

# 1 Бүтэц

Логикоор холбогдсон өгөгдлүүд бүтэц гэж нэрлэгдэх нэг өгөгдлийн нэгж (төрөл) болж хэрэглэгдэж болно. Жишээ нь: Оюутныг нэр, овог, хаяг, голч гэсэн мэдээллүүдээр тодорхойлон оюутан гэсэн бүтэц төрөл үүсгэж болно.

```
struct Student {  
    char firstname[20]; // Оюутны нэрийг хадгалах хүснэгт  
    char lastname[20]; // Оюутны овгийг хадгалах хүснэгт  
    char address[50]; // Оюутны гэрийн хаягийг хадгалах хүснэгт  
    float mark; // Оюутны дүнгийн голчийг хадгалах хүснэгт  
};
```

## 1.1 Бүтэц зарлах

Дүрэм:

```
struct    {  
    //  
};
```

Тухайн бүтцийг хэдэн ч өгөгдөл агуулж болдог байхаар *struct* түлхүүр үгийг ашиглан багц командын хаалтан дотор өгөгдлүүдээ зарлаж өгнө.

Бүтцийг зарлахдаа доорх бичиглэлийн алийг нь ч хэрэглэж болно.

1.

```
struct A {  
    // ...  
};
```

2.

```
struct {  
    // ...  
} A;
```

## 1.2 Бүтэц төрлийн хувьсагч

Бүтэц төрлийн хувьсагчийг зарлахдаа бүтцийн нэрийн өмнө *struct* түлхүүр үгийг хэрэглэх эсвэл бүтцийг тодорхойлж байхдаа хувьсагчуудыг зарлаж болно.

Бүтцийн нэрийн өмнө, бүтэц гэдгийг ялгаж өгөхийн тулд заавал *struct* түлхүүр үгийг хэрэглэдэг.

А бүтцийн *v1*, *v2*, *a*, *b*, *c* хувьсагчуудыг зарлая.

```
struct A {  
    // ...  
} v1, v2; // Бүтцийг тодорхойлох үед бүтцэн төрлийн  
           // хувьсагч зарлах  
// ...  
struct A a, b, c; // Бүтцэн төрлийн хувьсагчдыг зарлах
```

Бүтцийн доторх өгөгдлүүдэд зөвхөн бүтэц төрлийн хувьсагчаар дамжуулан хандана. Үнэндээ бүтэц төрлийн хувьсагч зарлагдах үед л тухайн бүтцэн хувьсагчид харгалзах бүтцийн гишүүд шинээр үүсдэг. Жишээ нь доор *a* сурагч зарлагдах үед л *a*-ийн *name*, *dun* гэсэн хувьсагчууд үүснэ.

```
struct Student {  
    char name[20];  
    int dun;  
};  
// ...  
struct Student a;
```

### 1.2.1 Бүтцэн төрөл

*typedef* түлхүүр үгийн тусламжтайгаар бүтэц төрөл үүсгэж болно. Ингэснээр тухайн төрлийн нэрийн өмнө *struct* түлхүүр үгийг хэрэглэх шаардлагагүйгээр хувьсагчид зарлаж болно.

```
typedef struct {  
    // ...  
} A;
```

ЭСВЭЛ

```
struct A {  
};  
typedef struct A A;
```

Дээрх кодуудын үр дүнд *A* гэсэн төрөл үүсэх тул *A* төрлийн хувьсагчдыг доорх байдлаар зарлаж болно.

```
A v1, v2;
```

## 1.3 Бүтцийн гишүүдэд хандах

Бүтцэд тодорхойлогдсон өгөгдлүүдрүү хандахдаа цэг (.) операторыг ашиглана. *bat* гэсэн *struct Student* төрлийн хувьсагчийн доторх *mark*, *name* гэсэн өгөгдлүүдэд хадъя.

```
struct Student bat;  
bat.mark = 100;  
strcpy(bat.name, "Bat");
```

Харин бүтэц төрлийн хаягийг хадгалах хаяган хувьсагчийн хувьд доорх байдлаар гишүүдрүү нь хандаж болдог.

```
struct Student bat;  
struct Student *p = &bat;  
p->mark = 100;  
strcpy(p->name, "Bat");
```

## 2 Дасгалууд

### 2.1 Ангид

1. Хоёр гурвалжны мэдээллийг доорх бүтцэд уншаад Героны томьёогоор талбайг олж аль том гурвалжныг ол.

```
struct Triangle {
    int a, b, c; // a, b, c нь гурвалжны 3 талын урт
};
```

Талбайг олохдоо дараах томьёогоор олно.

$$p = \frac{a + b + c}{2}, s = \sqrt{p \cdot (p - a) \cdot (p - b) \cdot (p - c)}$$

2. Доорх бүтцэд хоёр цэгийн мэдээллийг хадгалан хоёр цэгийн хоорондох зайг олох функцийг хэрэгжүүл.

```
struct Point {
    int x, y;
};
double distance(Point a, Point b);
```

$(x_1, y_1), (x_2, y_2)$  цэгийн хоорондох зайг дараах томьёогоор олдог.

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

3. 1-р дасгалыг хийх доорх функцийг хэрэгжүүл. Хоёр гурвалжны гурван талыг хэрэглэгчээс уншин, аль их талбайтай гурвалжин болохыг ол.

```
struct Triangle {
    int a, b, c;
};
// Гурвалжны талбайгоо олох функц
double find_area(Triangle g)
{
    // ....
}
```

4. *Rational* бүтэц нь энгийн бутархайн хүртвэр хуваарийг хадгалах бол тэдгээрийн хооронд үйлдэл хийх дараах функцүүдийг хэрэгжүүл.

Энгийн бутархайг хураахдаа хүртвэр хувиар хоёрыг ХИЕХ-д нь хувааж өгнө.

```
struct Rational {
    int d, n; // d/n гэсэн энгийн бутархай
};
typedef struct Rational Rational;
Rational add(Rational a, Rational b); // a, b энгийн бутархайг хооронд нь нэмэх
Rational sub(Rational a, Rational b); // a, b энгийн бутархайг хооронд нь хасах
Rational mult(Rational a, Rational b); // a, b энгийн бутархайг хооронд нь үржих
Rational div(Rational a, Rational b); // a, b энгийн бутархайг хооронд нь хуваах
Rational simplify(Rational a); // a энгийн бутархайг хураах
void print(Rational a); // a энгийн бутархайг хэвлэх
```

## 2.2 Гэрт

1. *Student* бүтцийг хэрэгжүүлж, дараах хайлтын болон голчоор эрэмбэлэх функцүүдийг хэрэгжүүл. *main()* функц доторхыг өөрчлөхгүй.

```

struct Student {
    char fname[20], lname[20], id[10];
    float golch;
};
typedef struct Student Student;
// n оюутны мэдээллийг гараас авч a хүснэгтэд хадгална.
void read_students(Student a[], int n);
// a хүснэгтэд хадгалсан n оюутны мэдээллийг хэвлэнэ.
void print_students(Student a[], int n);
// st оюутны мэдээллийг хэвлэнэ.
void print(Student st);
// a хүснэгтэд хадгалсан n оюутны мэдээллээс fname нэрийг хайна.
// Уг нэртэй оюутан олдож байгаа бол хүснэгтийн дугаарыг үгүй бол -1 утгыг буцаана.
int search_by_fname(Student a[], int n, char fname[]);
// a хүснэгтэд хадгалсан n оюутны мэдээллээс lname овгийг хайна.
// Уг овогтой оюутан олдож байгаа бол хүснэгтийн дугаарыг үгүй бол -1 утгыг буцаана.
int search_by_lname(Student a[], int n, char lname[]);
// a хүснэгтэд хадгалсан n оюутны мэдээллээс id id-г хайна.
// Уг id-тай оюутан олдож байгаа бол хүснэгтийн дугаарыг үгүй бол -1 утгыг буцаана.
int search_by_id(Student a[], int n, char id[]);
// a хүснэгтэд хадгалсан n оюутны мэдээллээс golch голчийг хайна.
// Уг голчтой оюутан олдож байгаа бол хүснэгтийн дугаарыг үгүй бол -1 утгыг буцаана.
int search_by_golch(Student a[], int n, float golch);
// a хүснэгтэд хадгалсан n оюутны мэдээллээс голчоор нь эрэмбэлнэ.
void sort_by_golch(Student a[], int n);

int main()
{
    Student a[100];
    int n, cmd, idx;
    printf("Oyutnii too: ");
    scanf("%d", &n);
    read_students(a, n);
    print_students(a, n);
    char fname[20], lname[20], id[20];
    float golch;
    while (1) {
        printf("1: Nereer xaix, 2: Ovgoor xaix, 3: ID-aar xaix, 4: Golchoor xaix, 5: Golchoor
            erembelex, 6: Xevlex, 7: Garax\n");
        scanf("%d", &cmd);
        if (cmd == 1) {
            printf("Xaix ner: ");
            scanf("%s", fname);
            idx = search_by_fname(a, n, fname);
            if (idx == -1)
                printf("Oyutan oldsongui\n");
            else
                print(a[idx]);
        } else if (cmd == 2) {
            printf("Xaix ovog: ");
            scanf("%s", lname);
            idx = search_by_lname(a, n, lname);
            if (idx == -1)
                printf("Oyutan oldsongui\n");
            else
                print(a[idx]);
        } else if (cmd == 3) {
            printf("Xaix id: ");
            scanf("%s", id);
            idx = search_by_id(a, n, id);
            if (idx == -1)
                printf("Oyutan oldsongui\n");
        }
    }
}

```

```
        else
            print(a[idx]);
    } else if (cmd == 4) {
        printf("Xaix golch: ");
        scanf("%f", &golch);
        idx = search_by_golch(a, n, golch);
        if (idx == -1)
            printf("Oyutan oldsongui\n");
        else
            print(a[idx]);

    } else if (cmd == 5)
        sort_by_golch(a, n);
    else if (cmd == 6)
        print_students(a, n);
    else
        break;
}
return 0;
}

void print(Student st)
{
    printf("Ner: %s\n", st.fname);
    printf("Ovog: %s\n", st.lname);
    printf("Id: %s\n", st.id);
    printf("Golch: %lf\n", st.golch);
}
```