```
psm@karas: ~  
psm@karas:~$ cat skrypt4  
# Napisz skrypt, który sprawdza wszystkie pliki z rozszerzeniem .log  
# znajdujące się w bieżącym katalogu, pod kątem występowania w tych plikach słowa fail  
# Efektem działania tego skryptu jest plik z wygenerowanym raportem.  
# Raport ten zawiera wynik powyższego sprawdzenia: liczbę wszystkich plików z rozszerzeniem  
# .log , oraz liczbę plików zawierających w swej treści przynajmniej jedno słowo fail  
# Nazwa pliku raportu jest podawana jako argument wywołania tego skryptu.  
# Należy wykrywać i sygnalizować sytuacje, w których nie podano żadnego argumentu  
# albo zbyt dużo tych argumentów.  
# Przykładowy raport:  
# Bieżący katalog zawiera 20 plików z rozszerzeniem .log  
# w tym: 12 plików zawiera przynajmniej jedno słowo fail  
# W skrypcie NIE MOŻNA wykorzystywać następujących poleceń:  
# grep egrep find awk vi vim i podobnych  
#
```

e. treść skryptu:

```
psm@karas: ~  
psm@karas:~$ cat skrypt4  
# Napisz skrypt, który sprawdza wszystkie pliki z rozszerzeniem .log  
# znajdujące się w bieżącym katalogu, pod kątem występowania w tych plikach słowa fail  
# Efektem działania tego skryptu jest plik z wygenerowanym raportem.  
# Raport ten zawiera wynik powyższego sprawdzenia: liczbę wszystkich plików z rozszerzeniem  
# .log , oraz liczbę plików zawierających w swej treści przynajmniej jedno słowo fail  
# Nazwa pliku raportu jest podawana jako argument wywołania tego skryptu.  
# Należy wykrywać i sygnalizować sytuacje, w których nie podano żadnego argumentu  
# albo zbyt dużo tych argumentów.  
# Przykładowy raport:  
# Bieżący katalog zawiera 20 plików z rozszerzeniem .log  
# w tym: 12 plików zawiera przynajmniej jedno słowo fail  
# W skrypcie NIE MOŻNA wykorzystywać następujących poleceń:  
# grep egrep find awk vi vim i podobnych  
#  
if [ $# -ne 1 ] ; then  
    echo "Skrypt powinien być wywoływany z jednym argumentem - nazwą pliku raportu."  
else  
    RAPORT=$1  
    LPLOG=0  
    LPFAIL=0  
    for PLIK in *.log ; do  
        if [ -f $PLIK ] ; then  
            let LPLOG++  
            LWFAIL=0  
            while read LINIA ; do  
                for SLOWO in $LINIA ; do  
                    if [ $SLOWO == "fail" ] ; then  
                        let LWFAIL++  
                    fi  
                done  
            done < "$PLIK"  
            if [ $LWFAIL -gt 0 ] ; then  
                let LPFAIL++  
            fi  
        fi  
    done  
    echo "Bieżący katalog zawiera $LPLOG plików z rozszerzeniem .log" >> $RAPORT  
    echo " w tym: $LPFAIL plików zawiera przynajmniej jedno słowo fail " >> $RAPORT  
fi  
psm@karas:~$
```

f. zawartość bieżącego katalogu:

```
psm@karas: ~  
psm@karas:~$ ls -l  
total 9  
drwxr-xr-x 1 psm psm 512 Apr 23 15:05 ALFA.log  
-rw-r--r-- 1 psm psm 58 Apr 23 15:04 a.log  
-rw-r--r-- 1 psm psm 43 Apr 23 14:20 ala.txt  
-rw-r--r-- 1 psm psm 36 Apr 23 15:04 b.log  
-rw-r--r-- 1 psm psm 50 Apr 23 15:04 c.log  
-rwxr-xr-x 1 psm psm 531 Apr 23 14:15 skrypt1  
-rwxr-xr-x 1 psm psm 230 Apr 23 14:26 skrypt2  
-rwxr-xr-x 1 psm psm 967 Apr 23 14:53 skrypt3  
-rwxr-xr-x 1 psm psm 1533 Apr 23 16:28 skrypt4  
-rw-r--r-- 1 psm psm 380 Apr 23 15:52 skrypt5  
psm@karas:~$
```

g. zawartość poszczególnych plików z rozszerzeniem .log

```
psm@karas: ~  
psm@karas:~$ cat a.log  
Ala ma kota fail  
A Ola ma psa fail inne słowo  
tera test  
psm@karas:~$ cat b.log  
To jest test  
Inna zawartość tutaj  
psm@karas:~$ cat c.log  
to jest fail inny  
Inne zdanie  
Jeszcze inne zdanie  
psm@karas:~$
```

h. wynik działania:

```
psm@karas: ~  
psm@karas:~$ ./skrypt4 wynik  
psm@karas:~$ cat wynik  
Bieżący katalog zawiera 3 plików z rozszerzeniem .log  
w tym: 2 plików zawiera przynajmniej jedno słowo fail  
psm@karas:~$ █
```

11. Napisać skrypt o treści jak poniżej:

a. treść zadania:

```
psm@karas: ~  
psm@karas:~$ cat skrypt5  
# Napisz skrypt, który drukuje choinkę z gwiazdek.  
# Przykładowa minimalna choinka o podstawie 5 wygląda tak:  
#  
#   *  
#  ***  
# *****  
#   *  
# Jako argument wywołania tego skryptu jest podana dolna krawędź choinki  
# Należy sprawdzić poprawność tego argumentu - musi być nieparzysty  
# i większy od 4.  
# Oto przykładowa choinka o podstawie 7  
#  
#   *  
#  ***  
# *****  
# *******  
#   *  
#  
#
```

b. treść skryptu:

```
#  
if [ $# -ne 1 ] ; then  
    echo "Błędna ilość argumentów wywołania tego skryptu."  
else  
    if [ $1 -lt 5 ] ; then  
        echo "Argument musi być większy od 4."  
    else  
        let WYNIK=$1%2  
        if [ $WYNIK -eq 0 ] ; then  
            echo "Argument musi być nieparzysty!"  
        else  
            let PW=$1/2  
            let PWPLUS=PW+1  
            for (( y=$PWPLUS ; $y >= 1 ; y-- )) ; do  
                for (( LSP=$y ; $LSP >= 1 ; LSP-- )) ; do  
                    echo -n " "  
                done  
                for (( LGW=1 ; LGW <= ($PWPLUS-$y)*2+1 ; LGW++ )) ; do  
                    echo -n "*"   
                done  
                echo ""  
            done  
            for (( LSP=$PWPLUS ; LSP >= 1 ; LSP-- )) ; do  
                echo -n " "  
            done  
            echo ""  
        fi  
    fi  
fi
```

c. wynik działania:

```
psm@karas: ~  
psm@karas:~$ ./skrypt5  
Błędna ilość argumentów wywołania tego skryptu.  
psm@karas:~$ ./skrypt5 6  
Argument musi być nieparzysty!  
psm@karas:~$ ./skrypt5 5  
*  
***  
*****  
*  
psm@karas:~$ ./skrypt5 7  
*  
***  
*****  
*****  
*  
psm@karas:~$ ./skrypt5 11  
*  
***  
*****  
*****  
*****  
*****  
*  
psm@karas:~$ █
```