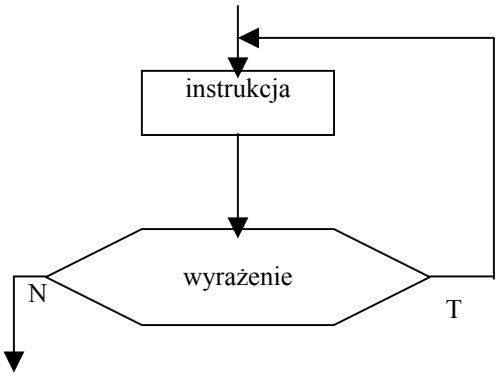
Temat zajęć: **Operatory i instrukcje w języku C - 3**

Autor: mgr inż. Sławomir Samolej

Zagadnienie 1. (instrukcja cyklu: do while)

Język C oferuje kilka instrukcji cyklu (pętli). Oprócz wprowadzonej wcześniej instrukcji „while", stosować można instrukcje „do while" i „for". Schemat blokowy i składnia instrukcji „do while" ma postać:




```
do
instrukcja
while(wyrażenie);
```

Instrukcja będzie wykonywana tak długo, dopóki spełniony będzie warunek zawarty w wyrażeniu „while". Warto zauważyć, że najpierw wykonywana jest instrukcja, a następnie sprawdzana wartość wyrażenia.

Przykłady instrukcji „do while"

1.
```
#include <stdio.h>
void main(void)
{
        char a=10;
        do
                a=a-1;
        while(a>0);
        printf("a=%d\t",a);
}
```

2.
```
#include <stdio.h>
void main(void)
{
        char a=10;
        do
        {
                a=a-1;
```

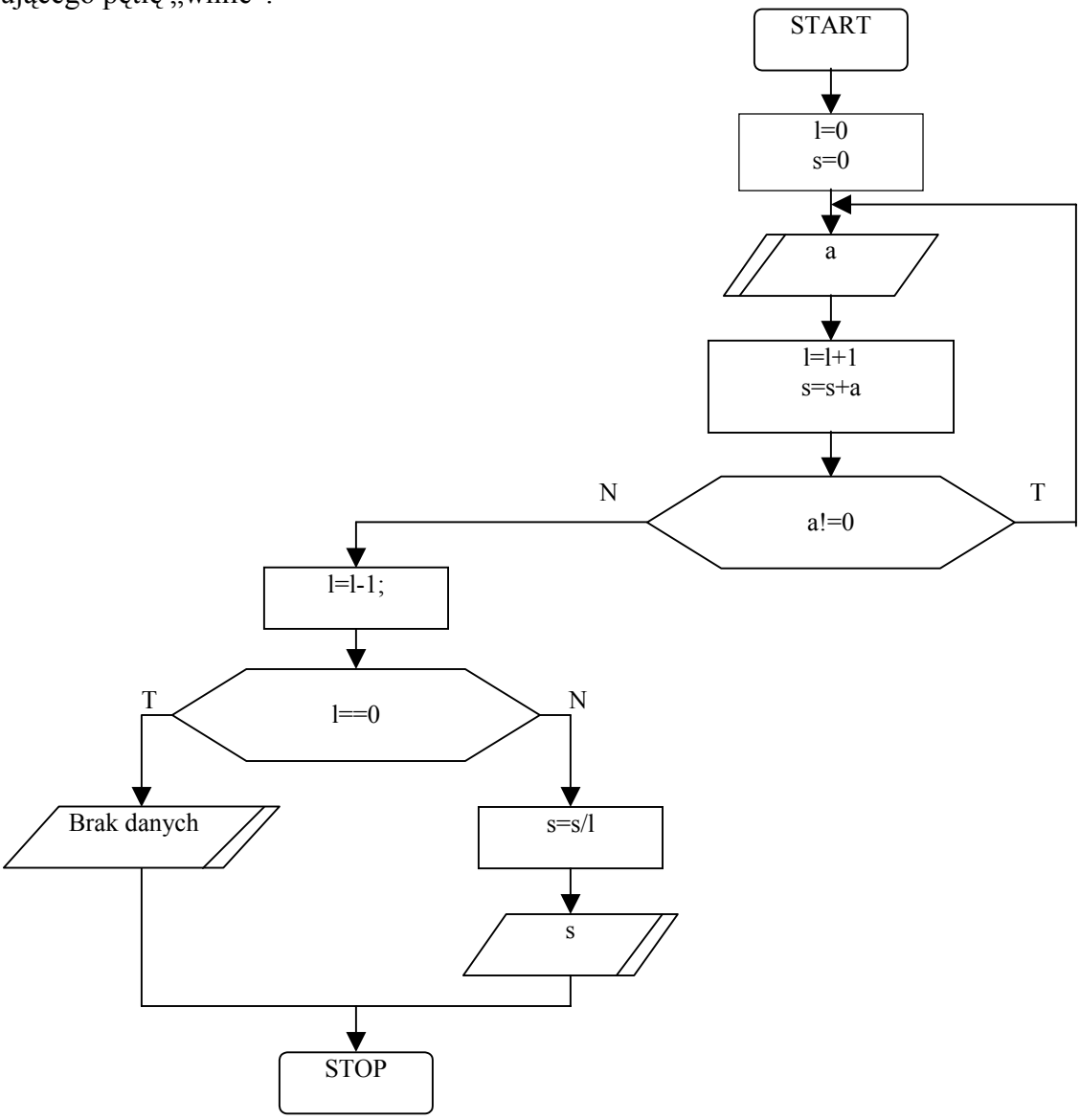
```
                printf("a=%d\t",a);
        }
        while(a>0);
}
```

3.

```
#include <stdio.h>
void main(void)
{
        char a=10;
        do
        {
                a=a-1;
                if(a%2==1)
                        printf("a=%d\t",a);
                else
                        printf("a=%d\t",2*a);
        }
        while(a>0);
}
```

Zadania:

- Uruchomić przykładowe programy i przeanalizować wyniki ich działania.
- Poniżej podany jest algorytm wyliczania średniej arytmetycznej z serii danych zakończonych wartością 0.0 (to samo zagadnienie było rozważane na poprzednich zajęciach). Obecny algorytm dostosowano do zastosowania instrukcji „do while". Należy napisać program według zaproponowanego algorytmu i porównać z programem napisanym według algorytmu stosującego pętlę „while".

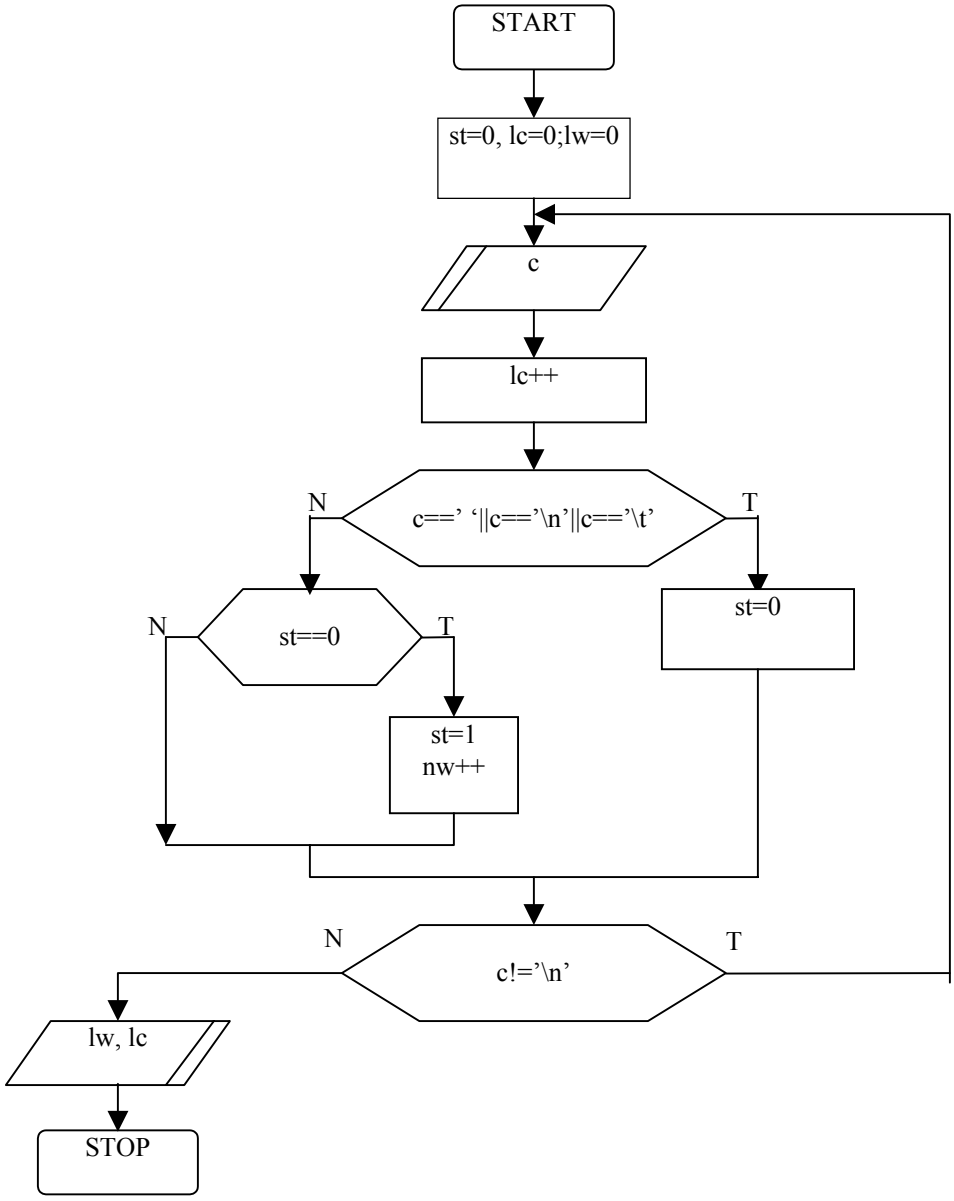


- Podany poniżej program umożliwia powielenie pojedynczej linii znaków wprowadzonych przez użytkownika z konsoli. Program czyta pojedynczy znak z wejścia, następnie wypisuje go na konsoli. Odczytywanie znaków trwa do momentu wykrycia znaku końca wiersza (\n):

```
#include <stdio.h>
void main(void)
{   int c;
```

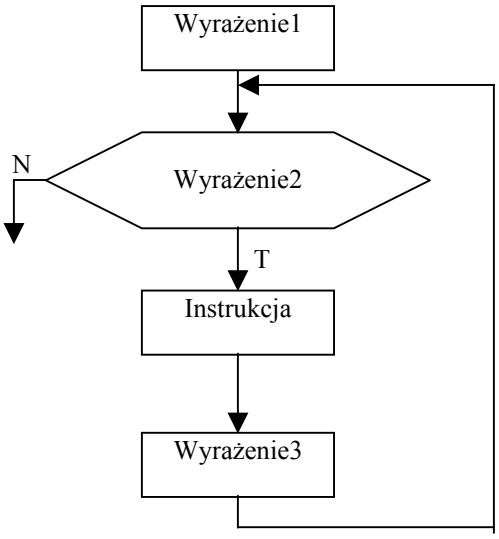
```
    do
    {
            c=getchar();
            putchar(c);
    }
    while(c!='\n');
}
```

Program jest odpowiednikiem programu z poprzednich zajęć, ale stosuje pętlę „do while”
Program należy uzupełnić o fragment kodu wyliczający ilość znaków oraz ilość wyrazów we wprowadzonej linii tekstu. Poniżej zaproponowano algorytm rozwiązania zadania:



Zagadnienie 2. (instrukcja cyklu: for)

Praktyka programistyczna wskazuje, że często spotyka się pewien ciąg instrukcji, który można opisać za pomocą następującego schematu blokowego:



Powyższy schemat blokowy można łatwo zapisać przy pomocy pętli „while”:

```
Wyrażenie1
while (Wyrażenie2)
{
    Instrukcja
    Wyrażenie3
}
```

Z uwagi na częste stosowanie opisanego ciągu operacji, w języku C zaproponowano zastąpienie go osobną instrukcją pętli „for”:

```
for(Wyrażenie1;Wyrażenie2;Wyrażenie3)
Instrukcja
```

Przykłady instrukcji „for”:

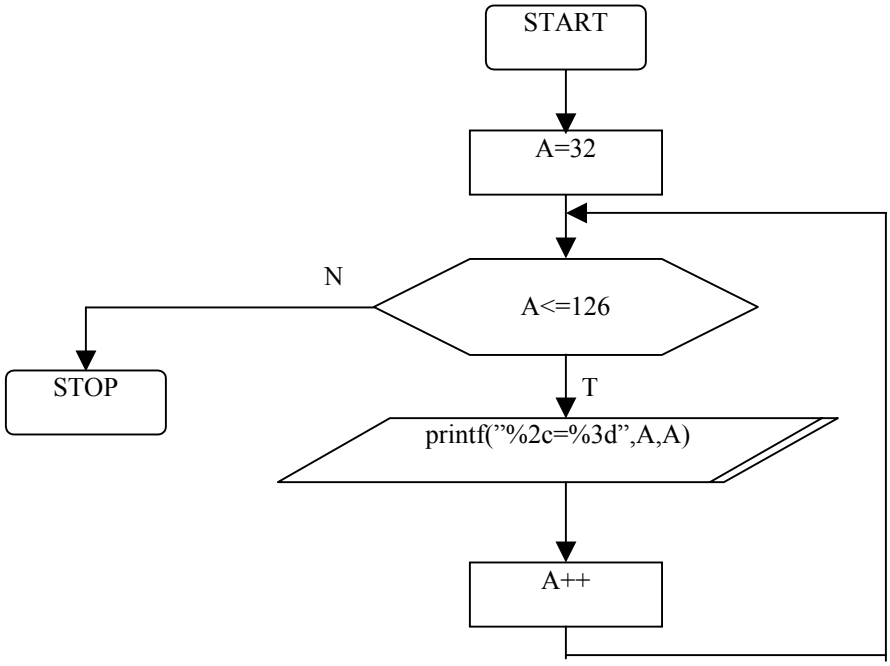
1.
```
#include <stdio.h>
void main(void)
{
	char a;
	for(a=10;a>0;a--);
	printf("a=%d\t",a);
}
```

2.
```
#include <stdio.h>
void main(void)
{
	char a;
	for(a=10;a>0;a--)
	printf("a=%d\t",a);
}
```

3.
```
#include <stdio.h>
void main(void)
{
	char a;
	for(a=10;a>0;a--)
	{
		if(a%2==1)
			printf("a=%d\t",a);
		else
			printf("a=%d\t",2*a);
	}
}
```

Zadania:

- Uruchomić przykładowe programy i przeanalizować wyniki ich działania.
- Zaproponować rozwiązanie zadania z wypisywaniem kolejnych potęg liczby 2 przy pomocy pętli „for” (treść zadania i algorytm zaproponowano na poprzednich zajęciach).
- Napisać program wypisujący na ekranie komputera tablicę znaków ASCII. Podstawowy zestaw znaków ASCII kodujących litery mieści się w przedziale <32,126>. Program powinien wypisać na ekranie pary: znak = kod znaku. Poniżej podano propozycję algorytmu rozwiązującego zagadnienie. Zadanie rozwiązać stosując pętle „while” i „for”.


START
A=32
N
A<=126
STOP
T
printf(”%2c=%3d”,A,A)
A++

Zagadnienie 3. (instrukcja przerwania pętli: break)

Instrukcja „break” daje możliwość wcześniejszego opuszczenia pętli („while”, „do-while”, „for”), bez potrzeby czekania aż wykona się pełny cykl obliczeniowy, określony warunkiem kontynuacji (zakończenia) obliczeń. Instrukcja powoduje natychmiastowy wyskok z najbardziej zagnieżdżonej pętli, w której występuje.

Rozważmy program powielający linię tekstu na ekranie konsoli:

```
#include <stdio.h>
void main(void)
{   int c;
        do
        {
                c=getchar();
                putchar(c);
        }
        while(c!='\n');
}
```

Rozbudowa programu w następujący sposób:

```
#include <stdio.h>
void main(void)
{   int c;
        do
        {
                c=getchar();
                if(c==' '||c=='\t') break;
                putchar(c);
        }
        while(c!='\n');
}
```

powoduje, że pętla odczytu znaków zostaje przerwana z chwilą napotkania pierwszego białego znaku. Stąd, jeśli użytkownik rozpocznie pisanie od białego znaku, to program w odpowiedzi nic nie wypisze, a jeśli użytkownik rozpocznie pisanie od dowolnego znaku różnego od białego znaku, to program powieli na ekranie tylko pierwszy wyraz.

Zadania:

- Uruchomić i przeanalizować przykładowy program

Zagadnienie 4. (instrukcja kontynuowania: continue)

Instrukcja „continue” powoduje przerwanie bieżącego i wykonanie od początku następnego kroku zawierającej ją pętli „do-while”, „while” lub „for”. Dla pętli „while” i „do-while” oznacza to natychmiastowe sprawdzenie warunku zatrzymania, natomiast w pętli „for” powoduje przekazanie sterowania do części przyrostowej („Wyrażenie3”).

Przykładowy program:

```
#include <stdio.h>
void main(void)
{
    int a, suma=0;
    do
    {
        scanf(“%d”,&a);
        suma=suma+a;
    }
    while(a!=0);
    printf(“suma=%d”,suma);
}
```

sumuje ciąg liczb podawany na jego wejściu do momentu podania jako elementu ciągu wartości 0.

Rozbudowa programu w następujący sposób:

```
#include <stdio.h>
void main(void)
{
    int a, suma=0;
    do
    {
        scanf(“%d”,&a);
        if(a<0) continue;
        suma=suma+a;
    }
    while(a!=0);
    printf(“suma=%d”,suma);
}
```

powoduje, że program odrzuca w sumowaniu ujemne dane wejściowe.

Zadania:

- Uruchomić i przeanalizować przykładowy program