Wykład 03 w dniu 15.X.08r

```
1 #include <stdio.h>
```

Zadanie 1: Dla danego $n \in \mathbb{N}$ wyświetlić na ekranie ciąg liczb: $0, 1, 2, \ldots, n-1$.

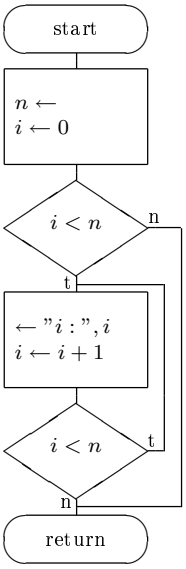
Rozwiązanie teoretyczne: W matematyce pisze się ... a w informatyce jest specjalna konstrukcja *języka programowania*[1] o nazwie: *„pętla*[2]*”*

Pseudokod:

Dane: n
Wynik: "i: ",i
for $i \leftarrow 0$ **to** $n-1$ **do**
  $i \rightarrow$
end

Algorytm 1: Pętla: **for{i=0 ; i<n ; i=i+1}{...}**

Schemat blokowy:



pętla: **for{i=0 ; i<n ; i=i+1}{...}**

Źródło w pliku `a.c`:

```
1  #include <stdio.h>
2
3  int main()
4  {
5      int i,n;
6
7      printf("n: ");
8      scanf("%i",&n);
9      for( i=0 ; i<n ; i=i+1 )
10     {
```

[1] *Wikipedia:* W programowaniu *strukturalnym*.

[2] *Wikipedia:* W programowaniu *pętla* to jedna z trzech podstawowych konstrukcji *programowania strukturalnego*. Umożliwia cykliczne wykonywanie ciągu instrukcji aż do momentu zajścia warunku zakończenia pętli.

```
11          printf("i: %2i\n",i);
12      }
13      return 0;
14  }
```

Wynik działania dla $n = 7$:

```
n: 7
i:  0
i:  1
i:  2
i:  3
i:  4
i:  5
i:  6
```

. . . . . .

Zadanie 2: Dla danego $n \in \mathbb{N}$ obliczyć sumę

$$S_n = 1 + 2 + \ldots + n = \sum_{i=1}^{n} i$$

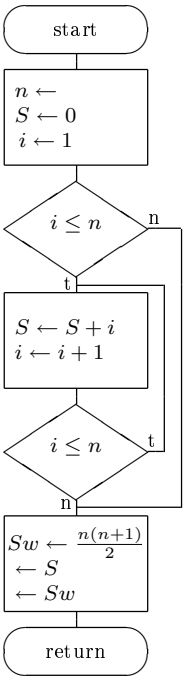
Matematycy wiedzą, że $S_n = \frac{n(n+1)}{2}$.

Pseudokod:

Dane: n
Wynik: S, S_w
$S \leftarrow 0$
for $i \leftarrow 1$ **to** n **do**
  $S \leftarrow S + i$
end
$S_w \leftarrow \frac{n(n+1)}{2}$

Algorytm 2: Obliczenie sumy: $S_n = 1 + 2 + \ldots + n$

Schemat blokowy:



suma: $S = 1 + 2 + \ldots + n$

Źródło w pliku `w3b.c`:

```
#include <stdio.h>

int main()
{
    int i,n,S,Sw;

    printf("n: ");
    scanf("%i",&n);
/* UWAGA:
     badanie warunku 0<n nie jest uwzglednione
     w pseudokodzie i schemacie blokowym
*/
    if(n<1)
    {
        printf("ERROR: n<1 !?\n");
        return -1;
    }
    S = 0;
    for( i=1 ; i<=n ; i=i+1 )
    {
        S = S+i;
    }
    Sw = n*(n+1)/2;
    printf(" S: %3i\nSw: %3i\n",S,Sw);
    return 0;
}
```

. . . . . .

Zadanie 3: Dla danego $n \in \mathbb{N}$ i $x \in \mathbb{R}$ obliczyć sumę

$$S_n = 1 + x + x^2 + \ldots + x^n = \sum_{i=0}^{n} x^i$$

Algorytm obliczeń będzie opierać się na tożsamości:

$$S_n = 1 + x + x^2 + \ldots + x^n = 1 + x(1 + x(1 + \ldots x(1 + x))\ldots)$$

np. dla $n = 4$ jest to:

$$S_4 = 1 + x + x^2 + x^3 + x^4 = 1 + x(1 + x(1 + x(1 + x)))$$

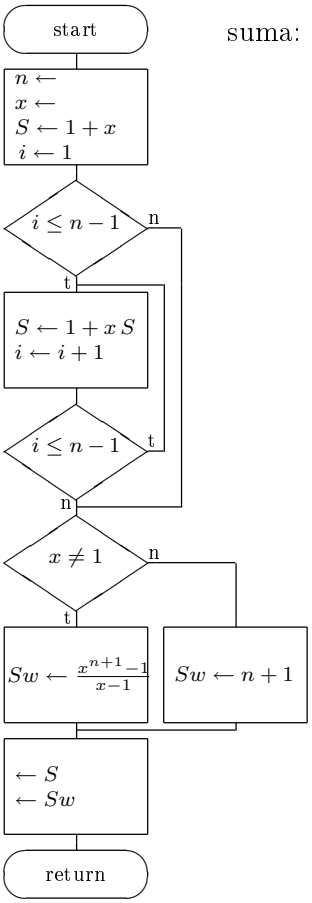
Tu też matematycy więdzą, że $S_n = \frac{x^{n+1}-1}{x-1}$ dla $x \neq 1$ oraz $S_n = n + 1$ dla $x = 1$.

Pseudokod:

Dane: $n,\ x$
Wynik: $S,\ S_w$
$S \leftarrow 1 + x$
for $i \leftarrow 1$ **to** n **do**
  $S \leftarrow 1 + x \cdot S$
end
if $x \neq 1$ **then**
  $S_w \leftarrow \frac{x^{n+1}-1}{x-1}$
else
  $S_w \leftarrow n + 1$
end

Algorytm 3: Obliczenie sumy $S = 1 + x + x^2 + \ldots + x^n$

Schemat blokowy:



suma: $S = 1 + x + x^2 \ldots + x^n$

Źródło w pliku[3] `w3c.c`:

```
1 #include <stdio.h>
2 #include <math.h>
3
4 int main()
5 {
```

[3] kompilacja: `gcc -Wall w3c.c -o c -lm`

```
6     double x,S,Sw;
7     int i,n;
8
9     printf("n: ");
10    scanf("%i",&n);
11 /* UWAGA:
12      badanie warunku 0<n nie jest uwzglednione
13      w pseudokodzie i schemacie blokowym
14 */
15    if(n<1)
16    {
17        printf("ERROR: n<1 !?\n");
18        return -1;
19    }
20    printf("x: ");
21    scanf("%lf",&x);
22    S = 1.0+x;
23    for( i=1 ; i<=n-1 ; i=i+1 )
24    {
25        S = 1.0+x*S;
26    }
27    if( x!=1.0 )
28    {
29        Sw = (pow(x,n+1.0)-1.0)/(x-1.0);
30    }
31    else
32    {
33        Sw = n+1.0;
34    }
35    printf(" S: %8.2lf\nSw: %8.2lf\n",S,Sw);
36    return 0;
37 }
```

. . . . . .

Zadanie 4: Napisać program, który dla $x \in \mathbb{R}$ obliczy wartość wielomianu

$$w_3(x) = 0.5x^3 - 3x^2 + 5.5x - 3$$

Rozwiązanie: Tutaj matematycznie zadanie jest oczywiste. Zwłaszcza, że wielomian ten ma pierwiastki 1, 2, 3 (tak został dobrany !) a więc można go przedstawić w postaci:

$$w_3(x) = \frac{1}{2}(x-1)(x-2)(x-3)$$

co umożliwi nam łatwe sprawdzenie poprawności otrzymywanych wyników. W algorytmie programu zastosujemy tożsamość

$$a_0x^3 + a_1x^2 + a_2x^1 + a_3 = ((a_0x + a_1)x + a_2)x + a_3,$$

która jest podstawą *schematu Hornera*[4] obliczania wartości wielomianu $w_3(x)$ dla danej wartości argumentu $x \in \mathbb{R}$ przy minimalnej liczbie mnożeń.

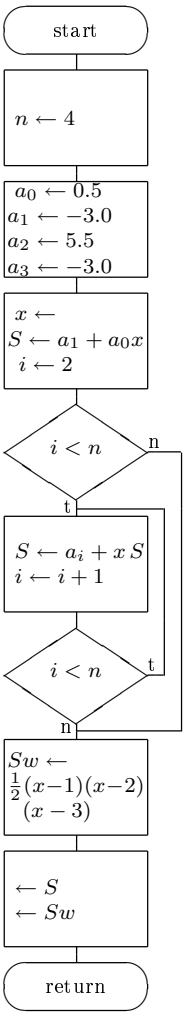
[4] patrz: *Wikipedia*

Pseudokod:

Dane: x
Wynik: $S,\ S_w$
1 $n \leftarrow 4$
2 $a_0 \leftarrow 0.5$
3 $a_1 \leftarrow -3.0$
4 $a_2 \leftarrow 5.5$
5 $a_3 \leftarrow -3.0$
6 $S \leftarrow a_1 + a_0 x$
7 **for** $i \leftarrow 2$ **to** $n-1$ **do**
8 $\quad S \leftarrow a_i + x \cdot S$
9 **end**
10 $S_w \leftarrow \frac{1}{2}(x-1)(x-2)(x-3)$

Algorytm 4: Obliczenie wartości $w_3(x) = 0.5x^3 - 3x^2 + 5.5x - 3$

Schemat blokowy:



wartość: $w_3(x) = 0.5x^3 - 3x^2 + 5.5x - 3$

Źródło w pliku `w3d.c`:

```
1  #include <stdio.h>
2
3  const int n = 4;
4
5  int main()
6  {
7      double x,S,Sw;
8      double a[n];
9      int i;
10
11     a[0] =  0.5;
12     a[1] = -3.0;
13     a[2] =  5.5;
14     a[3] = -3.0;
15     printf("x: ");
16     scanf("%lf",&x);
17     S = a[1]+a[0]*x;
18     for( i=2 ; i<n ; i=i+1 )
19     {
20         S = a[i]+x*S;
21     }
22     Sw = 0.5*(x-1.0)*(x-2.0)*(x-3.0);
23     printf("   S: %8.2lf\nSpr.: %8.2lf\n",S,Sw);
24     return 0;
25 }
```

.....

Wrocław, dnia 23/10/2008