lekcja 8
Rekurencja

Silnia – Euklides – Fibbonaci - Horner

Czym jest rekurencja? Gdy patrzymy w lustro, na które nakierujemy drugie – widzimy nieskończony ciąg odbić siebie samego. W matematyce i informatyce mamy do czynienia z rekurencją, gdy jakaś funkcja w swoim „wnętrzu" odwołuje się do siebie samej (takie przysłowiowe pudełko w pudełku).

Dlaczego stosujemy rekurencję, jeśli większość informatycznych problemów można by rozwiązać w sposób tradycyjny – iteracyjny? Po pierwsze jest to czasem bardziej naturalny sposób rozwiązania problemu, a po drugie zyskujemy zwięzły zapis problemu.

Co jest najważniejsze w rekurencji. Należy pamiętać o takim skonstruowaniu warunku zakończenia wywoływania funkcji wewnątrz nie samej, aby nie zapętliła się w nieskończoność, co może doprowadzić nawet do zawieszenia się systemu.

W algorytmice metodą rekurencyjną realizowane są min: algorytm Euklidesa, ciąg Fibonacciego, wyszukiwanie binarne, sortowanie szybkie, sortowanie przez scalanie, zamiana z dziesiętnego na dowolny inny system liczbowy, przeszukiwanie w głąb, podnoszenie do potęgi, wypisanie wyrazu wspak, silnia i wiele innych.

Zadanie - Silnia

W matematyce silnię zapisujemy za pomocą wykrzyknika i tak 3! jest równe 1*2*3 = 6, co można wyliczyć za pomocą pętli. Matematyczna (rekurencyjna) definicja silni:

dla n = 0 *silnia = 1*

dla n > 0 *silnia = silnia(n-1) * n*

Funkcję SILNIA definiujemy zgodnie z matematyczną definicją – dla wszystkich n większych od zera wywołujemy jeszcze raz funkcję z parametrem n-1

```
long SILNIA(int n){
  if (n==0) return 1;
  else return n*SILNIA(n-1);
}
...
cout << "n=";
int k;
cin>> k;
cout<<k<<"!="<<SILNIA(k)<<endl;
```

```
SILNIA
n=6
6!=720
```

Zadanie - Euklides NWD

Wyznaczamy największy wspólny dzielnik dwóch liczb, czyli taką możliwie największą liczbę, która dzieli obie liczby. Wybieramy większą z dwóch liczb i zamieniamy ją na różnicę większej i mniejszej. Czynność tą powtarzamy do momentu uzyskania dwóch takich samych wartości.

```
long EUKLIDES(int a, int b){
  if (a!=b)
    if (a>b) return EUKLIDES(a-b,b);
    else return EUKLIDES(a,b-a);
  return a;
}
...
cout << "a= b= ";
int a,b;
cin >> a >> b;
cout << "NWD("<<a<<","<<b<<")="
       << EUKLIDES(a,b) << endl;
```

```
EUKLIDES
a= b= 64 24
NWD(64,24)=8
```

Zadanie – Fibbonaci

Ciąg Fibonacciego definiujemy następująco: pierwszy i drugi element ciągu są równe 1. Każdy następny otrzymujemy dodając do siebie dwa poprzednie. Matematycznie wygląda to następująco:

$$F_n = \begin{cases} 1 & , dla\ n = 1 \\ 1 & , dla\ n = 2 \\ F_{n-2} + F_{n-1} & , dla\ n > 2 \end{cases}$$

```
long FIBBONACI(int n){
  if (n<3) return 1;
  else return FIBBONACI(n-2) + FIBBONACI(n-1);
}
...
cout << "miesiące=";
int f;
cin >> f;
cout << "królików =" << FIBBONACI(f) << endl;
```

```
FIBBONACI
miesiące=8
królików =21
```

Zadanie - Wspak

Wczytujemy tekst z klawiatury. Po naciśnięciu klawisza ENTER tekst zostaje wypisany w kolejności odwrotnej.

```
WSPAK
Napisz coś: libront
tnorbil
```

- wczytujemy z klawiatury jeden znak GETCHAR
- jeżeli znak nie jest ENTERem, to
- rekurencyjnie wywołujemy funkcję ODWROTNIE – wczytaj znak
- gdy rekurencja się skończy – ENTER
- wyświetlane są znaki PRINTF, a kolejne zakończenia funkcji wyświetlą znaki od końca

```
#include <cstdio>
...
void ODWROTNIE(){
  char znak;
  znak=getchar();
  if (znak!='\n') {
    ODWROTNIE();
    printf("%c",znak);
  }
}
```

Zadanie - Horner

Każdy wielomian można zapisać w postaci:

$$W(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_{n-1} x^{n-1} + a_n x^n$$

Przykładowo: f(-3)=3x^4 +7x^3=2x^2+10x-3. Jeśli chcemy obliczyć wartość tego wielomianu dla x=3, to należy wykonać 5 mnożeń. Jeśli rośnie stopień wielomianu, tych mnożeń będziemy musieli wykonać coraz więcej. Ponieważ operacja mnożenia jest dla przeciętnego komputera czasochłonna, dlatego poszukuje się metod, które obliczają wartość wielomianów z mniejszą liczbą mnożeń. Jednym z rozwiązań jest tzw. schemat Hornera

W programie współczynniki wielomianu zapisane są w tablicy WSP: w elemencie zerowym ostatni współczynnik (potęga 0), w pierwszym – 1, itd. Parametry funkcji HORNER:
n – największa potęga wielomianu
a[] – tablica ze współczynnikami
x – argument wielomianu

Jeśli wielomian jest stopnia zerowego, to wynikiem jest komórka zerowa tablicy. W przeciwnym razie wykonujemy rekurencyjne mnożenie i dodajemy współczynnik z tablicy: Wn-1(x)*x + A[n]

```
double HORNER(int n, double a[], double x){
  if (n==0) return a[0];
  else return HORNER(n-1, a, x) * x + a[n];
}
...
int s=4;
double wsp[]={-3,10,-2,7,3};
double x=-3;
cout << HORNER(s,wsp,x) << endl;
```

```
HORNER
równanie f(-3)=3x^4 +7x^3=2x^2+10x-3
-549
```

Zadania

1) *Stosując rekurencję oblicz wartość dwumianu Newtona (n,k)=n!/(k!*(n-k)!)*

```
DWUMIAN NEWTONA
Wpisz n i k: 5 3
10
```

2) *Stosując rekurencję oblicz wartość sumy kolejnych liczb naturalnych suma(n)=n+suma(n-1)*

```
SUMA NATURALNYCH
Ile liczb: 10
55
```

3) *Pewien ciąg opisany na poniższym obrazku generuje wyrazy w sposób rekurencyjny. Napisz program, który wyliczy wartość n-tego wyrazu ciągu.*

```
CIĄG
Numer: 6
Wartość: -0.03125
```

$$a_n = \begin{cases} 1 & dla\ n = 1 \\ 0.5 & dla\ n = 2 \\ -a_{n-1} \cdot a_{n-2} & dla\ n > 2 \end{cases}$$

4) *Napisz program, który zapisze podaną liczbę dziesiętną w systemie dwójkowym. Rozwiąż zadanie rekurencyjnie.*

```
DECBIN
dziesiętnie: 87
dwójkowo: 1010111
```

5) *Napisz program, który wyznaczy sumę cyfr liczby naturalnej z zakresu. Rozwiąż zadanie metodą rekurencyjną.*

```
SUMA CYFR
Liczba: 123456
Suma cyfr: 21
```