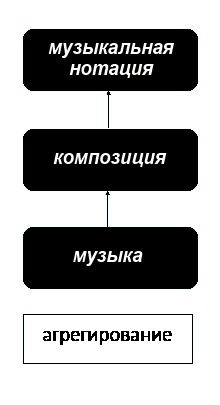
**Курсовая работа**

Абстрактный тип данных(АТД) – Музыка. Первая задача состояла в разработке схем агрегирования, одиночного наследования, множественного виртуального наследования. После завершения построения схем, мы начали с простейших форм класса – структуры и переменной структуры. Постепенно мы перешли к более сложным формам объектно-ориентированного программирования – классам. На следующем этапе были добавлены функции-друзья класса. А завершилась работа созданием одиночного наследования, где показано более сложное взаимодействие одного АТД с несколькими другими. Рассмотрим каждый этап работы.

**Агрегирование**

Отношение агрегирования между классами имеет место, когда один класс содержит в качестве составной части объекты другого класса. Иными словами, это отношение «целое/часть», между классами.

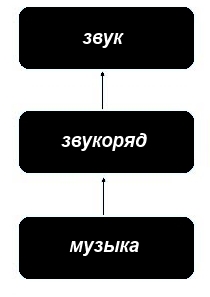
Данное отношение я применил и для своего абстрактного типа данных Музыка.



**Одиночное наследование**

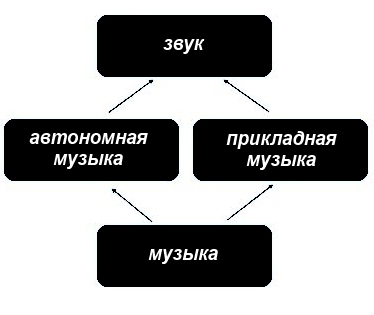
Одиночное наследование — это такое отношение между классами , когда абстрактный тип данных может наследовать данные и функциональность некоторого существующего типа, способствуя повторному использованию компонентов.

В своей программе я реализовал одиночное наследование таким образом. Музыка – есть (звукоряд) совокупность последовательных звуков, расположенных по высоте в восходящем или нисходящем порядке. А звукоряд – подвид звука.



**Множественное виртуальное наследование**

Множественное наследование позволяет создавать новый класс(наследник), который перенимает функционал нескольких базовых классов(родителей). При этом производный класс может быть наделен дополнительным функционалом. Множественное наследование достаточно мощный инструмент, но при его использовании могут возникнуть достаточно серьезные проблемы, если, например, производный класс наследует от двух классов, которые, в свою очередь, наследуют от одного и того же базового класса. Такая ситуация носит название "ромбовидное наследование". Решением служит виртуальное наследование. Оно решает проблему наследования нескольких экземпляров базового класса, когда нужен только один разделяемый экземпляр. Для АТД "Музыка" иерархия МВН представлена следующим образом:



**Структура**

Структура — это тип данных, содержащий без сокрытия набор значений различных типов. Порядок размещения значений в памяти задаётся при определении типа и сохраняется на протяжении времени жизни объектов, что даёт возможность косвенного доступа. Структура является простейшей формой класса. На первом этапе создания абстрактного типа данных я начал именно со структуры.

Структура Музыка содержит в себе компоненты:

* ФИО композитора(char)
* Название композиции(char)
* Длительность(float)
* Жанр композиции(unsigned short int)

Функции:

* int Init\_Kompozition //Инициализация полей структуры
* Print\_Albom //Печать всего массива композиций
* int Print\_Kompozition //Печать выбранной композиции

Фрагмент кода структуры:

//структура-описатель музыкальной композиции

typedef struct

{

char Name\_Kompozitor[MAXNAME\_K]; //ФИО композитора

char Name\_Music[MAXNAME\_M]; //Название композиции

float minute; //Длительность композиции в минутах

t\_usi ganr; //жанр произведения

}KOMPOZITION;

//Инициализация полей структуры

int Init\_Kompozition(KOMPOZITION \*komp,int MxNum\_Komp, char \*\_Name\_Kompozitor, char \*\_Name\_Music, float \_minute, t\_usi \_ganr)

// Печать всего массива композиций

void Print\_Albom(KOMPOZITION \*\_albom)

// Печать выбранной композиции

int Print\_Kompozition(KOMPOZITION \*\_albom,int \_Num\_Komp)

**Переменная структура**

Переменная структура является также простейшей формой класса. Переменная структура включает в себя совместное использование объединения и структуры. Объединение позволяет хранить данные разных типов в одном и том же месте памяти(но не одновременно). Объединение формируется подобно структурам. Структура может хранить значения типов(unsigned short int, float, char) одновременно, однако объединение может хранить значения тип unsigned short int или float или char.

Фрагмент кода:

//структура-описатель музыкальной композиции

typedef struct

{

char Name\_Kompozitor[MAXNAME\_K]; //ФИО композитора

char Name\_Music[MAXNAME\_M]; //Название композиции

float minute; //Длительность композиции в минутах

t\_usi ganr; //жанр произведения

}KOMPOZITION;

//Инициализация полей структуры

int Init\_Kompozition(KOMPOZITION \*komp,int MxNum\_Komp, char \*\_Name\_Kompozitor, char \*\_Name\_Music, float \_minute, t\_usi \_ganr)

// Печать всего массива композиций

void Print\_Albom(KOMPOZITION \*\_albom)

// Печать выбранной композиции

int Print\_Kompozition(KOMPOZITION \*\_albom,int \_Num\_Komp)

**Класс**

Класс является основной частью языка С++, и очень важны в объектно-ориентированном программировании. Класс - это некоторая абстракция, которая описывает свойства и методы необходимые для данного объекта. Члены класса могут иметь один из трех спецификаторов доступа: public-открытые, private-закрытые, protected-защищенные. При использовании класса появляется возможность защитить данные класса от внешнего вмешательства пользователя, присвоив им спецификатор доступа private. Для инициализации данных класса используются специальные методы - конструкторы, а за удаление объектов отвечает другой специальный метод - деструктор.

Для АТД "Музыка" весь компонентный состав имеет спецификатор private, а его методы - public. В классе имеется четыре вида конструкторов. Конструктор по-умолчанию - инициализирует поля класса некоторыми стандартными значениями. Конструктор с параметрами - инициализирует поля класса параметрами, которые передаются в качестве аргументов конструктора. Конструктор копирования - служит для создания копии выбранного объекта. Конструктор преобразования - изменяет значение одного из полей класса, а все остальные инициализируются некоторыми стандартными значениями. В деструкторе файл закрывается, т.е. происходит высвобождение захваченного ресурса. Так же были добавлены друзья класса - перегруженные операторы вывода данных. Благодаря перегруженному оператору вывода, отпала необходимость иметь отдельно функцию визуализации данных.

Фрагмент кода класса:

class TKOMPOZITION //структура-описатель музыкальной композиции

{

private:

char Name\_Kompozitor[MAXNAME\_K]; //ФИО композитора

char Name\_Music[MAXNAME\_M]; //Название композиции

t\_usi minute; //Длительность композиции в целых минутах

t\_usi ganr; //жанр произведения

static int Count\_Komp; //реальное число существующих в данный момент экземпляров класса

t\_usi Err; //код завершения

public:

TKOMPOZITION(); //конструктор по умолчанию

TKOMPOZITION(char \*\_Name\_Kompozitor, char \*\_Name\_Music, t\_usi \_minute, t\_usi \_ganr);

TKOMPOZITION(TKOMPOZITION &\_komp); //конструктор копирования

TKOMPOZITION(float \_minute); //конструктор преобразования

~TKOMPOZITION(); //деструктор

// Дружественный метод, обеспечивающий вывод

friend ostream &operator << (ostream &stream, TKOMPOZITION &obj);

};

**Реализация одиночного наследования**

Следующим шагом стала реализация приведенной выше иерархии одиночного наследования. На самом нижнем уровне иерархии находится "Музыка". Полезными методами класса является Методы, изменяющие значения приватных полей класса и возвращающие значения приватных полей класса .Также имеется Дружественный метод, обеспечивающий вывод.. класс "Музыка"(Муз. комп.) наследует от "Звукоряда"(Аккорд), а он наследует от “Звука”(нота).

Фрагмент кода реализации одиночного наследования:

// Класс НОТА (частота / длительность - звучание)

class TNOTA : public TERROR

{

public:

t\_ui freq;

t\_ui milisec;

TNOTA(){};

TNOTA(float \_freq, t\_ui \_milisec):freq(\_freq),milisec(\_milisec){};

void Set(float \_freq, t\_ui \_milisec);

t\_ui GetFreq(void) const {return freq;};

t\_ui GetMilisec(void) const {return milisec;};

void Play(){ Beep(freq,milisec);};

TNOTA operator = (TNOTA &t){ freq = t.freq; milisec = t.milisec;};

};

// Класс АККОРД (совокупность нот - звучание)

class TAKKORD : public TNOTA

{

public:

TNOTA \*ak;

t\_usi CoNot,IndNot;

TAKKORD(){IndNot = 0; ak = NULL; CoNot=0;};

TAKKORD(t\_usi CoNot);

~TAKKORD()

{

if(ak != NULL){delete ak; ak = NULL;}

}

void SetCoNot(t\_usi \_CoNot);

void AddNot(float \_freq, t\_ui \_milisec);

t\_usi GetCoNot() const {return IndNot;};

void PlayA()

{

for(t\_usi i=0; i< IndNot; i++) ak[i].Play();

};

// Класс МУЗ.КОМПОЗИЦИЯ (авор - название - жанр - звучание)

class TKOMPOZITION : public TAKKORD //структура-описатель музыкальной композиции

{

private:

char Name\_Kompozitor[MAXNAME\_K]; //ФИО композитора

char Name\_Music[MAXNAME\_M]; //Название композиции

t\_usi ganr; //жанр произведения

TAKKORD \*melodia; //мелодия

t\_usi CoAkkord,IndAkkord; //Max./реально существующее число аккордов в композиции

public:

TKOMPOZITION(); //конструктор по умолчанию

TKOMPOZITION(char \*\_Name\_Kompozitor, char \*\_Name\_Music, t\_usi \_ganr,t\_usi \_CoAkkord);

TKOMPOZITION(TKOMPOZITION &\_komp); //конструктор копирования

// TKOMPOZITION(float \_minute); //конструктор преобразования

~TKOMPOZITION(); //деструктор

void SetCoAkkord(t\_usi \_CoAkkord);

void AddAkkord(TAKKORD &ta);

// Методы, изменяющие значения приватных полей класса

void Put\_Name\_Kompozitor(char \*\_Name\_Kompozitor);

void Put\_Name\_Music(char \*\_Name\_Music);

void Put\_Ganr(t\_usi \_ganr){ganr = \_ganr;};

// Методы, возвращающие значения приватных полей класса

char \*Get\_Name\_Kompozitor(void) {return Name\_Kompozitor;};

char \*Get\_Name\_Music(void) {return Name\_Music;};

t\_usi Get\_Ganr(void)const {return ganr;};

void PlayK()

{

for(t\_usi i=0; i< CoAkkord; i++) melodia[i].PlayA();

};

// Дружественный метод, обеспечивающий вывод

};

**Множественное виртуальное наследование**

На этом этапе необходимо проработать компонентный состав классов, учавствующих в МВН. За основу будет взята иерархия МВН, приведенная выше. В самом низу иерархии находится класс "Музыка". Его компонентным составом ФИО композитора/Название композиции/Длительность композиции в минутах/жанр произведения. Класс имеет конструктор по-умолчанию и конструктор копирования. Полезный метод класса – вывод информации. Деструктор предназначен для высвобождения захваченного ресурса, т.е. для закрытия файла. Компоненты класса имеют спецификатор доступа - private, а конструкторы, деструктор и полезный метод - public.

Музыка является подвидом автономной и прикладной музыки одновременно.

Компоненты класса "Автономная музыка" : частота, длительность, художественные жанры

Компоненты класса "Прикладная музыка": частота, длительность, прикладные жанры

Оба класса являются подвидами "звука" в целом. Это значит, что они оба наследуют от одного и того же класса. Каждому виду звука присвоена нота. Поэтому компонентом класса "Звук" является частота, длительность -звучание. Члены каждого базового класса, в том числе и их конструкторы, имеют спецификатор доступа protected, чтобы исключить возможность создания объектов базовых классов, но в то же время предоставить доступ к компонентам базовых классов из производного.

**Диаграмма МВН**

