Лабораторі 4:

1. Байгуулагч функц гэж юу вэ?

Нэр нь классын нэртэй ижил байх гишүүн функц ба энэ функц нь ямар ч утга буцаадаггүй. Мөн байгуулагчийг дахин тодорхойлж болдог. Обьектод ой бэлдэж түүний гишүүн өгөгдөлд гарааны утга оноож өгөхийн тулд байгуулагчийг дууддаг.

2. Устгагч функц гэж юу вэ?

Устгагч функц нь цаашдаа хэрэглэгдэхгүй обьект устахад түүний эзэмшиж байсан санах ойг чөлөөлнө. Функцын нэр классынхаа нэртэй ижил байна. Зөвхөн өмнөө “~” тэмдэгтэй бичигддэгээрээ ялгаатай юм. Обьект байгуулагдсан дарааллынхаа эсрэгээр устана.

3. Функц дахин тодорхойлох гэж юу вэ? Жишээгээр тайлбарал.

Ижил нэртэй, нэгэн төрлийн боловч авах аргументийн тоо, төрөл, дараалал нь ялгаатай байх функц тодорхойлохыг функц дахин тодорхойлох гэнэ.

```cpp
int sum(int a, int b) {
    return a + b;
}
int sum(int a, int b, int c) {
    return a + b + c;
}

int main() {
    cout << sum(5, 4) << endl;
    cout << sum(5, 4, 3) << endl;

    return 0;
}
```

```
❯ cd "/Users/javhln/Desktop/Object Oriented/
9
12
```

4. Динамик санах ой (new, delete оператор ашигласан) болон байгуулагч, устагч функцийг хэрхэн хамтад нь ашиглах вэ?

Динамик санах ой нь ой бэлдэх, дараа нь чөлөөлөх гэсэн 2 үйлдэл хийнэ. Байгуулагч функц дотор объектын гишүүн өгөгдөлд санах ой нөөцлөх боломжийг динамикаар буюу new оператор ашиглаж хийж болно. Харин динамикаар нөөцөлсөн санах ойг чөөлөөчлөх үйлдлийг delete оператор хийнэ.

New оператор нь сул ойгоос нөөцөлж авсан ойн эхлэл хаягийг буцаана.

```
class Student {
  public:
     Student() {
        cout<<"Constructor\n";
     }
     ~Student() {
        cout<<"Destructor\n";
     }
};
int main() {
  Student *s = new Student();
  delete(s);
  return 0;
}
```

5. Хэрэгжүүлэлт

```
class Employee {
  private:
     int id;
     string name;
     string position;
     float hoursWorked;

  public:
     Employee() {
        id = 0;
        name = "";
        position = "";
```

```
        hoursWorked = 0;
    }
    Employee(int id, string fname, string pos, float hrs) {
        id = id;
        name = fname;
        position = pos;
        hoursWorked = hrs;
    }
    int getId() {
        return id;
    }
    string getName() {
        return name;
    }
    string getPosition() {
        return position;
    }
    int getHoursWorked() {
        return hoursWorked;
    }
    void setId(int i){
        id = i;
    }
    void setName(string fname) {
        name = fname;
    }
    void setPosition(string pos) {
        position = pos;
    }
    void setHoursWorked(float hrs) {
```

```
        hoursWorked = hrs;
    }
    float calSalary() {
        if(position == "zahiral") {
            return salaryBoss();
        }
        return hoursWorked * 5;
    }
  private:
    float salaryBoss() {
        return hoursWorked * 8;
    }
  public:
    bool incHoursWork(float hrs) {
        if(hrs ⩾ 0 && hrs ⩽ 24) {
            hoursWorked += hrs;
            return 1;
        }
        return 0;
    }
    ~Employee() {
        cout << "Destructor working\n";
    }
};

int main() {
  Employee emp[2];
  int i, j, id;
  float hrs;
  string pos, name;
```

```
  for (i = 0; i <= 2; i++) {
     cout << "ID: ";
     cin >> id;
     emp[i].setId(id);
     cout << "Ner: ";
     cin >> name;
     emp[i].setName(name);
     cout << "Alban tushaal: ";
     cin >> pos;
     emp[i].setPosition(pos);
     cout << "Ajilsan tsag: ";
     cin >> hrs;
     emp[i].setHoursWorked(hrs);
  }
  for(i = 0; i <= 2; i++) {
     for(j = i + 1; j <= 2; j++)
     {
        if(emp[i].calSalary() >emp[j].calSalary())
        {
           Employee temp = emp[i];
           emp[i] = emp[j];
           emp[j] = temp;
        }
     }
  }
  return 0;
}
```