Lekcja 11. (kl. IV. PR)

Temat: Programowanie strukturalne kontra obiektowe, czyli dane w centrum uwagi. **(1g.)**

Cele lekcji:

– poznanie i zrozumienie programowania obiektowego

Uczeń:

- poznanie nowego sposobu myślenia o pisaniu programów: programowanie obiektowe, w którym centralnym pojęciem nie są procedury, lecz typy danych (klasy) i zmienne (obiekty) poznanie podstawowych pojęć programowania obiektowego: klasa, obiekt, pole, metoda, a także – w dużym skrócie – mechanizmy dziedziczenia i polimorfizmu

Podręcznik str. 56

Przebieg lekcji:

1. Zapoznanie się z celami lekcji.
2. Programowanie strukturalne
3. Programowanie obiektowe: klasy, obiekty, pola i metody
4. Dziedziczenie i polimorfizm
5. Ćwiczenia praktyczne.

Zadania do wykonania:

- opis – podręcznik str. 60

Znajdowanie lidera w zbiorze

Lider to taka wartość w zbiorze

$$n$$

elementów, która powtarza się więcej niż

$$n \; div \; 2$$

razy. Jeśli istnieje taka wartość, to jest ona tylko jedna. Prześledźmy przykłady:

ciąg liczb

$$1, 3, 4, 3, 2, 1, 1$$

nie posiada lidera, ponieważ żadna z liczb nie wystąpiła co najmniej

$$7 \; div \; 2 + 1 = 4$$

razy.

Ciąg

$$1, 2, 2, 3, 3, 3, 3, 2$$

także nie posiada lidera, natomiast dla liczb

$$1, 2, 2, 3, 3, 3, 3, 2, 3$$

liderem jest liczba

$$3$$

.

Strategia jest następująca. Przeszukujemy liniowo kolejne elementy i w danym etapie dwa różne elementy zbioru wykreślamy - tak jak by ich nie było. Jeśli lider występuje w zbiorze, to jego "**pozycja**" w otrzymanym podzbiorze się nie zmieni. Przeanalizujmy następujący przykład:

$$1, 2, 1, 3, 3, 3, 3, 2, 3$$

$$\not{1}, \not{2}, 1, 3, 3, 3, 3, 2, 3$$

i pozostał na zbiór, w którym nadal mamy lidera:

$$1, 3, 3, 3, 3, 2, 3$$

Redukując w ten sposób elementy dochodzimy do sytuacji, gdy pozostanie element, który jest kandydatem na lidera. Musimy teraz tylko liniowo zliczyć i sprawdzić, czy dana wartość występuje więcej niż

$$n \; div \; 2$$

razy.

Warto podkreślić, że znaleziony element nie musi być liderem. Prześledźmy przykład, w którym nie ma lidera:

$$2, 3, 1, 1, 1, 4$$

Po wykreśleniu dwóch początkowych wyrazów pozostaje podzbiór

$$1, 1, 1, 4$$

,

w którym jest lider.

W programie przedstawionym poniżej ważną rolę odgrywa zmienna

$$dopary$$

. Odpowiedzialna jest ona za kontrolowanie wykreślania dwóch elementów o różnych wartościach. Jeśli wykreślimy wszystkie elementy, oznacza to, że żadna wartość nie spełnia oczekiwań. Przeanalizujmy przypadek:

liczby	2	3	4	4	3	5
lider	2	3	4	4	4	4
dopary	1	0	1	2	1	0

W przypadku znalezienia liczby różnej od aktualnego lidera, wykreślamy ją zmniejszając wartość zmiennej

$$dopary$$

. Gdy znajdziemy taką samą, wartość zmiennej inkrementujemy. Jeśli

$$dopary$$

będzie liczbą większą od zera, oznacza to, że zmienna

$$lider$$

przechowuje potencjalnego lidera zbioru.

```
//algorytm.edu.pl
#include<iostream>
using namespace std;

int szukaj_lidera(int tab[],int n)
{
        int lider = tab[0], do_pary = 1;

        //wykreślanie par o różnych wartościach
        for(int i=1;i<n;i++)
  if(do_pary > 0)
                if(tab[i]==lider)
                        ++do_pary;
                else
                        --do_pary;
        else
        {
                ++do_pary;
                lider = tab[i];
        }
        //koniec wykreślania

        if(do_pary==0)
                return -1; //zwrócenie -1 oznacza, że zbiór nie posiada lidera

        int ile = 0; //zmienna zliczająca wystąpienia potencjalnego lidera

        for(int i=0;i<n;i++)  //zliczamy wystąpienia lidera
                if(tab[i]==lider)
                        ++ile;

        if(ile>n/2) //sprawdzamy, czy potencjalny lider występuje oczekiwaną ilość razy
                return lider;

        return -1;
}

int main()
```

```
{
	int n, *tab, lider;

	cout<<"Ile liczb chcesz wczytać? ";
	cin>>n;

	tab = new int [n];

	for(int i=0;i<n;i++)
		cin>>tab[i];

	lider = szukaj_lidera(tab,n);

	if(lider==-1)
		cout<<"Zbiór nie posiada lidera"<<endl;
	else
		cout<<"Liderem zbioru jest "<<lider<<endl;

	delete [] tab;

	return 0;
}
```
