

O'zbekiston oliy va o'rta maxsus ta'lim vazirligi

**Abdulla Qodiriy nomli
Jizzax Davlat Pedagogika Instituti**

Fizika-matematika

Informatika o'qitish metodikasi

KURS' IS'HI

Algoritmlash va dasturlash asoslari

C++ dasturlash tili

Jahongir Do'stqulov

C++ da strukturalar bilan ishlash

JIZZAX-2015

C++ da strukturalar bilan ishlash

REJA:

I. **Kirish**

II. **Asosiy qism. Strukturalar, struktura yaratish**

II.1 Strukturali tiplar va strukturalar

II.2 Struktura tipli o'zgaruvchilarni e'lon qilish

II.3 Strukturalar uchun xotiradan joy ajratish

III. **Strukturalardan foydalanish usullari**

III.1 Strukturalarga murojaat

III.2 Strukturalar va massivlar

III.3 Strukturalar va funksiyalar

IV. **Xulosa**

V. **Foydalanilgan adabiyotlar**

KIRISH

Keyingi yillarda amaliy dasturchilarga juda ko'p integratsion dasturtuzish muhitlaritaklif etilayapti. Buning uchun hitlar uyokibu imkoniyatlar bilan bir-biridan farqqiladi.

Aksariyat dasturlashtirish muhitlarining fundamental asosi C++ tiliga olib qayiladi. C++ tili B. Straustrup tomonidan yaratilgan. Amerika milliy standartlar instituti (American National Standards Institute - ANSI) rahbarligi ostidagi C standartlari akkreditivlash komiteti (Accredited Standards Committee) C++ tilining xalqaro standartini tuzdi. C++ standartini vaqtida ISO - International Standards Organization (Standartlash bo'yicha xalqaro tashkilot) standartideb ham nomlanadi.

Vaqt o'tishi bilan dasturchilar oldiga quyilgan masalalar o'zgarib boryapti. Buning uchun yigirma yil oldin dasturlar kattahajmdagi ma'lumotlarni qayta ishlash uchun tuzilardi. Buning uchun dasturni yozuvchi ham, uning foydalanuvchisi ham kompyuter sohasidagi bilimlar bo'yicha professional bo'lishi talab etilardi. Hozirda esa ko'pgina o'zgarishlar ro'y berdi. Kompyuter bilan ko'proq uning apparat va dasturiy ta'minoti haqida tushunchalar ga ega bo'lgan kishilar ishlashyapti. Kompyuter odamlar tomonidan unichu qurilganish vositasi emas, ko'proq o'zlarining oldilariga qo'yilgan, o'zlarining ishlari ga tegishli bo'lgan muammolarini yechish instrumenti bo'lib qoldi. Dasturlash gatalabni o'zgarishina faqat tillarning o'zgarishiga balki uni yozish texnologiyasini ham o'zgarishiga olib keldi.

Shu vaqtgacha dasturlar berilgan ma'lumotlar ustida biror bira amal bajaruvchi protseduralar ketma-ketligidan iborat edi. Protседuralar yoki funksiyalar ham o'zida aniqlangan ketma-ket bajariluvchi komandalarga o'plamidan iboratdir. Buning uchun berilgan ma'lumotlarga murojaatlar protseduralarga ajratilgan holda amalga oshiriladi.

Strukturaviy dasturlashning asosiy g'oyasi «bo'laklari vahukmronlik qil» prinsipi ga butunlay mos keladi. Kompyuter dasturini masalalar to'plamidan iborat deb qaraymiz. Oddiy tavsiflash uchun murakkab bo'lgan ixtiyoriy masalaning bir nechta nisbatan kichik huroq bo'lgan tarkibiy masalalarga ajratamiz va bo'linishni toki masalalar tushuni shuncha netarlik darajada oddiy bo'lguncha davom ettiramiz. Misol sifatida kompaniya xizmatchilarining o'rtacha ish haqini hisoblashni olamiz. Buning uchun masalasodda emas. Uniqatorqism masalalar ga bo'lamiz:

1. Har bir xizmatchining oylik maoshi qanchaligini aniqlaymiz.
2. Kompaniyaning xodimlar sonini aniqlaymiz.

3. Barchaishhaqlariniyig'amiz.

4. Hosilbo'lganyig'indinikompaniyaxodimlarisonigabo'lamiz.

Xodimlarningoylikmaoshlariyig'indisinihisoblashjarayoninihambirnecha bosqichlargaajratishmumkin.

1. Harbixodimhaqidagiyozuvnio'qiymiz.

2. Ishxaqito'g'risidagima'lumotniolamiz.

3. Ishhaqiqiymatiniyig'indigaqo'shamiz.

4. Keyingixodimhaqidagiyozuvnio'qiymiz.

O'znavbatida,
harbixodimhaqidagiyozuvnio'qishjarayoninihamnisbatankichikroqqismoperatsiyalargaajratishmumkin:

1. Xizmatchifayliniochamiz.

2. Kerakliyozuvgao'tamiz.

3. Ma'lumotlarnidiskdano'qiymiz.

Strukturaviydasturlashmurakkabmasalalarniyechishdayetarlichamuvofaqqiyatliushlubbo'libqoldi.

STRUKTURALI TIPLAR VA STRUKTURALAR

Biz C++ dasturlash tilida qiymat turlari sifatida **int**(butun sonlar), **double**(haqiqiy sonlar), **char**(belgi), **string**(matn), **boolean**(mantiqiy qiymatlar) kabi qator tiplardan foydalanamiz. Bunday tiplar bir-biridan qo'llanilishi va xotiraga saqlashda qay tartibda kodlanishi bilan farq qiladi. Ushbu tiplar ostida e'lon qilingan o'zgaruvchilar faqat bitta qiymat saqlashi mumkin.

Dasturlash jarayonida shunday holatlar yuzaga keladiki, faqat bitta qiymat tipli o'zgaruvchili, va hatto, massiv o'zgaruvchili qiymatlar bilan ishlashda noqulayliklar tug'uladi. Bunday holatlarni qulaylashtirish uchun C++da struktura tushunchasi kiritilgan.

Struktura – bu turli tipdagi ma'lumotlarning biror nom ostida birlashtirilgan, dasturchi tomonidan beriladigan yangi tipdir.

Struktura har xil tipdagi elementlar-komponentalardan iborat bo'ladi. Strukturalarni hosil qilish quyidagicha amalga oshiriladi:

```
struct <strukturali_tip_nomi>
{
    <1_qiymat_turi><1_qiymat_nomi>;
    <2_qiymat_turi><2_qiymat_nomi>;
    ...
    <n_qiymat_turi><n_qiymat_nomi>;
};
```

Strukturali tip hosil qilish

Strukturali tiplar asosan ma'lum bir obyekt va guruhlar uchun tuziladi.

Misol uchun uchburchak qiymatlarini saqlash uchun struktura tuzaylik. Avvalo, uchburchak qanday qiymatlarga ega ekanligini ko'rib chiqaylik.

- **a, b, c** tomonlar;
- **S** yuza;
- **h** balandlik;
- **tur** Uchburchak turi;

Bu struktura dasturda quyidagicha ta'riflanadi:

```
struct uchburchak
{
    float a,b,c;
    double S;
    float h;
    char tur[25];
};
```

Strukturalarni tasvirlashda ixtiyoriy murakkab tip uchun nom berishga imkon beruvchi **typedef** hizmatchi so'zidan foydalanish mumkin. Bu holda strukturali tip quyidagi shaklda ta'riflanadi:

```
typedef struct
{
    <1_qiymat_turi><1_qiymat_nomi>;
    <2_qiymat_turi><2_qiymat_nomi>;
    ...
    <n_qiymat_turi><n_qiymat_nomi>;
}
<strukturali_tip_nomi>;
```

typedef orqali murakkab tip hosil qilish

Misol uchun:

```
typedef struct
{
    double real;
    double imag;
}
complex;
```

Bu misolda kompleks sonni tasvirlovchi strukturali tip **complex** kiritilgan bo'lib, kompleks son haqiqiy qismini tasvirlovchi real va mavhum qismini tasvirlovchi komponentalaridan iboratdir. Konkret strukturalar bu holda quyidagicha tasvirlanadi:

```
complex sigma, alfa;
```

Strukturali tip **typedef** yordamida aniqlangan nomdan tashqari, standart usulda aniqlangan nomga ega bo'lishi mumkin. Quyidagi misolda kasr sonni tasvirlovchi numerator –sur'at va denominator-mahraj komponentalaridan iborat struktura ta'rifi keltirilgan.

```
typedef struct rational_fraction
{
    int numerator;
    int denominator;
}
fraction;
```

Bu misolda `fraction` kasrning `typedef` orqali kiritilgan nomi, `rational_fraction` standart usulda kiritilgan nom. Bu holda konkret strukturalar quyidagicha tasvirlanishi mumkin:

```
struct rational_fraction alfa; fraction beta;
```

STRUKTURA TIPLI O'ZGARUVCHILARNI E'LON QILISH

Yuqoridagi misollarda konkret strukturalarni ta'riflashni ikki usuli ko'rib chiqilgan. Agar strukturali tip standart usulda kiritilgan bo'lsa konkret strukturalar quyidagi shaklda ta'riflanadi:

```
struct <struktura nomi><konkret strukturalar ruyhati>
```

Struktura tipli o'zgaruvchilarni e'lon qilish

Masalan:

```
struct goods food;
```

Agar strukturali tip `typedef` hizmatchi so'zi yordamida kiritilgan bo'lsa, konkret strukturalar quyidagi shaklda ta'riflanadi:

```
<struktura nomi><konkret strukturalar ruyhati>
```

`typedef` orqali hosil qilingan struktura
tipli o'zgaruvchini e'lon qilish

Masalan:

```
complex sigma;
```

Bu usullardan tashqari konkret strukturalarni ta'riflashning boshqa usullari ham mavjuddir. Strukturalar ta'riflanganda konkret strukturalar ro'yhatini kiritish mumkin:

```
struct<strukturali_tip_nomi>
{
    <1_qiymat_turi><1_qiymat_nomi>;
    <2_qiymat_turi><2_qiymat_nomi>;
    ...
    <n_qiymat_turi><n_qiymat_nomi>;
}
Konkret_strukturalar_ruyhati;
```

Konkret strukturalar hosil qilish

Misol:

```
struct student
{
    char name[15];
    char surname[20];
    int year;
}
student_1, student_2, student_3;
```

Bu holda student strukturali tip bilan birga uchta konkret struktura kiritiladi. Bu strukturalar student ismi (name[15]), familiyasi (surname[20]), tug'ilgan yilidan (year) iborat.

Strukturali tip ta'riflanganda tip nomi ko'rsatilmay, konkret strukturalar ro'yhati ko'rsatilishi mumkin:

```
struct
{
    <1_qiymat_turi><1_qiymat_nomi>;
    <2_qiymat_turi><2_qiymat_nomi>;
    ...
    <n_qiymat_turi><n_qiymat_nomi>;
}
```

Konkret_strukturalar_ruyhati;

Chekli o'zgaruvchilar uchun struktura hosil qilish

Quyidagi ta'rif yordamida uchta konkret struktura kiritiladi, lekin strukturali tip kiritilmaydi.

```
struct
{
    char processor [10];
    int frequency;
    int memory;
    int disk;
}
IBM_486, IBM_386, Compaq;
```

STRUKTURALAR UCHUN XOTIRADAN JOY AJRATISH

Strukturali tip kiritilishi bu tip uchun xotiradan joy ajratilishiga olib kelmaydi. Har bir konkret struktura (ob'ekt) ta'riflanganda, shu ob'ekt uchun elementlar tiplariga qarab xotiradan joy ajratiladi. xotiradan joy zich ajratilganda struktura uchun ajratilgan joy hajmi har bir element uchun zarur bo'lgan xotira hajmlari yig'indisiga teng bo'ladi. Shu bilan birga xotiradan joy zich ajratilmasligi ham mumkin ya'ni elementlar orasida bo'sh joylar ham qolishi mumkin. Bu bo'sh joy keyingi elementni xotira qismlarining qabul qilingan chegaralari bo'yicha tekislash uchun qoldiriladi. Misol uchun butun tipdagi elementlar juft adreslardan boshlansa, bu elementlarga murojaat tezroq amalga oshiriladi.

Konkret strukturalarni joylashuviga ba'zi kompilyatorlarda **#pragma**preprotsessori direktivasi yordamida ta'sir o'tkazish mumkin. Bu direktivadan quyidagi shakldafoydalanish mumkin:

```
#pragma pack(n)
```

Strukturalarning xotirada joylashishi

Bu yerda n qiymati 1,2 eki 4 ga teng bo'lishi mumkin.

Pack(1) – elementlarni bayt chegaralari bo'yicha tekislash;

Pack(2) – elementlarni so'zlar chegaralariga qarab tekislash;

Pack(4) – elementlarni ikkilangan yuzlar chegaralariga qarab tekislash.

Struktura uchun ajratilgan joy hajmini quyidagi amallar yordamida aniqlash mumkin:

Sizeof (strukturali_tip_nomi);

Sizeof (struktura_nomi);

Sizeof struktura_nomi;

Struktura egallagan joyni aniqlashsh

Oxirgi holda struktura nomi ifoda deb qaraladi. Ifodaning tipi aniqlanib, hajmi hisoblanadi.

Misol uchun:

Sizeof (struct goods);

Sizeof (tea);

Sizeof uchburchak;

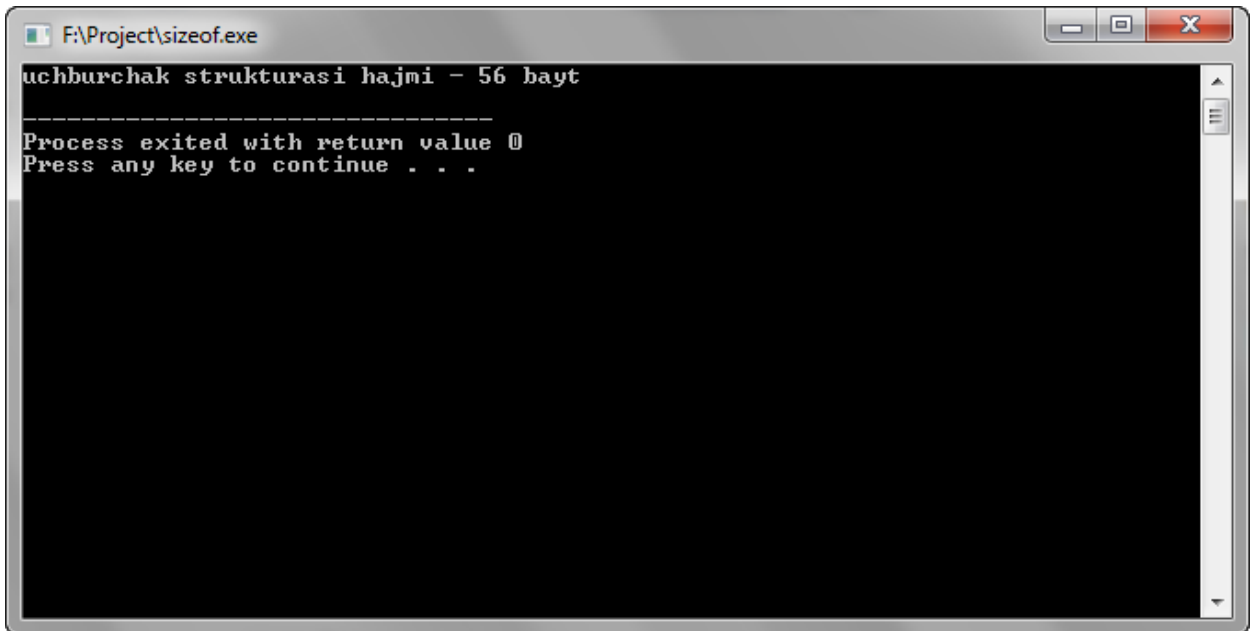
Yuqoridagi **uchburchak** nomli struktura hajmini aniqlab ko'raylik:

```
#include <iostream>
using namespace std;

struct uchburchak
{
    float a, b, c;
    double s;
    float h;
    char tur[25];
};

int main()
{
    cout<<"uchburchak strukturasi hajmi - " <<sizeof(uchburchak)<<"
    bayt" <<endl;
    return 0;
}
```

Dastur kodi



```
F:\Project\sizeof.exe
uchburchak strukturasi hajmi - 56 bayt
-----
Process exited with return value 0
Press any key to continue . . .
```

Dastur ko'rinishi

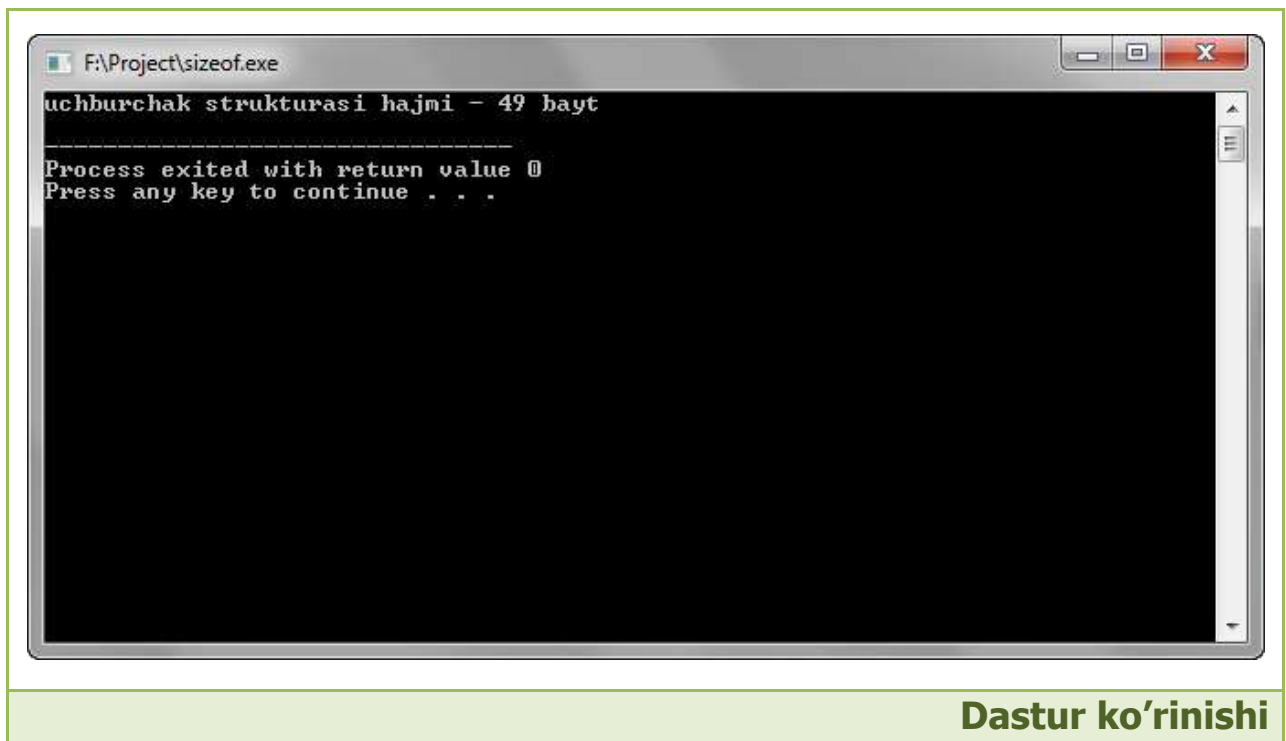
Demak, **uchburchak** nomli struktura xotiradan 56 bayt joy egallamoqda. Bu yerda struktura tarkibiy elementlari xotirada zich joylashmagan. Elementlarni bayt bo'yicha zichlash uchun **#pragma pack(1)** kiritilishi yetarli:

```
#include <iostream>
#pragma pack(1)
using namespace std;

struct uchburchak
{
    float a,b,c;
    double s;
    float h;
    char tur[25];
};

int main()
{
    cout<< "uchburchak strukturasi hajmi - " << sizeof(uchburchak) << "
bayt" << endl;
    return 0;
}
```

Dastur kodi



Dastur ko'rinishi

Bu yerda xotira har bir tip kattaligi bo'yicha ajratiladi:

$4 * \text{sizeof}(\text{float}) + \text{sizeof}(\text{double}) + 25 * \text{sizeof}(\text{char}) = 4 * 4\text{bayt} + 8\text{bayt} + 25 * 1\text{bayt} = 49\text{ bayt}.$

STRUKTURALARGA MUROJAAT

Konkret strukturalar ta'riflanganda massivlar kabi initsializatsiya qilinishi mumkin. Masalan:

```
complex sigma {1.3;12.6};  
Struct uchburchak  
t_uchburchak={3.0,4.0,5.0,6.0,2.4,"to`g`ri uchburchak"};
```

Bir hil tipdagi strukturalarga qiymat berish amalini qo'llash mumkin:

```
complex alfa; alfa=sigma;
```

Lekin strukturalar uchun solishtirish amallari aniqlanmagan.

Strukturalar elementlariga quyidagicha murojaat qilish mumkin:

`<struktura_nomi>.<element_nomi>`

Struktura elementiga murojaat

"."(nuqta) – amali struktura elementiga murojaat qilish amali deyiladi. Bu amal qavs amallari bilan birga eng yuqori ustivorlikka egadir. Masalan:

```
Complex alfa={1.2,-4.5},betta={5.6,-7.8},sigma;  
Sigma.real=alfa.real+betta.real;  
Sigma.imag=alfa.imag+betta.imag;
```

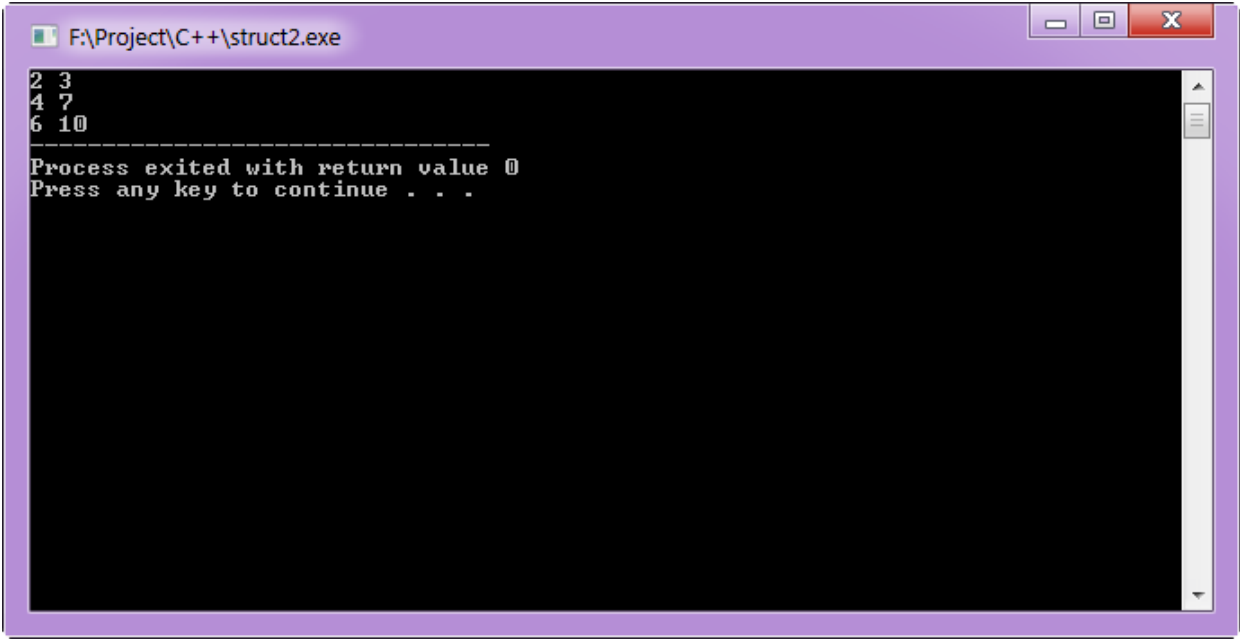
Konkret strukturalar elementlari dasturda alohida kiritilishi va chiqarilishi zarurdir.

Misol: Ikki kompleks son qiymatlari berilganda yig'indisini hisoblash dasturi tuzilsin.

Ikki kompleks sonning haqiqiy va mavhum qismlarini alohida-alohida kiritib olamiz va haqiqiy qismlar yig'indisi va mavhum qismlar yig'indisini alohida kompleks o'zgaruvchiga o'zlashtiramiz:

```
#include <iostream>  
using namespace std;  
  
struct complex {  
    int real;  
    int imag;  
};  
  
int main()  
{  
    complex x,y,z;  
    cin>>x.real>>x.imag;  
    cin>>y.real>>y.imag;  
    z.real=x.real+y.real;  
    z.imag=x.imag+y.imag;  
    cout<<z.real<<" "<<z.imag;  
  
    return 0;  
}
```

Dastur kodi



```
F:\Project\C++\struct2.exe
2 3
4 7
6 10
-----
Process exited with return value 0
Press any key to continue . . .
```

Dastur ko'rinishi

STRUKTURALAR VA MASSIVLAR

Massivlar strukturalar elementlari sifatida. Massivlarni strukturalar elementi sifatida ishlatilishi hech qanday qiyinchilik tug'dirmaydi. Quyidagi misol yordamida struktura elementlari sifatida massivlarni qo'llashni ko'rib chiqaylik.

Misol: Sakkizburchakning berilgan sakkizta tomoniga ko'ra perimetrini hisoblovchi dastur tuzilsin.

O'zgaruvchilar sifatida perimetr va sakkizta tomonni olamiz. Tomonlarni alohida-alohida o'zgaruvchilar bilan ifodalash noqulaylik tug'dirgani uchun tomonlarning qiymatlarini bitta massivga saqlaymiz. Dasturimiz quyidagicha bo'ladi:

```

#include <iostream>
using namespace std;

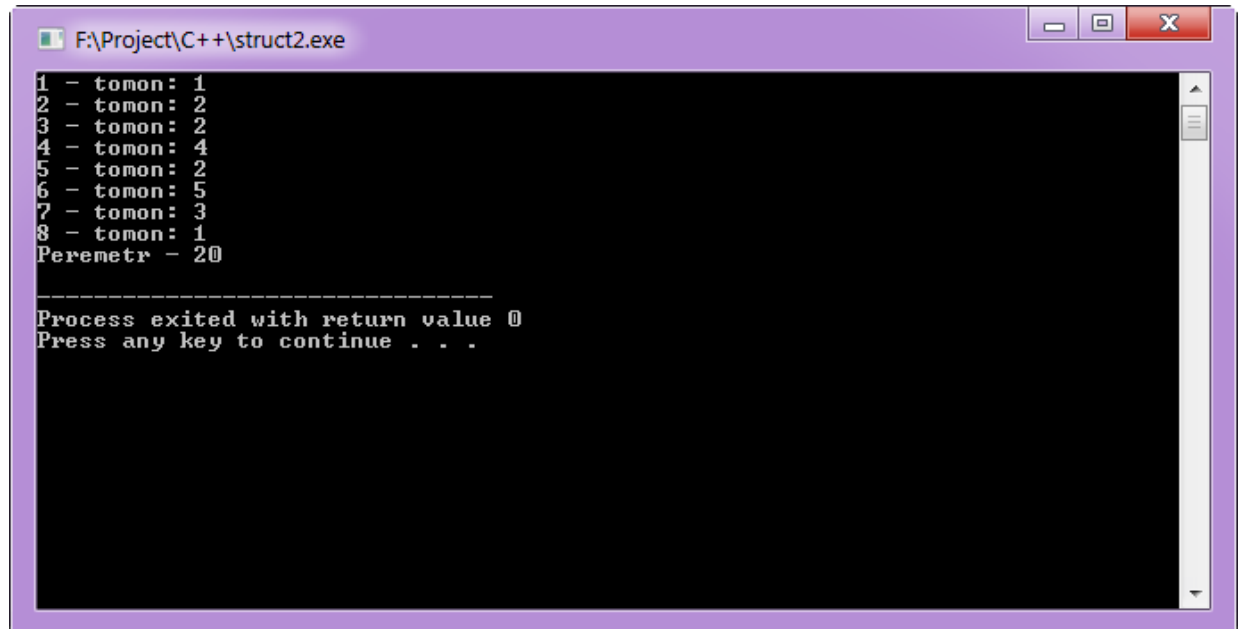
struct sakkizburchak
{
    double perimetr=0;
    double tomonlar[8];
};

int main()
{
    sakkizburchak x;
    for(int i=1; i<=8; i++)
    {
        cin>>x.tomonlar[i];
    }
    for(int i=1; i<=8; i++)
    {
        x.perimetr += x.tomonlar[i];
    }
    cout<<"Perimetr - "<<x.perimetr<<endl;

    return 0;
}

```

Dastur kodi



```

F:\Project\C++\struct2.exe
1 - tomon: 1
2 - tomon: 2
3 - tomon: 2
4 - tomon: 4
5 - tomon: 2
6 - tomon: 5
7 - tomon: 3
8 - tomon: 1
Perimetr - 20

-----
Process exited with return value 0
Press any key to continue . . .

```

Dastur ko'rinishi

Strukturalar massivlari.Strukturalar massivlari oddiy massivlar kabi tasvirlanadi. Yuqorida kiritilgan strukturali tiplar asosida quyidagi strukturalar massivlarini kiritish mumkin:

```
struct goods list[100];  
sakkizburchaktuplam[80];
```

Bu ta'riflarda `list` va `tuplam` strukturalar nomlari emas, elementlari strukturalardan iborat massivlar nomlaridir. Konkret strukturalar nomlari bo'lib `tuplam[0]`, `tuplam[1]` va hokazolar xizmat qiladi. Konkret strukturalar elementlariga quyidagicha murojaat qilinadi:

```
tuplam[i].perimetr
```

```
tuplami-strukturasiningperimetr maydoniga murojaat
```

Misol:Noutbook do'konlari uchun qo'shimcha servis dasturi. Bunda xaridor mo'ljalidagi summa kiritiladi va shu summaga yetarli bo'lgan noutbooklar ro'yhati qo'shimcha ma'lumotlari bilan birgalikda beriladi.

Avvalo, do'kondagi noutbooklar bazasi shakllantirilishi, strukturali massiv sifatida har bir noutbook dasturga kiritilishi lozim.

Dastur umumiy kodi quyidagicha:

```
#include <iostream>  
using namespace std;  
  
struct noutbook  
{  
    string firma;           //ishlab chiqqan firma  
    int op_xotira;          //operativ xotira hajmi (MBayt)  
    int hdd;                //qattiq disk hajmi (GBayt)  
    string prot_nomi;       //microprotsessors nomi  
    float prot_gers;        //microprotsessors chastotasi (GHz)  
    string sana;            //sotuvdagi sanasi  
    int sum;                //Narxi (so'm)  
};  
  
int main()  
{  
    int summa, tmp=0, n;  
    noutbook tuplam[]=  
    {  
        "Asus", 2048, 500, "Intel Celeron", 1.8, "2015.02.12", 1200000,  

```

```

        "Lenovo", 4096, 500, "AMD", 1.5, "2015.03.25", 1300000,
        "HP", 4096, 1000, "Intel Core i5", 1.8, "2015.01.03", 1600000,
        "Samsung", 2048, 320, "Intel Celeron", 1.1, "2014.08.18", 1000000
    };

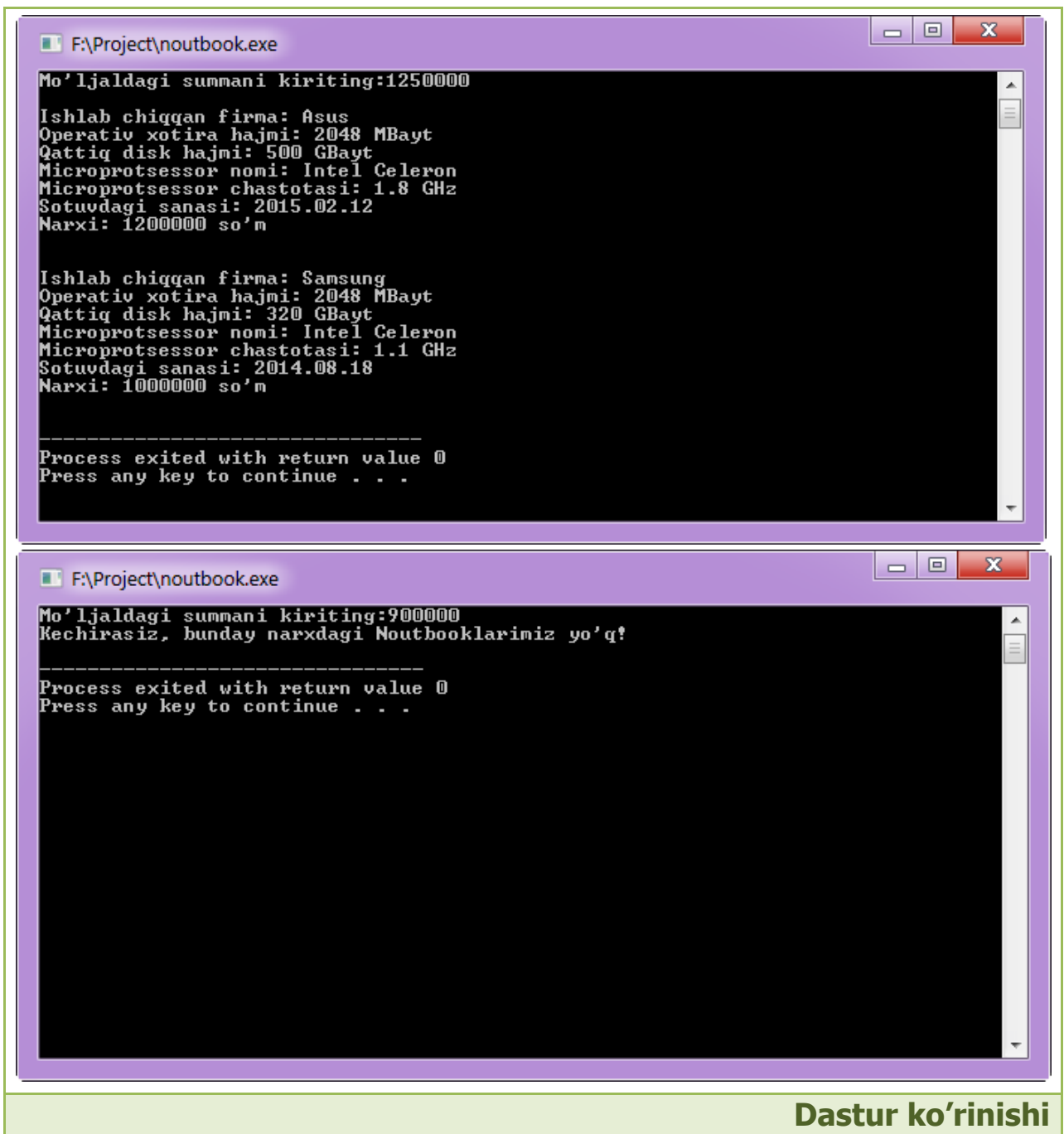
    cout<<"Mo'ljaldagi summani kiriting:";
    cin>>summa;

    n =sizeof(tuplam)/sizeof(tuplam[0]);
    for(int i=0; i<n; i++)
    {
        if( tuplam[i].sum <= summa ){
            cout<<endl;
            cout <<"Ishlab chiqqan firma: " <<tuplam[i].firma <<endl;
            cout <<"Operativ xotira hajmi: " <<tuplam[i].op_xotira <<
" MBayt"<<endl;
            cout<<"Qattiq disk hajmi: " <<tuplam[i].hdd<<
"GBayt"<<endl;
            cout<<"Microprotssessor nomi: " <<tuplam[i].prot_nomi<<
endl;
            cout<<"Microprotssessor chastotasi: " <<
tuplam[i].prot_gers<<" GHz"<<endl;
            cout<<"Sotuvdagi sanasi: " <<tuplam[i].sana<<endl;
            cout<<"Narxi: " <<tuplam[i].sum<<" so'm"<<endl;
            cout<<endl;
            tmp =1;
        }
    }
    if(tmp ==0) cout <<"Kechirasiz, bunday narxdagi Noutbooklarimiz
yo'q!"<< endl;

    return0;
}

```

Dastur kodi



Yuqorida strukturali massiv elementlari sonini aniqlash quyidagicha amalga oshirilmoqda:

```
n = sizeof(tuplam) / sizeof(tuplam[0]);
```

Bu yerda to'plam elementlari soni umumiy to'plam xotiradan egallagan hajmi bitta to'plam elementi egallagan hajmga bo'lib topilmoqda.

STRUKTURALAR VA FUNKSIYALAR

Strukturalar funktsiyalar argumentlari sifatida yoki funksiya qaytaruvchi qiymati sifatida kelishi mumkin. Bunda funksiya tarkibida struktura elementlariga yuqoridagi kabi murojaat qilinadi. Struktura tipli qiymat funksiya argumenti, yoki funksiya qaytaruvchi qiymat sifatida kelganda uning barcha elementlari keltiriladi.

Misol: Ikkita vektorning kordinatalari berilganda ularning skalyar ko'paytmasini hisoblovchi funksiya tuzilsin.

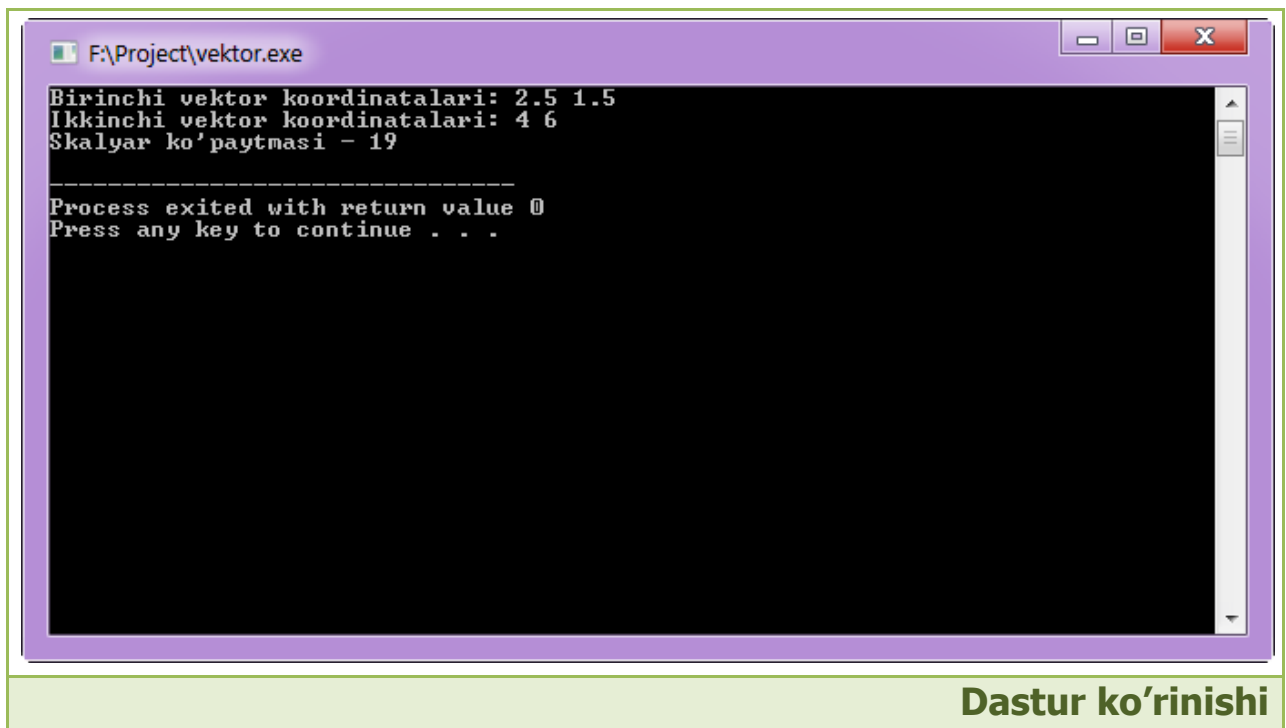
```
#include <iostream>
using namespace std;

struct vektor
{
    double x,y;
};

double skalyar(vektor a, vektor b)
{
    return a.x * b.x + a.y * b.y;
}

int main()
{
    vektor a,b;
    cout <<"Birinchii vektor koordinatalari:";
    cin >> a.x >> a.y;
    cout <<"Ikkinchi vektor koordinatalari:";
    cin >> b.x >> b.y;
    cout <<"Skalyar ko'paytmasi - "<< skalyar(a,b)<< endl;
    return 0;
}
```

Dastur kodi



```
F:\Project\vektor.exe
Birinchii vektor koordinatalari: 2.5 1.5
Ikkinchi vektor koordinatalari: 4 6
Skalyar ko'paytmasi - 19

-----
Process exited with return value 0
Press any key to continue . . .
```

Dastur ko'rinishi

Strukturalio'zgaruvchifunksiyaqaytaruvchiqiymatisifatidafoydalanilgand ahamhuddishundayishlatiladi. Ammo, funksiyaqiymatio'zlashtirilayotganqiymataynanfunksiyaqaytarayotganstruktura gategishlibo'lishikerak.

Misol: a va b vektorlar, hamda n va m haqiqiy sonlari berilgan. $n \cdot a + m \cdot b$ vektor koordinatalarini hisoblash dasturi tuzilsin.

Funksiya ichida qiymatlarni saqlash uchun, hamda bu funksiya qaytargan qiymatni o'zlashtirib olish uchun ikkita qo'shimcha strukturali qiymatdan foydalanib quyidagicha dastur tuzamiz:

```

#include <iostream>
using namespace std;

struct vektor
{
    double x,y;
};

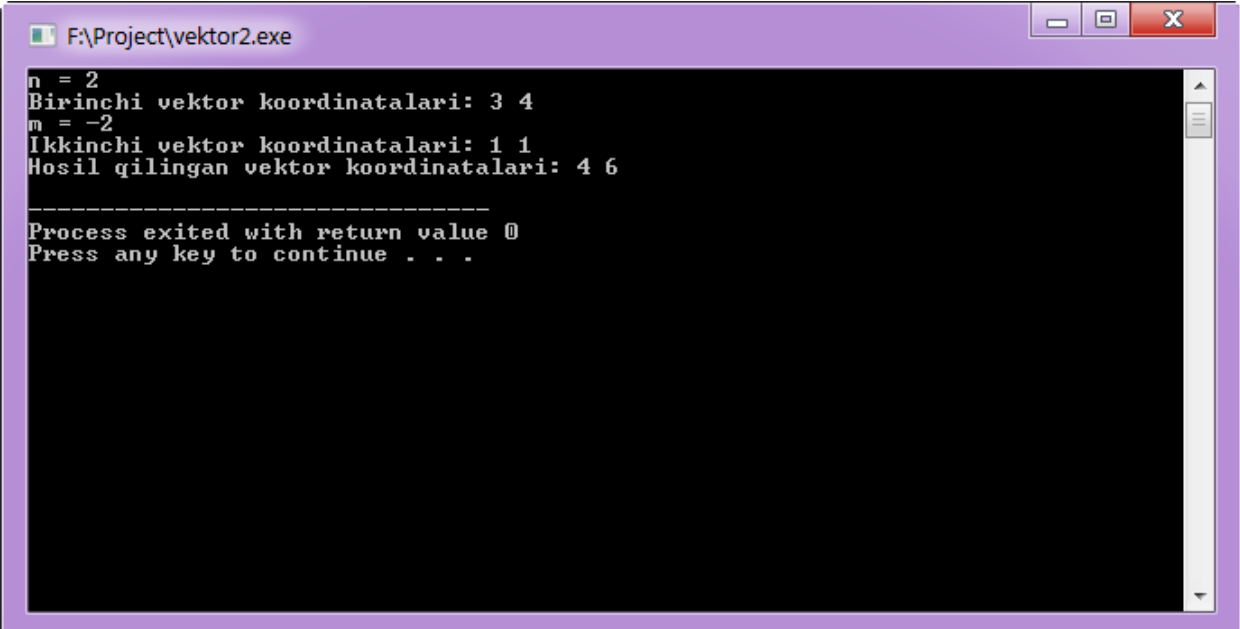
vektor yangi_v(double n, vektor a, double m, vektor b)
{
    vektor c;
    c.x = n * a.x + m * b.x;
    c.y = n * a.y + m * b.y;
    return c;
}

int main()
{
    vektor vektor1,a,b;
    double n,m;
    cout <<"n = ";
    cin >> n;
    cout <<"Birinchi vektor koordinatalari:";
    cin >> a.x >> a.y;
    cout <<"m = ";
    cin >> m;
    cout <<"Ikkinchi vektor koordinatalari:";
    cin >> b.x >> b.y;
    vektor1 = yangi_v(n,a,m,b);
    cout <<"Hosil qilingan vektor koordinatalari: ";
    cout << vektor1.x <<" "<< vektor1.y << endl;

    return 0;
}

```

Dastur kodi



```
F:\Project\vektor2.exe
n = 2
Birinchii vektor koordinatalari: 3 4
m = -2
Ikkinchi vektor koordinatalari: 1 1
Hosil qilingan vektor koordinatalari: 4 6

-----
Process exited with return value 0
Press any key to continue . . .
```

Dastur ko'rinishi

XULOSA

Demak, strukturalar dasturchilar tomonidan kiritilib qo'llaniladigan, C++ dasturlash tilining asosiy tiplaridan tuzilgan va o'zida massivlarni ham mujassam qila oladigan, biror nomga ega murakkab tipdir. Strukturalar asosiy tiplarning deyarli barcha xossalarini qo'llay oladi. Strukturalardan massivlarda, ko'rsatkichlarda va funksiyalarda ham qo'llash imkoniyati mavjud.

Strukturalar obyektga yo'naltirilgan dasturlashning asosidir. Biroq u obyektning funksiyalarini emas, balki faqat qiymatlarini o'zida saqlaydi. Strukturalar bilan ishlashni yaxshi bilgan dasturchi, albatta obyektga yo'naltirilgan dasturlashni ham yaxshi tushuna oladi – deb o'ylayman.

Men bu kurs ishimda yangi strukturalar yaratishni va xotiradan egallanuvchi joyni to'g'ri tashkil etishni, strukturadan massiv sifatida va funksiyalarda foydalanish jarayonlarini o'rganib, turli misollar yordamida yoritib chiqdim. Albatta, bundan olgan bilimlarim dasturlash olamida yana bir pog'ona ko'tarilganimni tasdiqlaydi.

FOYDALANILGAN ADABIYOTLAR

1. SH. A. Nazirov., R. V. Qobulov.
«Ob'ektagayonaltirilgandasturlashtillari»fanidano'quvqo'llanma. /TUITToshkent – 2011.
2. Стивен Прата **«Язык программирования С++»** - лекции и упражнения. /
Москва:Санкт-Петербург:Киев – 2012.
3. SayfiyevJ.F. **«C++ tiligakirish»** – uslubiyqo'llanma. /Buxoro – 2005.
4. <http://dastur.uz>
5. <http://wikipedia.org>
6. <http://google.uz>