Основы ООП

- Классы и объекты. Язык C++
- Члены класса: свойства и методы
- Обращение к членам класса: изнутри и вовне
- Инкапсуляция
- Статические члены класса
- Тип данных «ссылка»
- Константные объекты и методы
- Перегрузка функций и операций в языке C++. Значения аргументов по умолчанию.
- Конструкторы и деструкторы
- Управление доступом к членам класса

Объектно-ориентированное программирование

- Класс - это некое множество объектов, имеющих общую структуру и поведение.
- В языке C++ каждый класс является типом данных.
- В отличие от структуры в состав класса могут входить не только переменные, но и функции.
- Объект — это экземпляр класса, например переменная данного типа или динамически выделенная область памяти.

Описание класса

- Перед использованием класс должен быть описан: `class <имя класса> { <объявления членов класса> };`
- Членами класса могут быть переменные (свойства класса) и функции (методы класса).
- При описании класса указываются объявления переменных, входящих в его состав.
- Для функций указывается прототип, либо полное описание функции.

Задание

Опишите класс `Student`, содержащий номер курса `Course`, количество предметов `Subjects` (целые числа), и массив оценок по предметам `Rating` (15 дробных чисел), а также функции вычисления среднего рейтинга `AvgRating` (без параметров, возвращает дробное число) и принятия решения `Decision` (входной параметр — `finale` — целое число).

Описание класса

```
class Student
{
  int Course;
  int Subjects;
  float Rating[15];
  float AvgRating();
  void Decision(int finale);
};
```

Описание методов класса

- Код функции, принадлежащей классу, может быть описан сразу в классе (в этом случае прототип опускается).
- При описании функции, принадлежащей классу, вне описания класса ее заголовок выглядит следующим образом: `<тип возвращаемого значения> <имя класса>::<имя функции>(<аргументы>)`
- В теле функции, принадлежащей классу, могут использоваться переменные (свойства) класса как локальные переменные.

Задание

- Опишите код функции `AvgRating`, возвращающей средний рейтинг студента внутри класса.
- Напишите заголовок функции `Decision` (с многоточием в качестве тела функции), описываемой вне класса.

Описание методов класса

```
class Student
{
   int Course;
   int Subjects;
   float Rating[15];
   float AvgRating()
   {
      float sum=0;
      for(int i=0;i<Subjects;i++)
         sum+=Rating[i];
      return sum/Subjects;
   }
   void Decision(int finale);
};

void Student::Decision(int finale)
{...}
```

Объявление и описание класса

- Методы обычно описывают вне класса, чтобы сделать описание класса более удобным для чтения.
- Описание класса обычно размещается в заголовочном файле (`.h` или без расширения), который подключается к каждому файлу, в котором используется этот класс. Обычно имя файла совпадает с именем класса.
- Описание кода функций класса обычно размещается в `.cpp` файле с тем же именем, что и заголовочный.
- При необходимости использовать имя класса до описания он может быть объявлен: `class <имя класса>;`

«Внутри» и «вовне» класса

- С точки зрения класса весь код программы разделен на внутренний и внешний.
- Внутренним является код функций, принадлежащих классу, внешним — остальной код.
- Обращение к элементам (свойствам и методам) класса изнутри и извне различно.

Обращение к элементам класса

- Обращение к свойствам и методам данного класса изнутри происходит так же, как и к обычным переменным и функциям.
- При обращении к свойствам и методам класса снаружи используются операторы доступа `.` (точка) и `->` (стрелка), как и для структур.

Обращение к элементам класса

- При обращении к элементам класса извне необходимо указывать объект (значение или указатель), для которого осуществляется обращение.

  Любое действие производится над конкретным объектом.
- При обращении изнутри функция уже знает, для какого объекта она вызвана, поэтому дополнительных указаний не требуется.

Задание

Объявите переменную (объект) `Ivanov` типа `Student`.

Присвойте студенту Иванову первый курс, 2 предмета с рейтингом 67 и 89 баллов.

Напечатайте на экран средний рейтинг студента Иванова.

Обращение к элементам класса

```
Student ivanov;
ivanov.Course=1;
ivanov.Subjects=2;
ivanov.Rating[0]=67;
ivanov.Rating[1]=89;
printf("Ivanov: %f",
  ivanov.AvgRating());
```

Инкапсуляция

- Для того, чтобы пользоваться классом, не требуется знать его внутреннее устройство.
- Достаточно знать минимальный набор методов (иногда и свойств), предназначенный для взаимодействия с внешним миром — интерфейс класса.
- Говорят, что класс скрывает (инкапсулирует) детали своей реализации.

Задание

Существует класс `QString`, представляющий строку, содержащий, в частности, функцию replace, чьими входными параметрами являются две строки (`QString`): заменяемая часть строки и заменяющая ее часть. Объявите строку `MyOpinion` и присвойте ей надпись «`Life is bad!`». Замените подстроку «`bad`» на «`good`».

Примечание: строкой в кавычках можно инициализировать переменную типа `QString`.

Инкапсуляция

```
QString MyOpinion="Life is bad!";
QString re1="bad";
QString re2="good";
MyOpinion.replace(re1,re2);
```

Статические члены класса

- Статические свойства и методы класса принадлежат не конкретным объектам, а классу в целом. При их объявлении указывается ключевое слово `static`.
- Существует одно значение статического свойства на весь класс.
- Из статического метода можно обращаться лишь к статическим свойствам класса.
- К статическим свойствам и методам обращаются с помощью оператора ::

```
<имя класса>::<имя свойства> или
<имя класса>::<имя метода>(<аргументы>)
```

Задание

В классе `QString` существует обычный (динамический) метод `toInt`, возвращающий число, хранящееся в данной строке , а также статический метод `number`, возвращающий строку по данному числу.

Объявите целое число `n` и строку `str`, которой присвойте строку «`10`».
Присвойте `n` число из строки и прибавьте к нему единицу.
Занесите измененное `n` в строку `str`.

Статические и динамические члены класса

```
QString str="10";
n=str.toInt();
n++;
str=QString::number(n);
```

Тип данных «ссылка»

- В языке C++ введен новый тип данных — ссылка.
- Ссылка ведет себя как указатель, но не требует применения операций `*` (`->`) и `&`.
- Объявление ссылки: `<тип данных> &<имя переменной>`
- Переменная-ссылка должна быть инициализирована. Используется очень редко.
- Чаще всего ссылки применяются в качестве аргументов функций вместо указателей (передача по ссылке).

Задание

В данном заголовке функции найдите ссылки и определите их назначение:

```
QString & QString::replace ( int
 position, int n, const QString &
 after )
```

Использование ссылок

```
QString & QString::replace ( int position, int n, const QString & after )
```

Аргумент `after` передается по ссылке, поскольку он является большим объектом (ссылка указывает на константный объект, поэтому изменения невозможны).

Возвращаемое значение также является ссылкой. Это дает возможность модифицировать возвращаемое функцией значение (например поставив вызов слева от знака присваивания)

Псевдопеременная this

- Иногда методу класса требуется сослаться непосредственно на тот объект, для которого она была вызвана (а не на его свойства) — например для передачи его в другую функцию или указания в качестве возвращаемого значения.
- Динамический метод класса может использовать (без объявления) псевдопеременную `this`, являющуюся указателем на текущий объект класса.

Задание

Напишите метод `Get` класса `Data`, который считывает текущий объект из файла, вызывая *статический* метод `Read` класса `File`, передавая ему в качестве параметров указатель на текущий объект и размер объекта; после чего возвращает ссылку на текущий объект.

Псевдопеременная this

```
Data & Data::Get(void)
{
    File::Read(this,sizeof(Data));
    return *this;
}
```

Константные объекты и методы

- Объект (переменная-значение, либо указатель) может быть объявлен константным средствами языка C.
- Нельзя присвоить константный объект неконстантному, а также менять значения его свойств.
- Метод класса считается константным, если после заголовка метода стоит ключевое слово `const`.
- Константный метод не может менять свойства класса.
- Для константного объекта могут вызываться только константные методы.

Задание

Дан класс

```
class QString
{
  ...
  int length () const ;
  QString & replace ( const QString & before, const
  QString & after) ;
  ...
};
```

Найдите ошибки в следующем фрагменте кода

```
QString str="Life is bad!";
const QString str1="bad";
QString str2="good";
int n=str1.length();
int m=str.length();
str.replace(str1,str2);
str1.replace(str1,str2);
str1=str;
str=str1;
```

Константные объекты и методы

```
QString str="Life is bad!";
const QString str1="bad";
QString str2="good";
int n=str1.length();
int m=str.length();
str.replace(str1,str2);
str1.replace(str1,str2);/* ОШИБКА: Вызов
  неконстантной функции для константного
  объекта*/
str1=str;
str=str1;/*ОШИБКА: попытка присвоить
  константный объект неконстантному*/
```

Перегрузка функций в языке C++

- В языке C++ допускается наличие нескольких функций с одинаковым именем, если они различаются количеством и/или типами входных параметров.
- Различие в типе выходного значения не учитывается.
- При вызове язык определяет, какую из перегруженных функций вызывать по типам фактических аргументов.

Задание

Определите, какая из перечисленных ниже функций будет вызвана при выполнении строк

```
int i;
float f;
struct student {int age; float balance;};
struct student c;
char str[100];
function(f+2.0,&(c.age),str[i]);
```

Функции:

A. `void function(float*,int,char);`
B. `double function(float,int,int);`
C. `char function(float,int*,char);`
D. `float function(float*,int*,int);`

Перегрузка функций

- Первый аргумент является суммой двух `float` чисел (переменной `f` и константы `2.0`) и имеет тип `float`.
- Второй аргумент является адресом поля `age` структуры `student`, которое имеет тип `int`, поэтому его типом является указатель на `int` (`int *`).
- Третий аргумент является элементом символьного массива `str` и его тип - `char`.
- Поэтому правильным ответом будет:
  `C. char function(float,int*,char);`

Перегрузка операций

- Язык C++ позволяет описывать действия, выполняемые при применении операций (например `+`, `=`, `[ ]` и т.д.) к объектам классов.
- Имя функции, определяющей работу перегруженной операции, состоит из слова `operator` и символа(ов) операции (например `operator+`, `operator[]`)
- Например в классе `QString` операция `+=` выполняет конкатенацию строк (как функция `strcat`)

Значения аргументов по умолчанию

- При описании функции возможно задать значения по умолчанию для входных параметров, т.е. значение, которое будет присвоено им если они не указаны.
- Значение по умолчанию указывается в заголовке функции в виде инициализации соответствующих аргументов (<тип> <имя>=<значение>).
- Аргументы со значением по умолчанию принято располагать в конце списка аргументов.

Задание

- Напишите заголовок функции number класса QString, описываемой вне класса, переводящей число в строку. Функция получает два аргумента: n — переводимое целое число, base — основание системы счисления (по умолчанию 10). Функция возвращает строку.
- Напишите вызовы этой функции (она является *статической*) для перевода числа 20 в строку в десятичной и двоичной системе счисления

Значения аргументов по умолчанию

```
QString QString::number (int n,int
  base=10)
{...}

QString str10=QString::number(20);
QString
  str2=QString::number(20,2);
```

Конструкторы

- Конструктором называется функция, которая автоматически вызывается языком программирования при создании объекта класса. Напрямую вызвать конструктор нельзя.
- Класс может иметь несколько перегруженных конструкторов (они должны различаться аргументами).
- Конструктором в языке C++ является функция, не имеющая возвращаемого значения, имя которой совпадает с именем класса.
- Заголовок конструктора:
  `<имя класса>(<аргументы>)`

Использование конструкторов

- При объявлении переменной-объекта значения аргументов конструктора указываются в круглых скобках после имени переменной: `<тип> <имя переменной>(<аргументы конструктора>);`
- Конструктор также может быть вызван для создания временного объекта без объявления переменной: `<тип>(<аргументы конструктора>)`
- Третий вариант вызова конструктора включает использование динамического выделения памяти.

Виды и назначение конструкторов

- Конструктор по умолчанию — конструктор без аргументов — вызывается при обычном создании объекта.
- Конструктор копирования — конструктор с одним параметром, являющимся ссылкой на объект данного класса — вызывается при инициализации, а также передаче аргумента в функцию по значению

Виды и назначение конструкторов

- Конструктор с одним аргументом — часто используется для преобразования типа данных. Например класс `QString` имеет конструктор с аргументом типа const char *, что позволяет легко преобразовывать строковые константы в объекты `QString`.
- Конструктор с несколькими аргументами обычно используется для задания начального состояния сложных объектов.

Задание

- Напишите часть класса `Student`, разместив в нем конструктор по умолчанию с кодом (курс - 1, количество предметов — 0), а также прототипы конструктора копирования, и конструктора с двумя целыми числами: курсом и количеством предметов.
- Опишите код конструктора с двумя целыми числами вне пределов описания класса.
- Создайте студента иванова на 2-м курсе с 10 предметами используя конструктор.

Конструкторы

```
class Student
{
    ...
    Student()
    {
        Course=1;
        Subjects=0;
    }
    Student(Student &source);
    Student(int cour, int subj);
    ...
};

Student::Student(int cour, int subj)
{
    Course=cour;
    Subjects=subj;
}

Student Ivanov(2,10);
```

Деструкторы

- Деструктором называется функция, автоматически вызываемая языком программирования при уничтожении объекта (завершении времени жизни переменной, освобождение динамически выделенной памяти).
- Деструктор не имеет типа возвращаемого значения и аргументов, его имя — это имя класса со знаком ~ (тильда) перед ним.
- Заголовок деструктора: `~<имя класса>()`

Управление доступом к членам класса

- В описании класса могут встречаться модификаторы доступа `public`, `protected` и `private`.
- Форма записи: `<модификатор доступа>:`
- Действие модификатора распространяется на все члены класса, объявленные после него до следующего модификатора.
- По умолчанию члены класса имеют тип доступа `private`

Управление доступом к членам класса

- Свойства и методы класса с типом доступа `public` можно вызывать в любом месте программы, `protected` и `private` — только из своего класса (с определенными исключени
- Интерфейс класса (методы, предназначенные для вызова внешними объектами) имеет тип доступа `public`, реализация (свойства и служебные методы) — обычно `protected`.

Задание

Опишите класс `Student`, содержащий номер курса `Course`, количество предметов `Subjects` (целые числа), и массив оценок по предметам `Rating` (15 дробных чисел), а также функции вычисления среднего рейтинга float `AvgRating()`, подсчета суммы баллов int `Sum()` и `void SetRating(int subject, int rating)`.

Функции `AvgRating` и SetRating имеют тип доступа `public`, все остальное - `protected`.

Управление доступом к членам класса

```
class Student
{
 public:
  float AvgRating();
  void SetRating(int subject, int rating);
 protected:
  int Course;
  int Subjects;
  float Rating[15];
  int Sum();
};
```

Задание

Опишите код функций `AvgRating`, `SetRating` и `Sum` вне класса так, чтобы функция `AvgRating` использовала `Sum`, а `SetRating` проверяла бы номер предмета и диапазон баллов на допустимость

Номер предмета должен быть меньше количества предметов, баллы — от 0 до 100.

Управление доступом к членам класса

```
float Student::AvgRating()
{
    return Sum()/Subjects;
}
int Student::Sum()
{
    int sum=0;
    for(int i=0;i<Subjects;i++)
    {
        sum+=Rating[i];
    }
    return sum;
}
void Student::SetRating(int subject, int rating)
{
    if(subject>=0 && subject<Subjects && rating>=0 &&
      rating<=100)
        Rating[subject]=rating;
}
```

Агрегация

- Агрегацией называют включение объекта одно класса в качестве свойства объекта другого класса.
- Агрегация — это связь между объектами, в обычном мире ей соответствует отношение «быть частью».
- Объект, который является свойством другого объекта называют агрегируемым объектом (частью); объект, включающий в себя несколько объектов как свойства называется агрегатом.

Виды агрегации

- При агрегации по значению в состав агрегата входит свойство со значением агрегируемого объекта, при агрегации по указателю — указатель на агрегируемый объект.
- При агрегации по значению агрегируемый объект является неотъемлемой частью агрегата, он создается и уничтожается одновременно с ним.
- При агрегации по указателю агрегируемый объект создается и уничтожается независимо от агрегата.

Вопрос

Какой бы вид агрегации вы выбрали для отображения следующих связей:

- монитор является частью компьютера;
- голова является частью тела;
- студент является частью группы.