WYKŁAD 8

Funkcje i algorytmy rekurencyjne
Proste przykłady

Programy: c3_1.c ....., c3_6.c

Tomasz Zieliński

METODY REKURENCYJNE (1) - program c3_1

==========

Funkcja rekurencyjna to funkcja, która wywołuje samą siebie.

==========

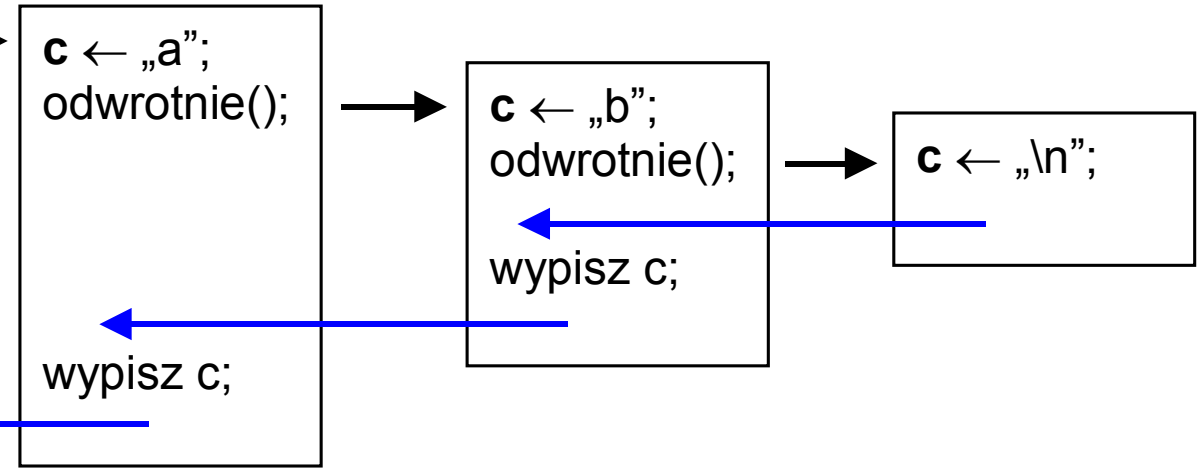
```
/* Przyklad 3.1 - wypisz znaki z klawiatury w odwrotnej kolejnosci  */

#include <stdio.h>

void odwrotnie( void );

void main()
{
    printf("\nPisz w linii. Zakoncz <Enter>. \n\n");
    odwrotnie();
}

void  odwrotnie( void )
{
   char  c;                          // wczytaj znak z klawiatury i podstaw jego
   if (  (c=getchar() ) != '\n' )    // kod ASCII do c; jeśli nie jest to koniec linii, to:
      {                              //
            odwrotnie();             // wywołaj ponownie funkcję odwrotnie()
            printf( "%c", c);        // wyświetl na ekranie znak c
      }
}
```

METODY REKURENCYJNE (2) - program c3_1

Wypisz w odwrotnej kolejności znaki z klawiatury

ZAŁOŻENIE: z klawiatury: „ab [ENTER]”

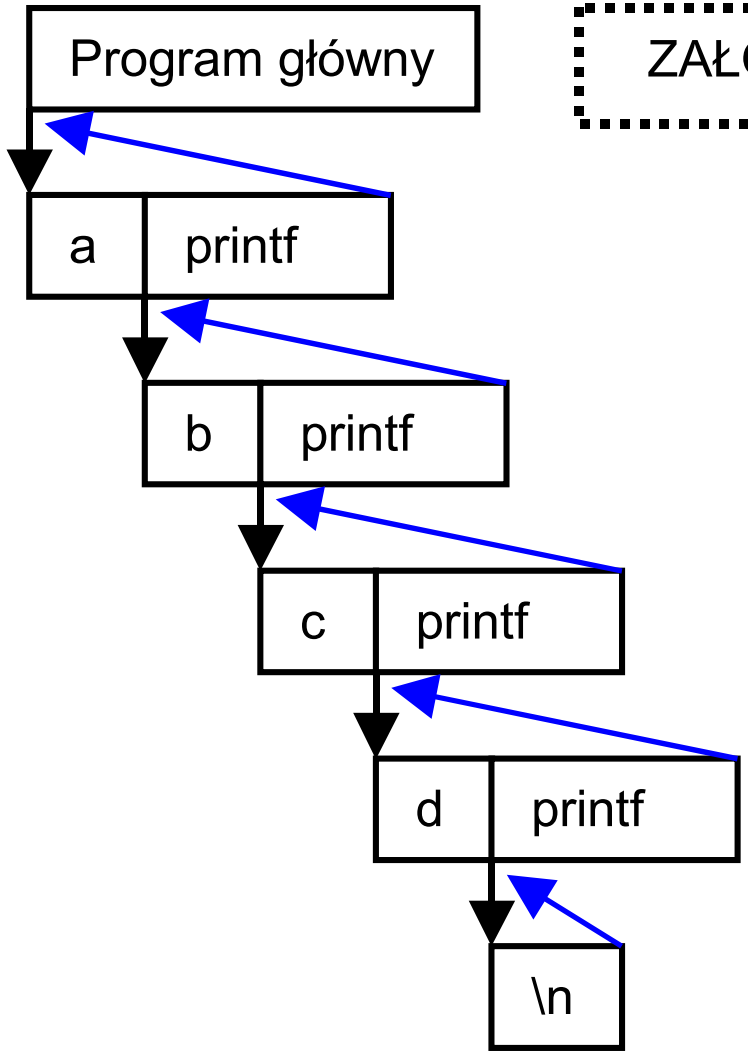
```
void main()
{   .....
    odwrotnie();
    .....
}

void  odwrotnie( void )
{
   char  c;
   if ( (c=getchar()) != '\n' )
      {
         odwrotnie();
         printf( "%c", c);
      }
}
```



c jest lokalną zmienną każdego z wywołań funkcji, która ma różną wartość w każdym z wywołań

METODY REKURENCYJNE (3) - program c3_1

Wypisz w odwrotnej kolejności znaki z klawiatury



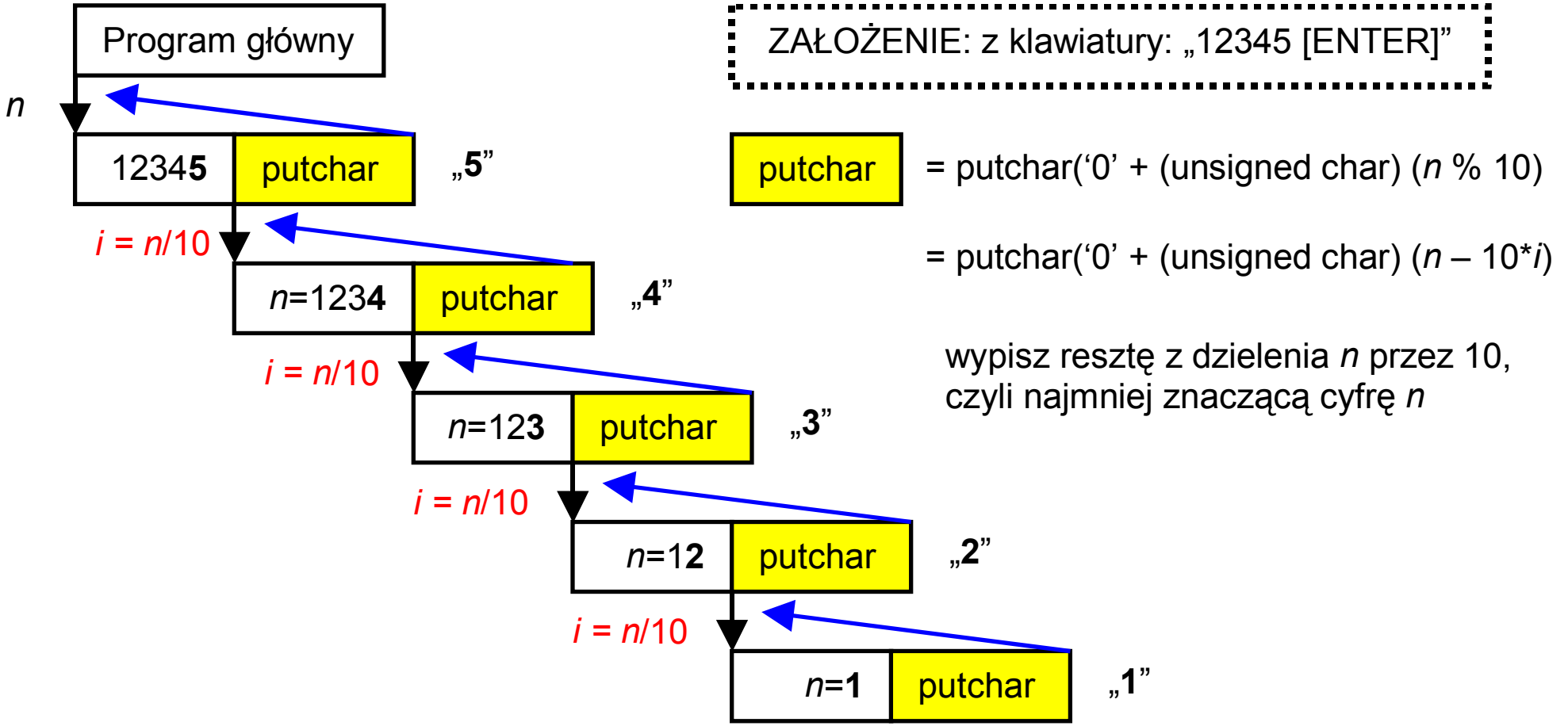
ZAŁOŻENIE: z klawiatury: „abcd [ENTER]”

METODY REKURENCYJNE (4) - program c3_2
Zamiana integer na ASCII

```
/* Przyklad 3.2  -  wypisz ma ekranie liczbę calkowitą                */
/*                  czyli INTEGER na ASCII                            */

#include <stdio.h>
#include <math.h>

void wypisz( int n );

/* program glowny ------------------------------------------------------------ */

void main()
{
    int   x;

    printf(" \n  jaka liczba calkowita ? ");
    scanf("%d", &x);
    printf(" \n  podane   = %d \n", x);

    printf(" rekursja = ");  wypisz( x );
    printf("\n");
}
```

```
/* Funkcja rekurencyjna ---------------------------------------------------- */

void  wypisz( int n )
{
    int  i;

    if ( n < 0 )  {  putchar('-'); n = -n; }          //  jeśli liczba ujemna,
                                                      //  to napisz „-” oraz zaneguj liczbę

    if ( (i=n/10) != 0 )  wypisz( i );                //  jeśli „i”, czyli część całkowita z „n/10”,
                                                      //  jest różna od zera, to wywołaj ponownie
                                                      //  ale z argumentem „i”

     putchar(‘0’+n%10);                               //  to (reszta z dzielenia n przez 10)
//   putchar( '0' + (unsigned char) (n - 10 * i) );   //  lub to (czyli najmniej znacząca cyfra)
}
```

METODY REKURENCYJNE (5) - program c3_2
Zamiana integer na ASCII



ZAŁOŻENIE: z klawiatury: „12345 [ENTER]”

putchar = putchar(‘0’ + (unsigned char) (*n* % 10)

= putchar(‘0’ + (unsigned char) (*n* – 10**i*)

wypisz resztę z dzielenia *n* przez 10, czyli najmniej znaczącą cyfrę *n*

METODY REKURENCYJNE (6) - program c3_3

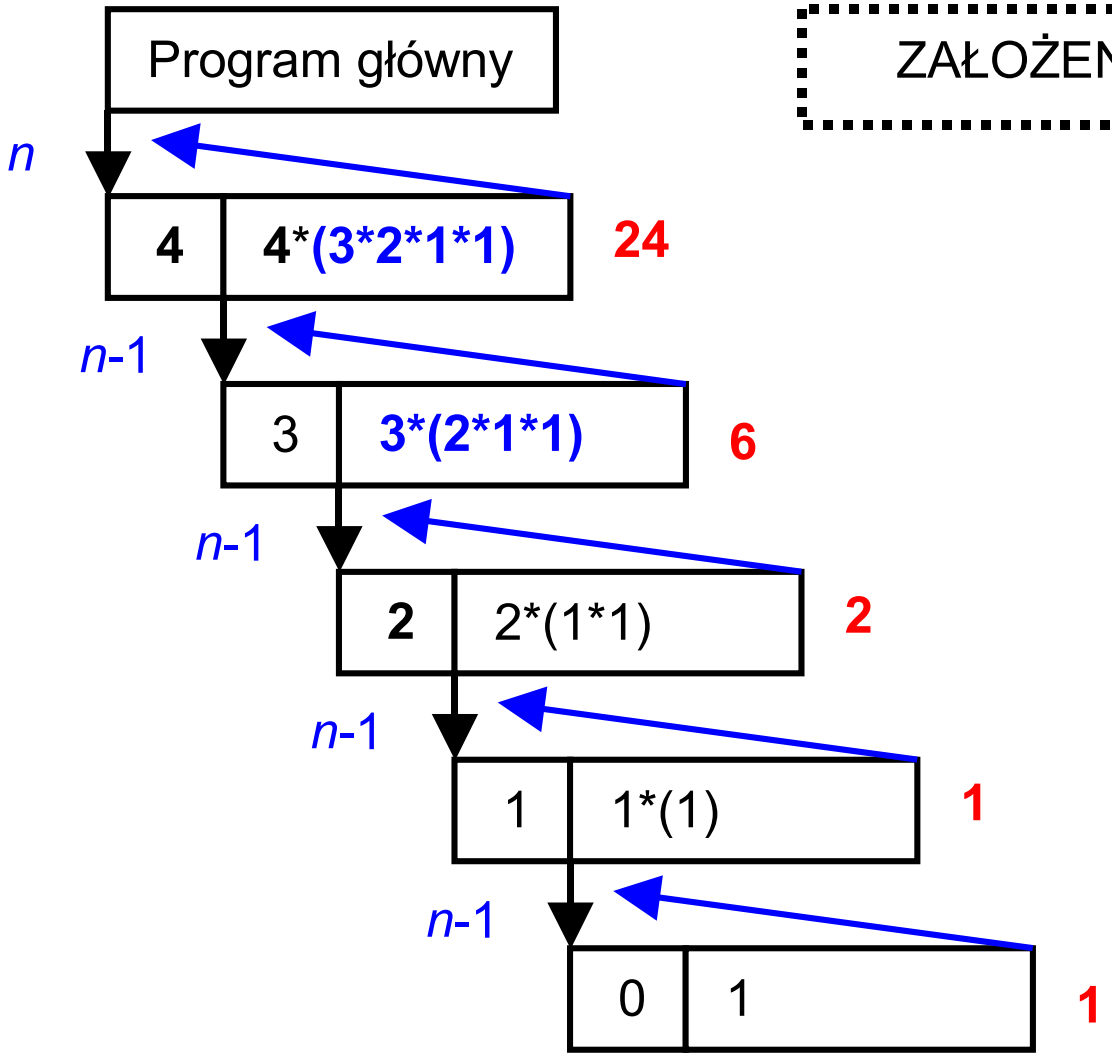
n! czyli *n-silnia*

```
/*  Przyklad 3.3 - METODY REKURENCYJNE - oblicz n!          */
/*  n! = 1*2*3*...*(n-1)*n  np. 5! = 1*2*3*4*5              */

#include <stdio.h>

long  silnia( long n );

/* program glowny ------------------------------------------------------- */

void main()
{
    long   n, x, iloczyn;

    printf("\n   Obliczam funkcje n !   Podaj n ? [max 12] ");
    scanf("%ld", &n);

/* Metoda iteracyjna */

    iloczyn = 1;
    for( x = n; x > 0; x--)
          {  iloczyn = iloczyn * x;  }
    printf( "\n  Metoda iteracyjna --> n! = %ld \n", iloczyn );
```

```
/* Metoda rekurencyjna */

     iloczyn = silnia( n );
     printf( "\n  Metoda rekurencyjna --> n! = %ld \n", iloczyn);
}

/* Funkcja rekurencyjna ---------------------------------------------------- */

long  silnia( long n )
{
   long  x, y;

   if( n == 0L ) return(1);          // L znaczy long
   x = n-1;
   y = silnia(x);                    /* rekurencyjne wywolanie */
   return(n*y);
}

long  silnia( long n )
{
   if( n == 0L ) return(1);
   return(n* silnia(n-1));           /* rekurencyjne wywolanie */
}
```

METODY REKURENCYJNE (7) - program c3_3

n! czyli *n-silnia*

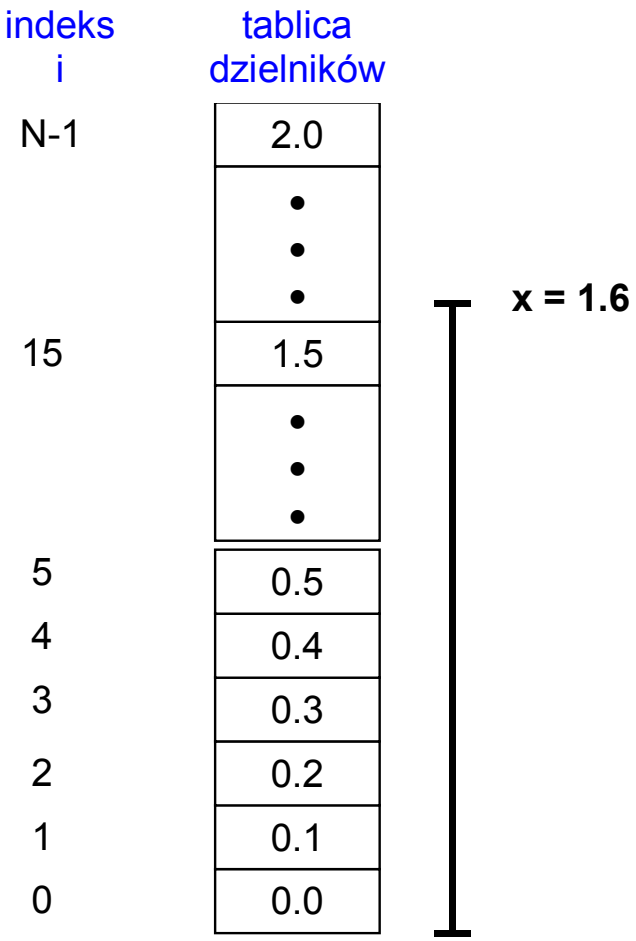


ZAŁOŻENIE: z klawiatury: „4 [ENTER]”

```
long  silnia( long n )
{
     long  x, y;

     if( n == 0L ) return(1);
     x = n-1;
     y = silnia(x);
     return(n*y);
}

long  silnia( long n )
{
     if( n == 0L ) return(1);
     return( n*silnia(n-1) );
}
```

METODY REKURENCYJNE (7) - program c3_5

szukanie elementu metodą podziału zbioru na dwie części



ZADANIE: znaleźć indeks „i" tablicy, dla którego: tab[i] == x

ROZWIĄZANIE 1 (sekwencyjne):

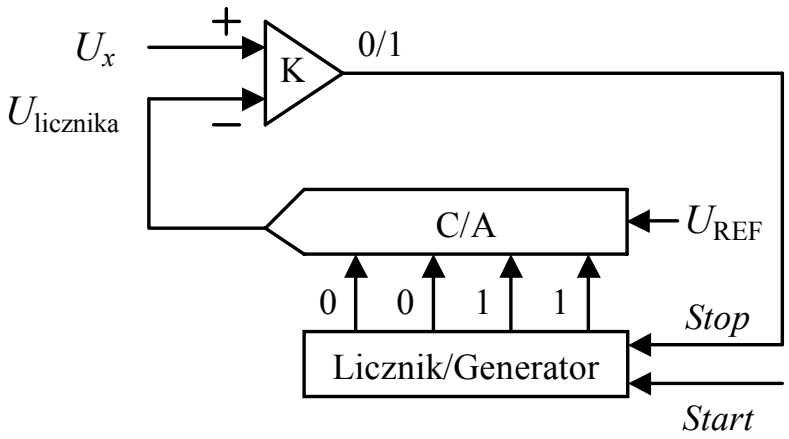
```
i = 0;
while( ( x != tab[i] ) & (i < N-1) )
        i++;
```

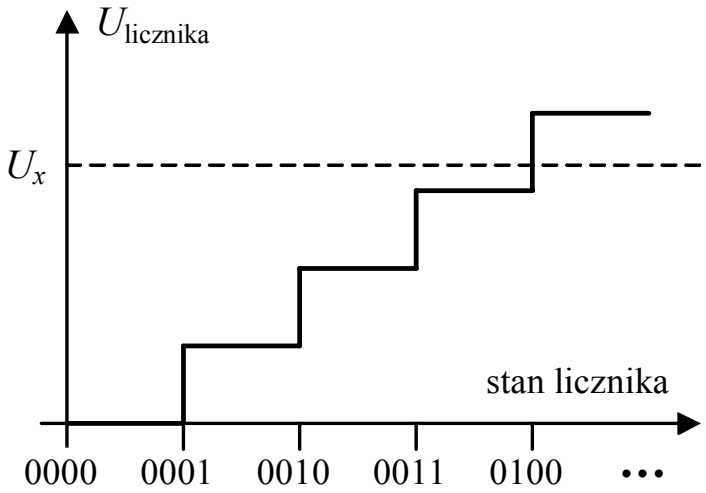
ROZWIĄZANIE 2 (rekurencyjne):

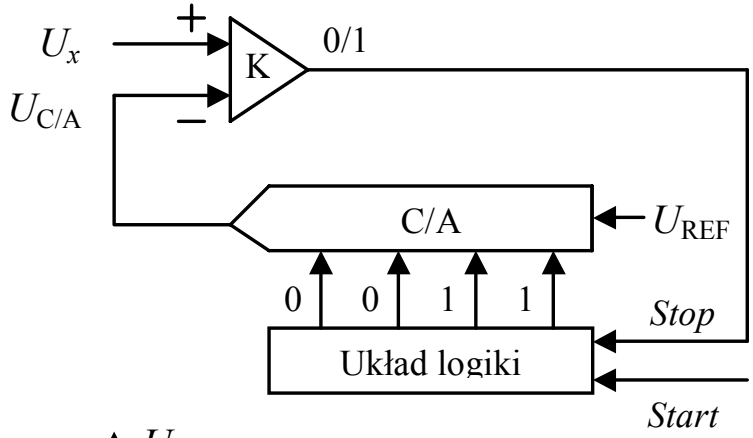
```
int szukaj( int tab[ ],  int x,  int dol,  int gora )
{    int  srodek;

     srodek = (dol + gora)/2;
     if ( x == tab[ srodek ] ) return( srodek);
     if ( x < tab[ srodek ] )
          return(  szukaj( tab,  x,  dol,  srodek - 1 )  );
     else
          return(  szukaj(  tab,  x,  srodek + 1,  gora )  );
}
```

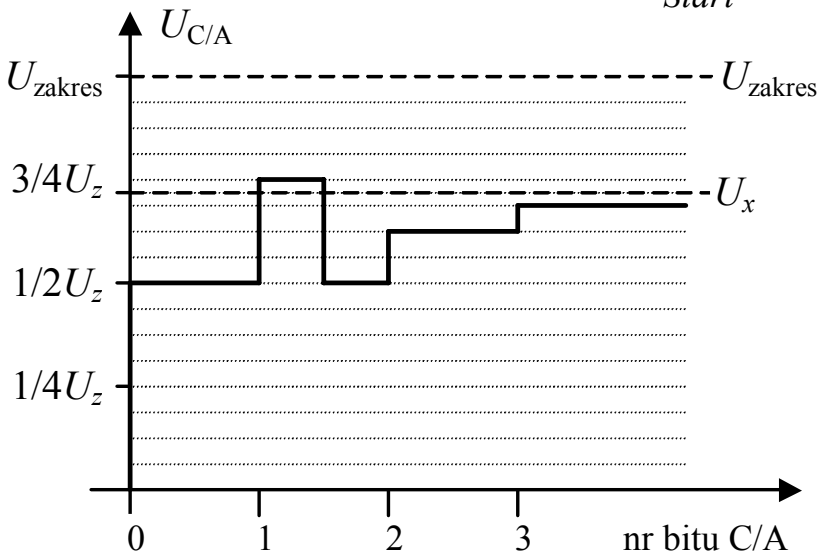
METODY REKURENCYJNE (8) - program c3_5

PRZYKŁAD: przetworniki A/C kompensacyjne: równomierny (po lewej) i wagowy (po prawej)









METODY REKURENCYJNE (9) - program c3_5
szukanie metodą podziału na dwie części

```
/* Przyklad 3.5 - METODY REKURENCYJNE - szukanie "dwojkowe"                    */
/*                    WEJŚCIE:     tab = [ 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 ]  */
/*                                 x = liczba naturalna z klawiatury           */
/*                    WYJŚCIE:     położenie (indeks) elementu x w tablicy tab[] */

#include <stdio.h>

int szukaj( int tab[ ], int x, int dol, int gora );

void main()/* program glowny ------------------------------------------------------------------ */
{
    int  tab[ 11 ] = { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9,10 };
    int  x, indeks, dol, gora;

    printf("\n  Jaka jest calkowita wartosc poszukiwanego elementu ? ");
    scanf("%d", &x);

    dol = 0;   gora = 10;

    indeks = szukaj(  tab,  x,  dol,  gora );

    printf( "\n  Indeks = %i \n", indeks);
}
```

```
/* funkcja rekursywna ---------------------------------------------------------- */

int szukaj( int tab[ ],  int x,  int dol,  int gora )
{
    int  srodek;

    if ( dol > gora )  return(-1);        /* -1 oznacza brak elementu w zbiorze */

    srodek = (dol + gora)/2;
    if ( x == tab[ srodek ] ) return( srodek);
    if ( x < tab[ srodek ] )
        return(  szukaj( tab,  x,  dol,  srodek - 1 )  );
    else
        return(  szukaj(  tab,  x,  srodek + 1,  gora )  );
}
```