**Содержание**

[Постановка задачи и общие требования 2](#_Toc99236202)

[Авторское понимание задачи 3](#_Toc99236203)

[Текст программы 3](#_Toc99236205)

[} 13](#_Toc99236206)

[Результаты тестирования 13](#_Toc99236207)

[Вывод 13](#_Toc99236208)

# Постановка задачи и общие требования

Создать класс Octal (Класс для работы с беззнаковыми целыми восьмеричными числами, использует для представления числа массив из 100 элементов типа unsigned char, каждый элемент которого является восьмеричной цифрой. Младшая цифра имеет меньший индекс (единицы - в нулевом элементе массива).) и класс Hex (Класс для работы с беззнаковыми целыми шеснадцатеричными числами, используя для представления числа массив из 100 элементов типа unsigned char, каждый из которых является шеснадцатеричной цифрой. Младшая цифра имеет меньший индекс.). Общие требования:

В каждом упражнении требуется реализовать в том или ином виде определение нового класса.

Создать базовый класс Array с полями: массив типа unsigned char. Максимально возможный размер массива задается статической константой. Реализовать конструктор инициализации, задающий количество элементов и начальное значение (по умолчанию 0). Реализовать метод доступа к элементу, перегрузив операцию индексирования []. При этом должна выполняться проверка индекса на допустимость.

Реализовать в классе Array виртуальную функцию поэлементного сложения массивов. Реализовать два класса, переопределив виртуальную функцию сложения. Вызывающая программа должна продемонстрировать все варианты вызова виртуальных функций.

Для выполнения работы необходимо:

* формализовать задачу;
* составить описание класса;
* составить алгоритм для каждого метода, описанного в классе, и определить все методы;
* создать объекты описанного типа;
* разработать набор тестов, на которых будет проверяться программа;
* продемонстрировать работу программы на наборе тестов;
* оформить работу и отчитаться по ней.

# Авторское понимание задачи

# Для решения данной задачи необходимо создать базовый класс Array, от которого наследуют два произвольных класса Octal и Hex. Соответственно, в базовом классе создаем виртуальные методы преобразования из int в char(и наоборот) , а также повышение следующего разряда. Кроме этого нужно перегрузить операторы ввода и вывода (в базовом классе) и переопределить виртуальные методы в произвольных классах.

# Текст программы

Array.h

#pragma once

#include <iostream>

using namespace std;

class Array   
{

protected:

static const int max\_dlina = 100;

unsigned char\* arv;

int dlina;

public:

Array();

Array(int ra3mer);

~Array();

int length() { return dlina; }

virtual Array& operator +(Array& perv);

unsigned char& operator [](const int index);

friend istream& operator>>(istream& in, Array& t);

friend ostream& operator<<(ostream& out, Array& t);

virtual int charToint(char od);

virtual char intTochar(int dv);

unsigned char getMass(int index) { return arv[index]; }

void setMass(int index, char od) { arv[index] = od; }

int Lastind();

virtual void Dobav(const int index);

};

Array.cpp

#include "Array.h"

Array::Array() : dlina(0), arv(NULL)   
{

}

Array::Array(int ra3mer) : dlina(ra3mer)   
{

arv = new unsigned char[ra3mer];

for (int i = 0; i < ra3mer; i++)   
{

arv[i] = 0;

}

}

Array::~Array()   
{

delete[] arv;

}

int Array::Lastind()   
{

int chislo = 99;

while (chislo >= 0)   
{

if (arv[chislo] != '\0')   
{

break;

}

chislo--;

}

return chislo;

}

Array& Array::operator+(Array& perv)   
{

int pervoe, vtoroe = 0;

int suMma = 0;

do   
{

pervoe = perv.Lastind();

vtoroe = Lastind();

if (pervoe == vtoroe)   
{

for (int pol=0; pol <= pervoe; pol++)   
{

suMma = charToint(getMass(pol)) + perv.charToint(perv[pol]);

if (suMma >= 16)   
{

setMass(pol, intTochar(suMma - 16));

Dobav(pol + 1);

}

else   
{

setMass(pol, intTochar(suMma));

}

}

}

else   
{

if (pervoe > vtoroe)   
{

for (int pol = vtoroe + 1; pol <= pervoe; pol++)   
{

setMass(pol, '0');

}

}

else   
{

if (pervoe < vtoroe)   
{

for ( int pol = pervoe + 1; pol <= vtoroe;pol++)   
{

perv.setMass(pol, '0');

}

}

}

}

}   
while ((pervoe > vtoroe) || (pervoe < vtoroe));

return \*this;

}

unsigned char& Array::operator[](const int index)   
{

unsigned char t = '0';

if (index<0 || index>max\_dlina)   
{

return t;

}

return arv[index];

}

void Array::Dobav(const int index)   
{

int sledyuch;

if (getMass(index) == '0')   
{

setMass(index, '0');

}

sledyuch = charToint(getMass(index));

if (arv[index] == 'F')   
{

setMass(index, '0');

Dobav(index + 1);

}

else   
{

setMass(index, intTochar(sledyuch + 1));

}

}

int Array::charToint(char od)   
{

switch (od)   
{

case ('0'): return 0; break;

case ('1'): return 1; break;

case ('2'): return 2; break;

case ('3'): return 3; break;

case ('4'): return 4; break;

case ('5'): return 5; break;

case ('6'): return 6; break;

case ('7'): return 7; break;

case ('8'): return 8; break;

case ('9'): return 9; break;

case ('A'): return 10; break;

case ('B'): return 11; break;

case ('C'): return 12; break;

case ('D'): return 13; break;

case ('E'): return 14; break;

case ('F'): return 15; break;

default: return -1; break;

}

}

char Array::intTochar(int dv)   
{

switch (dv)

{

case (0): return '0'; break;

case (1): return '1'; break;

case (2): return '2'; break;

case (3): return '3'; break;

case (4): return '4'; break;

case (5): return '5'; break;

case (6): return '6'; break;

case (7): return '7'; break;

case (8): return '8'; break;

case (9): return '9'; break;

case (10): return 'A'; break;

case (11): return 'B'; break;

case (12): return 'C'; break;

case (13): return 'D'; break;

case (14): return 'E'; break;

case (15): return 'F'; break;

default: return -1; break;

}

}

istream& operator>>(istream& in, Array& t)   
{

int schetchik = 0, elem = 0, obch = 0; char chislo;

for (int schetchik = 0; schetchik < t.max\_dlina; schetchik++)  
{

t.arv[schetchik] = 0;

}   
in >> chislo;

for (schetchik = 99; schetchik >= 0; schetchik--)

{

if (t.charToint(chislo)!=-1)  
{

t[schetchik] = chislo;

}   
else if (t.charToint(chislo)==-1)

{

break;

}

chislo = getchar();

}

if (schetchik != -1)

{

elem = 99 - schetchik - 1;

obch = 99;

for (; elem >= 0; elem--)

{

t[elem] = t[obch];

t[obch] = '\0';

obch--;

}

}

return in;

}

ostream& operator<<(ostream& out, Array& t)

{

int schet = t.Lastind();

while (schet >= 0) {

if (t.arv[schet] == '0')  
{

schet--;

}   
else  
{

break;

}   
}

for (; schet >= 0; schet--)  
{

out << t[schet];

}   
return out;

}

Octal.h

#pragma once

#include "Array.h"

class Octal :

public Array

{

public:

Octal();

Octal(int ra3mer);

~Octal();

virtual Octal& operator +(Octal& perv);

virtual int charToint(char od);

virtual char intTochar(int dv);

virtual void Dobav(const int index);

};

Octal.cpp

#include "Octal.h"

Octal::Octal() :Array()   
{

}

Octal::Octal(int ra3mer) : Array(ra3mer)   
{

}

Octal::~Octal()   
{

delete[]arv;

}

Octal& Octal::operator+(Octal& perv)   
{

int pervoe, vtoroe = 0;

int suMma = 0;

do   
{

pervoe = perv.Lastind();

vtoroe = Lastind();

if (pervoe == vtoroe)   
{

for (int pol = 0; pol <= pervoe; pol++)   
{

suMma = charToint(getMass(pol)) + perv.charToint(perv[pol]);

if (suMma >= 8)   
{

setMass(pol, intTochar(suMma - 8));

Dobav(pol + 1);

}

else   
{

setMass(pol, intTochar(suMma));

}

}

}

else   
{

if (pervoe > vtoroe)   
{

for (int pol = vtoroe + 1; pol <= pervoe; pol++)   
{

setMass(pol, '0');

}

}

else   
{

if (pervoe < vtoroe)   
{

for (int pol = pervoe + 1; pol <= vtoroe; pol++) {

perv.setMass(pol, '0');

}

}

}

}

}   
while ((pervoe > vtoroe) || (pervoe < vtoroe));

return \*this;

}

int Octal::charToint(char od)   
{

switch (od)   
{

case ('0'): return 0; break;

case ('1'): return 1; break;

case ('2'): return 2; break;

case ('3'): return 3; break;

case ('4'): return 4; break;

case ('5'): return 5; break;

case ('6'): return 6; break;

case ('7'): return 7; break;

default: return -1; break;

}

}

char Octal::intTochar(int dv)   
{

switch (dv)

{

case (0): return '0'; break;

case (1): return '1'; break;

case (2): return '2'; break;

case (3): return '3'; break;

case (4): return '4'; break;

case (5): return '5'; break;

case (6): return '6'; break;

case (7): return '7'; break;

default: return -1; break;

}

}

void Octal::Dobav(const int index)   
{

int sledyuch;

if (getMass(index) == '0')   
{

setMass(index, '0');

}

sledyuch = charToint(getMass(index));

if (arv[index] == 'F')   
{

setMass(index, '0');

Dobav(index + 1);

}

else   
{

setMass(index, intTochar(sledyuch + 1));

}

}

Hex.h

#pragma once

#include "Array.h"

class Hex :

public Array

{

public:

Hex();

Hex(int ra3mer);

~Hex();

virtual Hex& operator+(Hex& perv);

virtual int charToint(char od);

virtual char intTochar(int dv);

virtual void Dobav(const int index);

};

Hex.cpp

#include "Hex.h"

Hex::Hex() :Array()   
{

}

Hex::Hex(int ra3mer) : Array(ra3mer)   
{

}

Hex::~Hex()   
{

delete[]arv;

}

Hex& Hex::operator+(Hex& perv)   
{

int pervoe, vtoroe = 0;

int suMma = 0;

do   
{

pervoe = perv.Lastind();

vtoroe = Lastind();

if (pervoe == vtoroe)   
{

for (int pol = 0; pol <= pervoe; pol++)   
{

suMma = charToint(getMass(pol)) + perv.charToint(perv[pol]);

if (suMma >= 16)   
{

setMass(pol, intTochar(suMma - 16));

Dobav(pol + 1);

}

else   
{

setMass(pol, intTochar(suMma));

}

}

}

else   
{

if (pervoe > vtoroe)   
{

for (int pol = vtoroe + 1; pol <= pervoe; pol++)   
{

setMass(pol, '0');

}

}

else   
{

if (pervoe < vtoroe)   
{

for (int pol = pervoe + 1; pol <= vtoroe; pol++) {

perv.setMass(pol, '0');

}

}

}

}

}   
while ((pervoe > vtoroe) || (pervoe < vtoroe));

return \*this;

}

int Hex::charToint(char od)   
{

switch (od)   
{

case ('0'): return 0; break;

case ('1'): return 1; break;

case ('2'): return 2; break;

case ('3'): return 3; break;

case ('4'): return 4; break;

case ('5'): return 5; break;

case ('6'): return 6; break;

case ('7'): return 7; break;

case ('8'): return 8; break;

case ('9'): return 9; break;

case ('A'): return 10; break;

case ('B'): return 11; break;

case ('C'): return 12; break;

case ('D'): return 13; break;

case ('E'): return 14; break;

case ('F'): return 15; break;

default: return -1; break;

}

}

char Hex::intTochar(int dv)   
{

switch (dv)

{

case (0): return '0'; break;

case (1): return '1'; break;

case (2): return '2'; break;

case (3): return '3'; break;

case (4): return '4'; break;

case (5): return '5'; break;

case (6): return '6'; break;

case (7): return '7'; break;

case (8): return '8'; break;

case (9): return '9'; break;

case (10): return 'A'; break;

case (11): return 'B'; break;

case (12): return 'C'; break;

case (13): return 'D'; break;

case (14): return 'E'; break;

case (15): return 'F'; break;

default: return -1; break;

}

}

void Hex::Dobav(const int index)   
 {

int sledyuch;

if (getMass(index) == '0')   
{

setMass(index, '0');

}

sledyuch = charToint(getMass(index));

if (arv[index] == 'F')   
{

setMass(index, '0');

Dobav(index + 1);

}

else   
{

setMass(index, intTochar(sledyuch + 1));

}

}

Main.cpp

#include "Array.h"

#include "Octal.h"

#include "Hex.h"

#include <iostream>

#include <conio.h>

using namespace std;

int main()   
{

Octal proverka\_pervoe(6), proverka\_vtoroe(6);

Hex prov\_perv(6), prov\_vtor(6);

cout << "Vvod chisel iz 8-richnoi systemi: ";

cin >> proverka\_pervoe;

cin >> proverka\_vtoroe;

cout << "Rezyltat v 8-richnoi sisteme: " <<proverka\_pervoe+proverka\_vtoroe<< endl;

cout << "Vvod chisel iz 16-richnoi systemi: ";

cin >> prov\_perv;

cin >> prov\_vtor;

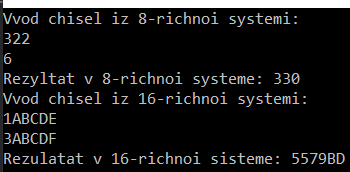
cout << "Rezyltat v 16-richnoi sisteme: " << prov\_perv + prov\_vtor << endl;

\_getch();

return 0;

# }

# Результаты тестирования



# Вывод

Результаты работы программы совпадают с ожидаемыми результатами программа работает правильно.