Билет 1:производный класс, атрибуты.   
Найти площадь прямоугольника в базовом классе, в переопределенном методе производного класса найти площадь круга

#include "stdafx.h"

#include <iostream>

using namespace std;

class Bcls{

protected:

double a,b;

public:

Bcls(double x, double y): a(x), b(y){cout<<"oject PRIAMOUGOL'NIK is created";}

Bcls(double x):a(x){}

~Bcls(){cout<<"oject PRIAMOUGOL'NIK is destructed";}

double square(){

double sq=1;

sq=a\*b;

return sq;}

};

class Pcls: public Bcls{

public:

Pcls(double q): Bcls(q){cout<<"oject KRUG is created";}

double square(){

double sq=1;

sq=2\*3.14\*a\*a;

return sq;}

~Pcls(){cout<<"oject KRUG is destructed";}

};

int \_tmain(int argc, \_TCHAR\* argv[])

{double sq;

Bcls a(5,4);

sq=a.square();

cout<<sq<<endl;

Pcls b(1);

cout<<b.square()<<endl;

system("pause");

return 0;

}

Билет 2  
1) дружественные функции  
2) в базовом классе написать метод для подсчета длины   
окружности, в производном переопределить его для нахождения длин прямоугольника.   
Использовать конструктор и деструктор

#include "stdafx.h"

#include <iostream>

using namespace std;

const double pi=3.14;

class Bcls{

protected:

int r;

public:

Bcls(){}

Bcls(int a):r(a){}

void foo(){

double l=1;

l=2\*pi\*r\*r;

cout<<"L: "<<l<<endl;

}

};

class Pcls: public Bcls{

int k;

public:

Pcls(int a, int b):Bcls(a), k(b){}

void foo(){

cout<<"S: "<<r\*k<<endl;

}

};

int \_tmain(int argc, \_TCHAR\* argv[])

{Bcls a(1);

a.foo();

Pcls b(2,3);

b.foo();

system("pause");

return 0;

}

3 билет  
Поиск элемента в массиве, контролировать выход счётчика за пределы массива переопределением операции [ ], использовать базовый и производный класс  
Теорией конструктор, пример

#include "stdafx.h"

#include <iostream>

using namespace std;

const int n=5;

class Bcls{

protected:

int mas[n];

};

class Pcls: public Bcls{

public:

int & operator [](int m){

if(m<0||m>=n) {cout<<"Massive is overloaded!"<<endl; exit(1);}

return mas[m];}

int poisk(int x){

for(int i=0;i<n;i++){

if(mas[i]==x) return i;

}

cout<<"X wasn't found.";

return -1;}

};

int \_tmain(int argc, \_TCHAR\* argv[])

{

Pcls m;

for(int i=0;i<n;i++) m[i]=i\*10;

cout<<"the position of number 40 is: "<<m.poisk(40)<<endl;

system ("pause");

return 0;

}

билет 4   
1.Стандартный класс String   
2. Найти сумму элементов массива. Базовый класс должен быть абстрактным.

[#include](https://vk.com/im?sel=512906158&st=%23include) "stdafx.h"   
[#include](https://vk.com/im?sel=512906158&st=%23include) "iostream"   
using namespace std;   
  
const int n = 5;   
class AbstractArrayClass {   
  
protected:   
int \*mas;   
public:   
AbstractArrayClass() { mas = new int[n]; }   
  
virtual int ArraySum() = 0;   
  
~AbstractArrayClass() {delete []mas;}   
  
};   
  
  
class ArrayClass : public AbstractArrayClass {   
  
public:   
  
ArrayClass() {}   
  
int ArraySum() {   
  
int s = 0;   
  
for (int i = 0; i < n; i++) {   
  
s += mas[i];   
  
}   
  
return s;   
}   
  
void PutOut() {   
  
for (int i = 0; i<n; i++) {   
  
cout « "[" « i « "]: "; cin » mas[i];   
  
}   
}   
  
};   
  
int \_tmain(int argc, \_TCHAR\* argv[])   
  
{   
ArrayClass b;   
  
b.PutOut();   
  
cout « "b: " « b.ArraySum() « endl;   
  
system("pause");   
  
return 0;   
}

Билет 5  
1. Контроль исключительных ситуаций ввода-вывода.  
2. В базовом классе вычислить сумму элементов массива, в производном вычислить их произведение, структурная обработка исключений

[#include](https://vk.com/im?sel=138446909&st=%23include)<iostream>   
using namespace std;   
class Scls   
{   
protected:   
int ms[10];   
public:   
Scls() {}   
~Scls() {}   
  
  
void init()   
{   
for (int i = 0; i < 10; i++)   
{   
ms[i] = rand() % 15;   
}   
}   
int sum\_elem()   
{   
int sum=0;   
for (int i = 0; i < 10; i++)   
{   
sum += ms[i];   
}   
return sum;   
}   
  
};   
  
class Pcls :public Scls   
{   
public:   
Pcls(){}   
~Pcls(){}   
int mult\_elem()   
{   
int mult=1;   
for (int i = 0; i < 10; i++)   
{   
mult = mult\*ms[i];   
}   
return mult;   
}   
  
};   
int main()   
{   
Scls obj1;   
Pcls obj2;   
obj1.init();   
int sum = obj1.sum\_elem();   
cout << "sum of elements=" << sum << endl;   
obj2.init();   
int mult = obj2.mult\_elem();   
cout << "multiplication of elements=" <<mult << endl;   
system("pause");   
}

Билет 8  
1. Перегрузка операций, свойства, особенности.  
2. Написать базовый класс, в котором нужно описать функцию суммы чисел, а в производном переопределить эту функцию, чтобы она искала сумму элементов массива

#include "stdafx.h"

#include <iostream>

using namespace std;

class Bcls{

protected:

int a,b,c;

public:

Bcls(int n, int m, int t):a(n), b(m), c(t){}

Bcls(int k): a(k){}

int sum(){

return a+b+c;

}

};

class Pcls: public Bcls{

int mas[100];

public:

Pcls(int k):Bcls(k){}

void pop(){

for(int i=0;i<a;i++){

cout<<"["<<i<<"]: ";

cin>>mas[i];

}

}

int sum(){

int s=0;

for(int i=0;i<a;i++)

s+=mas[i];

return s;

}

};

int \_tmain(int argc, \_TCHAR\* argv[])

{Bcls a(1,2,3);

Pcls b(5);

cout<<"a: "<<a.sum()<<endl;

b.pop();

cout<<"b: "<<b.sum()<<endl;

system ("pause");

return 0;

}

Билет 11  
1)Производный класс. Синтаксис  
2)Написать программу с перегрузкой(или переопределением) оператора декремент. Использовать базовый и производный классы.

[#include](https://vk.com/im?sel=138446909&st=%23include)<iostream>   
using namespace std;   
class Base   
{ protected:   
int a;   
int b;   
public:   
Base(int a1, int b1) { a = a1, b = b1; }   
Base operator--(int)   
{   
  
a--;   
b--;   
return \*this;   
}   
int get\_a()   
{   
return a;   
}   
int get\_b()   
{   
return b;   
}   
};   
class Son : public Base   
{   
int c;   
public:   
Son(int a1, int b1, int c1): Base(a1,b1),c(c1){}   
Son operator--(int)   
{   
a--;   
b--;   
c--;   
return \*this;   
}   
int get\_c()   
{   
return c;   
}   
  
};   
  
  
int main()   
{   
Base obj1(5, 6);   
Son obj2(2, 3, 4);   
cout << "a=" << obj1.get\_a() << endl << "b=" << obj1.get\_b() << endl;   
obj1--;   
cout<<"using operator —"<<endl << "a=" << obj1.get\_a() << endl << "b=" << obj1.get\_b() << endl;   
cout << "a=" << obj2.get\_a() << endl << "b=" << obj2.get\_b() << endl « "c=" << obj2.get\_c() << endl;   
obj2--;   
cout << "using operator —" << endl << "a=" << obj2.get\_a() << endl << "b=" << obj2.get\_b() << endl << "c=" << obj2.get\_c() << endl;   
system("pause");   
}

12 билет  
1. Деструктор и его особенности  
2. Найти частное. Использовать шаблон функции. Базовый м производный

#include "stdafx.h"

#include <iostream>

using namespace std;

class A1 {

public:

};

class A2 : public A1 {

public:

template<typename T>

T Fun(T A, T B)   
{   
if(B!=0)   
return A/B;   
else   
{   
cout « “B=0!”; return -1;   
}   
}

};

int main()

{

A2 obj;

cout « [obj.Fun](https://vk.com/away.php?utf=1&to=http%3A%2F%2Fobj.Fun" \t "_blank)(5, 2) « endl;   
 cout « [obj.Fun](https://vk.com/away.php?utf=1&to=http%3A%2F%2Fobj.Fun" \t "_blank)(5.00, 2.00);

system("pause");

return 0;

}

БИЛЕТ 13

1. Конструкторы вложенных классов

2. Найти минимальный элемент массива, при этом при этом создать производный класс и использовать функциональный оператор

[#include](https://vk.com/im?sel=294932335&st=%23include) "stdafx.h"   
[#include](https://vk.com/im?sel=294932335&st=%23include) <iostream>   
using namespace std;   
class Bcls{   
protected:   
int n;   
int \*mas;   
public:   
Bcls(int k):n(k){mas=new int[n]; for(int i=0; i<n; i++) cin»mas[i];

}   
int min(){   
int m=mas[0];   
for(int i=1;i<n;i++) if(mas[i]<m) m=mas[i];   
return m;   
}   
~Bcls(){   
delete [n]mas;   
}   
};   
class Pcls: public Bcls{   
public:   
Pcls(int k):Bcls(k){}   
int&operator ()(int a){   
if(a<0||a>n){   
cout<<"Going beyond the array";   
exit(1);}   
return mas[a];   
}   
};   
int \_tmain(int argc, \_TCHAR\* argv[])   
{int k; cout<<"Size of array: "; cin>>k;   
Pcls b(k);   
cout<<"Min: "«b.min();   
cout >>"mas[i]=… i?">>endl;   
cin<<k;   
cout<<b(k);   
system ("pause");   
return 0;   
}

14 билет.  
1. Перегрузка унарных операций. Пример.   
2. Найти площадь круга. Использовать и описать шаблон функции. Использовать базовый и производный классы

#include "stdafx.h"

#include <iostream>

using namespace std;

class A {

public:

const double pi = 3.14;

double r = 0;

};

class B : public A{

public:

B() {}

template<typename T>

T Square(T R)

{

return pi\*R\*R;

}

};

int main()

{

B obj;

cout << obj.Square(2.00)<<endl;

cout << obj.Square(2);

system("pause");

return 0;

}

Билет 15. 1- производный класс. Пример.  
2- в базовом классе описать функцию нахождения суммы массива, в производном переопределить эту функцию для нахождения суммы чётных элементов массива. Использовать дружественную функцию

[#include](https://vk.com/im?sel=138446909&st=%23include)<iostream>   
using namespace std;   
class Base   
{protected:   
friend void init(Base &obj);   
int ms[10];   
public:   
Base() {}   
void show()   
{   
for (int i = 0; i < 10; i++)   
{   
cout « ms[i] << " ";   
}   
cout << endl;   
}   
int sum\_elem()   
{   
int sum = 0;   
for (int i = 0; i < 10; i++)   
{   
sum += ms[i];   
}   
return sum;   
}   
  
};   
class Son : public Base   
{   
friend void init(Son &obj);   
public:   
int sum\_elem()   
{   
int sum = 0;   
for (int i = 0; i < 10; i++)   
{   
if(i%2==1)sum += ms[i];   
}   
return sum;   
}   
};   
  
void init(Base &obj)   
{   
for (int i = 0; i < 10; i++)   
{   
[obj.ms](https://vk.com/away.php?utf=1&to=http%3A%2F%2Fobj.ms)[i] = rand() % 15;   
}   
};   
void init(Son &obj)   
{   
for (int i = 0; i < 10; i++)   
{   
[obj.ms](https://vk.com/away.php?utf=1&to=http%3A%2F%2Fobj.ms)[i] = rand() % 15;   
}   
};   
  
int main()   
{   
Base obj;   
init(obj);   
[obj.show](https://vk.com/away.php?utf=1&to=http%3A%2F%2Fobj.show)();   
int sum = obj.sum\_elem();   
cout << sum << endl;   
  
Son obj2;   
init(obj2);   
[obj2.show](https://vk.com/away.php?utf=1&to=http%3A%2F%2Fobj2.show)();   
sum = obj2.sum\_elem();   
cout << sum;

system("pause");

return 0;  
  
}

16 билет  
1. Множественное наследование  
2. Найти минимальный элемент массива, используя переопределение операции [ ] и наследование

Билет 17  
1. Виртуальные функции  
2. Написать класс с методом, вычисляющим площадь квадрата. В производном от него классе перегрузить этот метод, чтобы теперь вычислялись длины сторон трапеции

#include "stdafx.h"

#include <iostream>

using namespace std;

class Bcls{

protected:

int a;

public:

Bcls(int m): a(m){}

void vychisl(){

cout<<"Ploshchad' kvadtata: "<<a\*a<<endl;

}

};

class Pcls: public Bcls{

int b,c,d;

public:

Pcls(int x, int y, int z, int q): Bcls(x),b(y),c(z),d(q){}

void vychisl(){

int per=(a+b+c+d);

cout<<"Perimetr storon: "<<per<<endl;

}

};

int \_tmain(int argc, \_TCHAR\* argv[])

{Bcls a(5);

a.vychisl();

Pcls b(5,7,8,8);

b.vychisl();

system("pause");

return 0;

}

Билет 18  
1. Потоковый ввод-вывод файлов  
2. Вычислить сумму элементов массива в базовом классе и среднее арифметическое в производном. Использовать статическую переменную для хранения размера массива

#include "stdafx.h"

#include <iostream>

using namespace std;

class Bcls{

protected:

static int n;

int \*mas;

public:

Bcls(){mas=new int[n];}

double foo(){

int s=0;

for(int i=0;i<n;i++){

s+=mas[i];

}

return s;}

void pop(){

for(int i=0;i<n;i++){

cout<<"["<<i<<"]: "; cin>>mas[i];

}}

~Bcls(){

delete [n]mas;

}

};

int Bcls::n=5;

class Pcls: public Bcls{

public:

Pcls(){}

double foo(){

int sr=0;

for(int i=0;i<n;i++){

sr+=mas[i];

}

return sr/n;

}

};

int \_tmain(int argc, \_TCHAR\* argv[])

{Bcls a;

a.pop();

cout<<"a: "<<a.foo()<<endl;

Pcls b;

b.pop();

cout<<"b: "<<b.foo()<<endl;

system ("pause");

return 0;

Билет 19  
1. Работа с объектами класса  
2. Базовый и производный класс. Использовать функциональный оператор, написать функцию нахождения количества отрицательных элементов массива.

Билет 20  
1 Конструкторы производных классов  
2 Среднее арифметическое массива, использовать шаблонную функцию, использовать производный и базовый

#include "stdafx.h"

#include <iostream>

using namespace std;

class A {

public:

int sum;

int chislo;

};

template <typename T>

class B : public A{

public:

T \*nmas;

B(int B)

{

chislo = B;

nmas = new T[chislo];

for (int i = 0; i < chislo; i++)

{

nmas[i] = i;

}

}

int Sum() {

for (int i = 0; i < chislo; i++)

{

sum += nmas[i];

}

return sum/chislo;

}

};

int main()

{

B <int>objINT(20);

B <float>objFLOAT(10);

cout << objFLOAT.Sum() << endl;

cout << objINT.Sum();

system("pause");

return 0;

}

БИЛЕТ 22  
1. Использование потоков для работы с памятью.  
2. Вычислить корни квадратного уравнения. Использовать базовый и производный классы. Использовать структурную обработку исключений.вы

#include "stdafx.h"

#include <iostream>

#include <math.h>

#include <Windows.h>

using namespace std;

class Bcls{

protected:

double a,b,c;

public:

Bcls(int x, int y, int z): a(x),b(y),c(z){}

float diskr(){

return sqrt(b\*b-(4\*a\*c));}

};

class Pcls: public Bcls{

double x1,x2;

public:

Pcls(int q, int w, int e): Bcls(q,w,e){}

void korn(float d){

x1=(b+d)/2;

x2=(b-d)/2;

cout<<"x1="<<x1<<" x2="<<x2;

}

};

int \_tmain(int argc, \_TCHAR\* argv[])

{int a,b,c;

cout<<"a\*x^2+b\*x+c=0"<<endl<<"Vvedite a: "; cin>>a;

cout<<"Vvedite b: "; cin>>b;

cout<<"Vvedite c: "; cin>>c;

Pcls y(a,b,c);

float d=y.diskr();

\_\_try{

y.korn(d);}

\_\_except(EXCEPTION\_EXECUTE\_HANDLER){

cout<<"otricatel'nyj diskrimenant";

}

system ("pause");

return 0;

}

Билет23  
1. Понятие класса  
2. Найти минимальное значение в массиве в базовом классе и максимальное в производном. Использовать this

#include "stdafx.h"

#include <iostream>

using namespace std;

class Min{

protected:

int n;

int s[100];

public:

Min(int k): n(k){cout<<"creating of"<<this<<endl;}

void push(){

for(int i=0;i<n;i++) {cout<<"Vvedite ["<<i<<"]: "; cin>>s[i];}}

int Pmin(){

int m=s[0];

for(int i=1;i<n;i++){if(s[i]<m) m=s[i];}

return m;}

};

class Max:public Min{

public:

Max(int r): Min(r){}

int Pmax(){

int m=s[0];

for(int i=1;i<n;i++){if(s[i]>m) m=s[i];}

return m;}

};

int \_tmain(int argc, \_TCHAR\* argv[])

{

Min a(5);

cout<<"massive 1"<<endl;a.push();

cout<<"min element massiva 1: "<<a.Pmin()<<endl;

Max b(5);

cout<<endl<<"massive 2"<<endl;b.push();

cout<<"max element massiva 2: "<<b.Pmax();

system ("pause");

return 0;

}

Билет 24  
1. Структурная обработка исключений.  
2. Увеличить поле класса на 10, используя переопределение оператора инкремент в префиксной и постфиксной формах.

#include "stdafx.h"

#include <iostream>

using namespace std;

class cls{

protected:

int x;

public:

cls(int a): x(a){}

};

class Pcls: public cls{

public:

Pcls(int a):cls(a){}

Pcls&operator ++(){

x\*=10;

return \*this;

}

Pcls operator ++ (int){

Pcls tmp=\*this;

(this->x)\*=10;

return tmp;

}

friend ostream& operator<<(ostream&os,const Pcls o);

};

ostream& operator<<(ostream&os,const Pcls o){

os<<o.x<<endl;

return os;

}

int \_tmain(int argc, \_TCHAR\* argv[])

{

Pcls p(5);

cout<<p++;

cout<<p;

system("pause");

return 0;

25 билет  
1. Преобразование типов. Динамическое приведение типа.  
2. Найти среднее арифметическое двух полей используя дружественную функцию в базовом  
и производном классе

#include "stdafx.h"

#include <iostream>

using namespace std;

class A {

friend int Sum(A &);

private:

int b;

int a;

public:

A() {}

A(int a, int b) { this->a = a; this->b = b; }

};

int Sum(A &a) {

return (a.a + a.b )/ 2;

}

class B : public A

{

friend int Sum(B &);

private:

int c;

int d;

public:

B(int c, int d){ this->c = c; this->d = d; }

};

int Sum(B &b) {

return (b.c + b.d) / 2;

}

int main()

{

A objA(12, 24);

B objB(24, 32);

cout << Sum(objA) << endl;

cout << Sum(objB);

system("pause");

return 0;

}

26 билет  
1.Стандартная библиотека шаблонов. Алгоритмы  
2. Производный класс со вложенным конструктором, который считает сумму элементов массива

#include "stdafx.h"

#include <iostream>

using namespace std;

class Bcls{

protected:

int n;

int mas[100];

public:

Bcls(int m):n(m){

for(int i=0;i<n;i++){

cout<<"Vvedite ["<<i<<"]: "; cin>>mas[i];}cout<<endl<<endl;}

};

class Pcls: public Bcls{

public:

Pcls(int k):Bcls(k){

int s=0;

for(int i=0;i<k;i++){s+=mas[i];}

cout<<"Summa elementov massiva: "<<s;}

};

int \_tmain(int argc, \_TCHAR\* argv[])

{Pcls a(5);

system("pause");

return 0;

}

28 билет   
1 указатель this   
2 создать класс который находит сумму полей класса, базовый класс должен быть абстрактным

#include "stdafx.h"

#include <iostream>

using namespace std;

class Bcls{

protected:

double a,b,c;

public:

Bcls(double x, double y, double z): a(x), b(y), c(z){}

virtual void sum()=0;

};

class Pcls: public Bcls{

public:

Pcls(double x, double y, double z):Bcls(x,y,z){}

void sum(){

double s=0;

s=a+b+c;

cout<<"Summa polej ravna: "<<s<<endl;

}

};

int \_tmain(int argc, \_TCHAR\* argv[])

{Pcls a(1,1,1);

a.sum();

system("pause");

return 0;

}

Билет 30.1) Написать класс с вложенным конструктором для нахождения суммы элементов массива в базовом и произведения элементов массива в производном. (Хз зачем, но ему оочень нужен был именно вложенный конструктор)  
2) Абстрактный класс и чисто виртуальные функции

[#include](https://vk.com/im?sel=138446909&st=%23include)<iostream>   
using namespace std;   
class Scls   
{   
protected:   
int ms[4];   
public:   
Scls(){}   
Scls(int a,int b,int c,int d)   
{   
ms[0] = a;   
ms[1] = b;   
ms[2] = c;   
ms[3] = d;   
}   
~Scls() {}   
  
int sum\_elem()   
{   
int sum = 0;   
for (int i = 0; i < 4; i++)   
{   
sum += ms[i];   
}   
return sum;   
}   
int getelem(int i)   
{   
return ms[i];   
}   
  
};   
  
class Pcls :public Scls   
{   
Scls tmp;   
public:   
Pcls(){}   
Pcls(int a, int b, int c, int d) : tmp(a, b, c, d){}   
~Pcls() {}   
int mult\_elem()   
{   
int mult = 1;   
for (int i = 0; i < 4; i++)   
{   
mult = mult\*tmp.getelem(i);   
}   
return mult;   
}   
  
};   
  
int main()   
{   
Scls obj1(2,2,3,4);   
Pcls obj2(4,2,6,7);   
  
int sum = obj1.sum\_elem();   
cout « "sum of elements=" « sum « endl;   
  
int mult = obj2.mult\_elem();   
cout « "multiplication of elements=" « mult « endl;   
system("pause");   
}

Билет 31   
1. Конструктор. Отличие от других функций   
2. Написать класс для расчета среднего арифметического полей базового и производного классов. Базовый класс должен быть абстрактным

32 билет  
1. Перегрузка методов  
2. Переопределение инкримента. Базовый производный классы

[#include](https://vk.com/im?sel=138446909&st=%23include)<iostream>   
using namespace std;   
class Base   
{   
protected:   
int a;   
int b;   
public:   
Base(int a1, int b1) { a = a1, b = b1; }   
Base operator++(int)   
{   
  
a++;   
b++;   
return \*this;   
}   
int get\_a()   
{   
return a;   
}   
int get\_b()   
{   
return b;   
}   
};   
class Son : public Base   
{   
int c;   
public:   
Son(int a1, int b1, int c1) : Base(a1, b1), c(c1) {}   
Son operator++(int)   
{   
a++;   
b++;   
c++;   
return \*this;   
}   
int get\_c()   
{   
return c;   
}   
  
};   
  
  
int main()   
{   
Base obj1(5, 6);   
Son obj2(2, 3, 4);   
cout « "a=" « obj1.get\_a() « endl « "b=" « obj1.get\_b() « endl;   
obj1++;   
cout « "using operator ++" « endl « "a=" « obj1.get\_a() « endl « "b=" « obj1.get\_b() « endl;   
cout « "a=" « obj2.get\_a() « endl « "b=" « obj2.get\_b() « endl « "c=" « obj2.get\_c() « endl«endl;   
obj2++;   
cout « "using operator ++" « endl « "a=" « obj2.get\_a() « endl « "b=" « obj2.get\_b() « endl « "c=" « obj2.get\_c() « endl;   
system("pause");   
}