Структуры и классы

Структура – группировка массивов нескольких типов данных.
struct Person
{
char name[50];
int date.b;
long int passport;
}
...
Person A;
A.date=2001;
A.passport="1111111111";

Программой будет выделена память в бОльшем объёме, чем требует пользователь.
Компилятор может потребовать другого объявление структуры:

typedef struct {...} Person;

Можно ставить указатель, но мы можем потерять данные в динамической памяти.
В классах есть не только поля. Есть встроенные функции (методы) для работы с данными.

struct student
{
char* name;
int A[10];
double avg();
};

student n; ...
cout«n.avg – считает среднее для одного студента
double student:: avg()
{
double sum=0;
for(int i=0; i<10; sum+=A[i++]);
return sum/10;
}

Модификаторы доступа:
public – обращение снаружи класса
privat – обращение только внутри класса
В структуре все поля по умолчанию открыты, а в классе – закрыты.
Деструктор – метод, вызываемый, когда элемент исчезает. С помощью него

можно очищать динамическую память. Используем, если вызывалась дополнительная память.

```
struct student
{ char* name;
int A[10];
double avg();
student()
{
delete[]name;
} – удаляет память, выделенную под имя
}
```

Второй стандартный метод – конструктор. У него могут быть аргументы. Вызывается, когда мы создаем объект структуры.

```
student (char* Ar, int* a)
{
name = new char[256];
for (int i=0; str[i]='0';i++)
{
name[i]=str[i];
for (int i=0; i<10; i++)
A[i]=a[i]
}
```

Перегрузка функций - способ создания функций с одинаковым названием, но разными аргументами.
Функция operator= определена почти для всех типов.
A.operator=[B] то же, что A=B.
Вся арифметика перегружаема, все сравнения перегружаемы Конструктор тоже перегружаем.
Есть конструктор, который принимает в себя ссылку на следующую структуру student (const student& a) – конструктор копий
Если напишем A(B), то будет создан объект A, такой же как B, т.е. A=B.

Правило Трёх. Если определяется или деструктор, или конструктор копий, или оператор самостоятельно, то остальные два тоже определяются вручную. Перегрузка сравнений. Если есть сравнение A=B можно сделать A != B логическим отрицанием !(A==B)
A>B
A< B !(A>B) && (A != B)