Curs 12. Plan de curs

0. Un element dintr-un tablou de numere întregi se numeşte ***compus*** dacă poate fi scris ca suma altor două elemente din tablou, distincte între ele. Scrieţi o funcţie care numără câte elemente compuse are un tablou.

Rezolvări:

```
bool esteCompus(int a[], int dim, int k){
	//decide daca a[k] este compus
	//cautam prima descompunere a lui a[k]
	int val=a[k];
	for(int i=0;i<dim; i++)
	{
		if(i==k) continue;
		for(int j=0; j<i;j++)
		{
			if(j==k) continue;
			if( a[i]+a[j]==val) return true;
		}
	}
	return false;
}

int cateCompuse1(int a[], int dim){
	int contor=0;
	for(int k=0;k<dim;k++)
	{
		if(esteCompus(a,dim,k)) contor++;
	}
	return contor;
}

//versiunea fara esteCompus()
int cateCompuse2(int a[], int dim){
	int contor=0;
	for(int k=0;k<dim;k++)
	{
		//stabilim daca a[k] este compus
		//cautam prima descompunere
		int val=a[k];
		bool amGasit=false;
		for(int i=0;i<dim && !amGasit; i++)
		{
			if(i==k) continue;
			for(int j=0; j<i && !amGasit;j++)
			{
				if(j==k) continue;
				if( a[i]+a[j]==val) amGasit=true;
			}
		}
		if(amGasit) contor++;
	}
	return contor;
}
```

Pointeri şi referinţe&

I. Pointeri C/C++, pointeri şi tablouri

1. Operatorul adresă (operatorul &), operatorul ţintă (opreatorul *)

```
int main(){
    int a = 12, b = 120;
    cout << &a << endl; //010FFE8C
    int *p;
    p = &a;
    cout << p << endl;  //010FFE8C
    cout << *p << endl; //12
    p = &b;
    *p = 13;
    cout << b << endl;  //13
    return 0;
}
```

2. Pointeri către structuri. Operatorul săgeată ->

3. Aritmetica pointerilor, pointeri şi tablouri, operatorul de indexare p[i]==*(p+i)

a) Incrementare/decrementare
b) Suma şi diferenţa dintre un pointer şi un întreg.
c) Diferenţa a doi pointeri.
d) Comparaţia a doi pointeri.

Parcurgeri de tablouri

```
const int dim = 10;
int main(){
    int tab[dim];
    for (int i = 0; i < dim; i++) tab[i] = i*i;
    for (int i = 0; i < dim; i++) cout << *(tab + i) << " ";
    cout << endl;                 //0 1 4 9 16 25 36 49 64 81

    int* p = tab;    //<=> p=&tab[0]      // => *p=tab[0];
    for (int i = 0; i < dim; i++) cout << p[i] << " ";
    cout << endl;                 //0 1 4 9 16 25 36 49 64 81

    // tab++; //     error: '++' needs l-value
    // (tab este o constanta de tip "pointer catre int")
    for (int i = 0; i < dim; i++) cout << *p++ << " ";
    cout << endl;                 //0 1 4 9 16 25 36 49 64 81

    for (int *q = tab, *qfin = tab + dim; q < qfin; q++) cout << *q << " ";
    cout << endl;                 //0 1 4 9 16 25 36 49 64 81
    return 0;
}
```

5. Tablouri ca parametri formali

```
double suma(double *tab, int dim){
    //<=>double suma(double tab[ ], int dim){
    //<=>double suma(double tab[5], int dim){
    double suma = 0;
    for (double* tabFinal = tab + dim; tab < tabFinal; tab++) suma += *tab;
    return suma;
}
```

II. Referinţe& C++, transmisie prin referinţe.

1) declarare

```
int alfa = 100;
int& beta = alfa;
beta = 12;
cout << alfa << endl; //12
```

2) utilizare

```
#include<iostream>
using namespace std;
void schimbaPrinValoare(int va, int vb){
    int aux = va;
    va = vb;
    vb = aux;
}
void schimbaPrinPointeri(int* pa, int* pb){
    int aux = *pa;
    *pa = *pb;
    *pb = aux;
}
void schimbaPrinReferinta(int& ra, int& rb){
    int aux = ra;
    ra = rb;
    rb = aux;
    return;
}
int main(){
    int a = 1, b = 10;
    schimbaPrinValoare(a, b);
    cout << "a=" << a << " b=" << b << endl;
    //a=1 b=10;
    a = 11;
    b = 22;
    schimbaPrinPointeri(&a, &b);
    cout << "a=" << a << " b=" << b << endl;
    //a=22 b=11
    a = 11;
    b = 22;
    schimbaPrinReferinta(a, b);
    cout << "a=" << a << " b=" << b << endl;
    //a=22 b=11
    return 0;
}
```