**1**

**Цель**

* Изучение средств языка С++ для определения новых типов данных через механизм классов.
* Получение навыков в написании программ с использованием классов.

**А. Минимальное задание**

**Постановка задачи**

1.Составить программу с использованием классов для реализации "бегущей" точки из левого верхнего угла экрана в нижний правый угол экрана по строкам. Цвет точки выбирается произвольно. Продемонстрировать работу двух конструкторов: обычного конструктора и конструктора по умолчанию. Сделать выводы.

2. **Для классов: class Point {int x,y;**

**public:**

#### Point (int a, int b) {x=a; y=b; };

**…..**

**};**

**class Rect {Point lt, rb;**

**public:**

**Rect (Point a, Point b):lt(a), rb(b){ };**

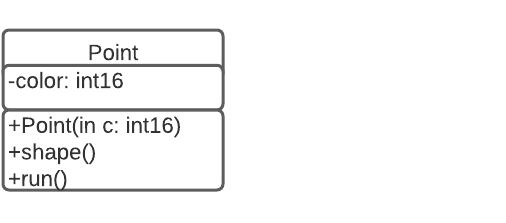
**void draw(void);**

**};**

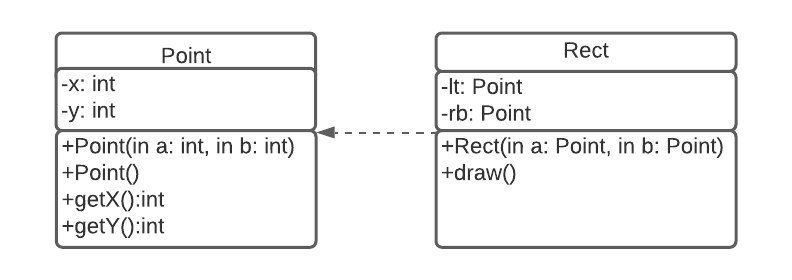
доопределить соответствующие член-функции таким образом, чтобы, объявив в main-программе объект класса Rect, вызовом функции draw() можно было бы вычертить на экране прямоугольник с указанием заданных размеров с помощью псевдографических символов. Сделать выводы.

**Структурная схема классов**

**1.**



**2.**



#### Исходные тексты программ

**1.**

#include <iostream>

#include <Windows.h>

class Point { short color;

public: Point(short c);

Point();

void run(); //начать отрисовку };

// **реализация**

Point::Point(short c) : color(c) { }

Point::Point() : Point(0xFFFFFF) { }

void Point::run() {

HANDLE handle = GetStdHandle(STD\_OUTPUT\_HANDLE);

SetConsoleTextAttribute(handle, color);

//получаем размеры окна консоли

CONSOLE\_SCREEN\_BUFFER\_INFO inf;

GetConsoleScreenBufferInfo(handle, &inf);

int width = inf.srWindow.Right - inf.srWindow.Left + 1;

int height = inf.srWindow.Bottom - inf.srWindow.Top + 1;

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

for (int j = 0; j < width; j++) {

COORD p = { j, i }; //координата точки

SetConsoleCursorPosition(handle, p);

std::cout << '\*';

Sleep(1);

system("CLS"); }

}

int main() {

Point p1;//обычная точка

p1.run();

Point p2(0x00FF00);//зелёная точка

p2.run();

}

**2.**

#include <iostream>

#include <Windows.h>

class Point { int x, y;

public: Point(int a, int b) { x = a; y = b; };

Point() : Point(0, 0) { }

int getX() { return x; }

int getY() { return y; } };

class Rect { Point lt, rb;

public: Rect(Point a, Point b) : lt(a), rb(b) { }

void draw() {

HANDLE handle = GetStdHandle(STD\_OUTPUT\_HANDLE);

COORD p = { lt.getX(), lt.getY() };

SetConsoleCursorPosition(handle, p);

std::cout << static\_cast<char>(218);//левый верхний угол

p.Y = rb.getY();

SetConsoleCursorPosition(handle, p);

std::cout << static\_cast<char>(192);//левый нижний угол

p.X = rb.getX();

SetConsoleCursorPosition(handle, p);

std::cout << static\_cast<char>(217);//правый нижний угол

p.Y = lt.getY();

SetConsoleCursorPosition(handle, p);

std::cout << static\_cast<char>(191);//левый верхний угол

int width = rb.getX() - lt.getX();

int height = rb.getY() - lt.getY();

for (int i = 1; i < width; i++) {//рисуем горизонтальные стороны

p.X = lt.getX() + i; p.Y = lt.getY();

SetConsoleCursorPosition(handle, p);

std::cout << static\_cast<char>(196);

p.Y = rb.getY();

SetConsoleCursorPosition(handle, p); std::cout << static\_cast<char>(196);

}

for (int i = 1; i < height; i++) { //рисуем вертикальные стороны

p.X = lt.getX();

p.Y = lt.getY() + i;

SetConsoleCursorPosition(handle, p);

std::cout << static\_cast<char>(179);

p.X = rb.getX();

SetConsoleCursorPosition(handle, p);

std::cout << static\_cast<char>(179);

}

} };

int main() {

Rect r(Point(2, 2), Point(50, 25));

r.draw();

getchar(); }

**Результат работы программы №2**

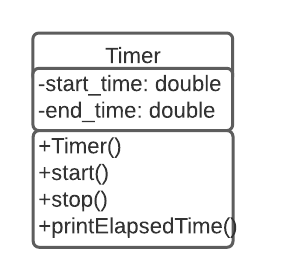


**Б. Основное задание**

**Постановка задачи**

Создайте класс для имитации секундомера. Члены класса: переменные типа double для хранения начального и конечного значения времени. Обязательны функции- члены класса для инициализации секундомера (присваивает переменным нулевые значения), для запуска и остановки секундомера, для вывода на экран величины истекшего времени.

**Структурная схема класса**



**Исходный текст программы**

class Timer {private: double start\_time, end\_time;

public: Timer();

void start();

void stop();

void printElapsedTime(); };

#include "Timer.h"

#include <ctime>

//реализация

Timer::Timer() : start\_time(0.), end\_time(0.) { }

void Timer::start() { start\_time = (double)clock() / 1000.; }

void Timer::stop() { end\_time = (double)clock() / 1000.; }

void Timer::printElapsedTime() { std::cout << end\_time - start\_time << "s" << std::endl; }

#include "Timer.h"

#include <Windows.h>

int main() { Timer t;

t.start();

Sleep(15);

t.stop();

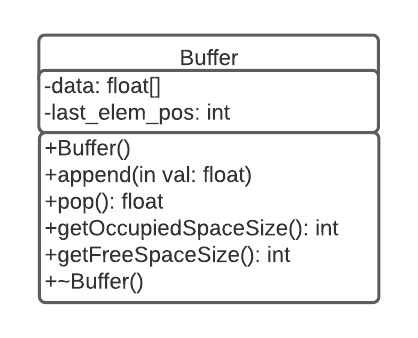
t.printElapsedTime(); }

**Результат работы программы**

0.015s

**В. Дополнительное задание**

**Структурная схема класса**



**Исходный текст программы**

class Buffer {private: float \*data;

int last\_elem\_pos; //позиция последнего занесенного элемента

public: Buffer();

void append(float val);

float pop(); //получение элемента

int getOccupiedSpaceSize();

int getFreeSpaceSize();

~Buffer(); };

#include "Buffer.h"

const int max\_buff\_size = 1024;

// **реализация**

Buffer::Buffer() {

data = new float[max\_buff\_size];

last\_elem\_pos = 0; }

void Buffer::append(float val){

data[last\_elem\_pos % max\_buff\_size] = val;

if (++last\_elem\_pos > max\_buff\_size\*2) last\_elem\_pos -= max\_buff\_size; }

float Buffer::pop() {

if (!last\_elem\_pos) throw "Empty buffer";

float val = data[0];

for (int i = 1; i < last\_elem\_pos; ++i) //сдвигаем элементы буфера назад

data[(i - 1) % max\_buff\_size] = data[i % max\_buff\_size];

--last\_elem\_pos;

return val; }

int Buffer::getOccupiedSpaceSize() { return last\_elem\_pos <= max\_buff\_size ? last\_elem\_pos : max\_buff\_size; }

int Buffer::getFreeSpaceSize() { int difference = max\_buff\_size - last\_elem\_pos;

return difference >= 0 ? difference : 0; }

Buffer::~Buffer() { delete[] data; }

#include <iostream>

#include "Buffer.h"

int main() { Buffer buf;

buf.append(1.f);

buf.append(2.f);

buf.append(3.f);

std::cout << "Size: " << buf.getOccupiedSpaceSize() << std::endl;

std::cout << "Space left: " << buf.getFreeSpaceSize() << std::endl;

std::cout << "First element: " << buf.pop() << std::endl;

std::cout << "Second element: " << buf.pop() << std::endl;

std::cout << "Third element: " << buf.pop() << std::endl;

std::cout << "Size: " << buf.getOccupiedSpaceSize() << std::endl;

std::cout << "Space left: " << buf.getFreeSpaceSize() << std::endl; }

**Результат работы программы**

Size: 3

Space left: 1021

First element: 1

Second element: 2

Third element: 3

Size: 0

Space left: 1024

**Выводы**

* Классы используются, когда требуется множество объектов, обладающих схожим функционалом.
* Инкапсуляция и сокрытие данных помогают избежать ошибок в работе объектов.
* Классы облегчают создание программ тем, что разбивают разработку на множество отдельных частей.

**2.**

**Цель**

* Изучение механизма перегрузки функций.
* Получение навыков в написании программ с использованием перегрузки конструкторов.
* Изучение механизма перегрузки операций.
* Получение навыков в написании программ с использованием перегрузки операций для объектов, введенных пользователем.

**2.1.**

**А. Минимальное задание**

**Постановка задачи**

Создайте функцию **prompt ( )** для выдачи подсказки пользователю и ожидания ввода данного. Перегрузите функцию **prompt ( )** так, чтобы она осуществляла ввод данных типов **int, double, long**.

**Исходный текст программы**

#include <iostream>

void prompt(int \*val);

void prompt(long \*val);

void prompt(double \*val);

int main() { int a1;

long a2;

double a3;

prompt(&a1);

prompt(&a2);

prompt(&a3);

std::cout << "You entered: " << a1 << '\t' << a2 << '\t' << 3; }

void prompt(int \*val) { std::cout << "Enter integer number: "; std::cin >> \*val; std::cout << std::endl; }

void prompt(long \*val) { std::cout << "Enter long integer number: "; std::cin >> \*val; std::cout << std::endl; }

void prompt(double \*val) { std::cout << "Enter floating number: "; std::cin >> \*val; std::cout << std::endl;}

**Результат работы программы**

Enter integer number: 100

Enter long integer number: 10000

Enter floating number: 56.34

You entered: 100 10000 56.34 Вывод: Area of triangle = 105

**Б. Основное задание**

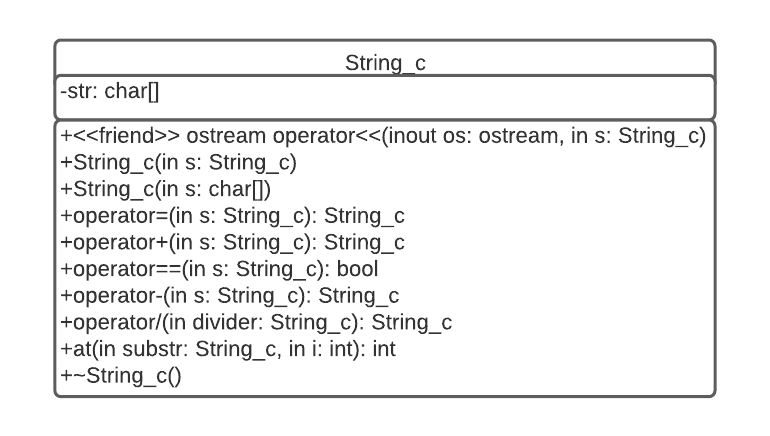
**Постановка задачи**

Создать класс “Строка\_Си”, определить для него операции:

* 1. слияния строк (операция + );
  2. сравнение строк на равенство (операция = = );
  3. удаления из строки подстроки (операция - );
  4. удаления из строки всех символов, содержащихся в строке-делителе (операция / );
  5. копирования строки
  6. функцию at (char \*substr, char \*dest, int i ), возвращающую номер позиции i-того вхождения строки substr в строкуdest.

Продемонстрируйте выполнение операций с объектами этого класса.

**Структурная схема класса**



**Исходный текст программы**

#include <ostream>

class String\_c { friend std::ostream& operator<<(std::ostream& os, String\_c s);

private: char \*str;

public: String\_c(const String\_c &s); //конструктор копирования

String\_c(const char \*s);

String\_c& operator=(const String\_c &s); //копирование

String\_c operator+(const String\_c &s);

bool operator==(const String\_c &s);

String\_c operator-(const String\_c &s);

String\_c operator/(const String\_c &divider);

//возвращает номер i-го вхождения substr или -1

int at(String\_c& substr, int i);

~String\_c(); };

#include "String\_c.h"

#include <iostream>

#include <string>

#include <vector>

#include <algorithm>

// **реализация**

String\_c::String\_c(const char \*s){ int size = strlen(s);

str = new char[size + 1];

strcpy(str, s); }

String\_c::String\_c(const String\_c &s) : String\_c(s.str) { }

String\_c& String\_c::operator=(const String\_c &s) { delete[] str;

str = new char[strlen(s.str) + 1];

strcpy(str, s.str);

return \*this; }

String\_c String\_c::operator+(const String\_c &s) {

int len1 = strlen(str), len2 = strlen(s.str);

String\_c result(" ");

delete[] result.str;

result.str = new char[len1 + len2 + 1];

strcpy(result.str, str);

strcpy(result.str + len1, s.str);

return result; }

bool String\_c::operator==(const String\_c &s) { return strcmp(str, s.str) == 0; }

String\_c String\_c::operator-(const String\_c &s) {

char \*temp = new char[strlen(str) + 1];

strcpy(temp, str);

//получаем позицию вхождения вычитателя в вычитаемое

char \*entry\_pos = strstr(temp, s.str);

if (!entry\_pos) { String\_c result(str); return result; }

strcpy(entry\_pos, entry\_pos + strlen(s.str)); //сжимаем строку

String\_c result(temp);

delete[] temp;

return result; }

String\_c String\_c::operator/(const String\_c &divider) {

char \*temp = new char[strlen(str) + 1];

strcpy(temp, str);

std::vector<char> deleted\_chars;

for (char \*i = divider.str; \*i != '\0'; i++) { //проходимся по всем знакам делителя

if (std::find(deleted\_chars.begin(), deleted\_chars.end(), \*i) != deleted\_chars.end()) //символ уже удалён

continue;

for (char \*j = temp; \*j != '\0'; j++) { //удаляем символ из строки

if (\*j == \*i) { //встретили удаляемый символ

strcpy(temp, j + 1); //сдвигаем "хвост" строки влево, стирая удаляемый символ

}

} deleted\_chars.push\_back(\*i);

}

String\_c result(temp);

delete[] temp;

return result; }

int String\_c::at(String\_c &substr, int i) { int entry\_n = 0;

char \*temp = str;

int substr\_len = strlen(substr.str);

while (true) {

if (entry\_n == i) break;

char \*entry = strstr(temp, substr.str);

if (!entry) break;

entry\_n++;

temp = entry + substr\_len;

}

if (entry\_n != i) return -1;

return temp - str - substr\_len + 1; }

String\_c::~String\_c() { delete[] str; }

std::ostream& operator<<(std::ostream& os, String\_c s) { os << s.str; return os; }

#include <iostream>

#include "String\_c.h"

int main() {

String\_c s1("abcdabceabcf");

String\_c s2("abc");

std::cout << "s1 = " << s1 << std::endl;

std::cout << "s2 = " << s2 << std::endl;

std::cout << "s1 - s2 = " << s1 - s2 << std::endl;

std::cout << "s1 / s2 == " << s1 / s2 << std::endl;

std::cout << "s3 = s1 + s2"; std::cout << std::endl;

String\_c s3 = s1 + s2;

std::cout << "s3 = " << s3 << std::endl;

std::cout << "s3 := s2" << std::endl;

s3 = s2;

std::cout << "s3 = " << s3 << std::endl;

std::cout << std::boolalpha << "s3 == s2? - " << (s3 == s2) << std::endl;

std::cout << std::boolalpha << "s1 == s2? - " << (s1 == s2) << std::endl;

std::cout << "Position of 3rd entry of s2 into s1 = " << s1.at(s2, 3); }

**Результат работы программы**

s1 = abcdabceabcf

s2 = abc

s1 - s2 = dabceabcf

s1 / s2 == f

s3 = s1 + s2

s3 = abcdabceabcfabc

s3 := s2

s3 = abc

s3 == s2? - true

s1 == s2? - false

Position of 3rd entry of s2 into s1 = 9

**2.2**

**Постановка задачи**

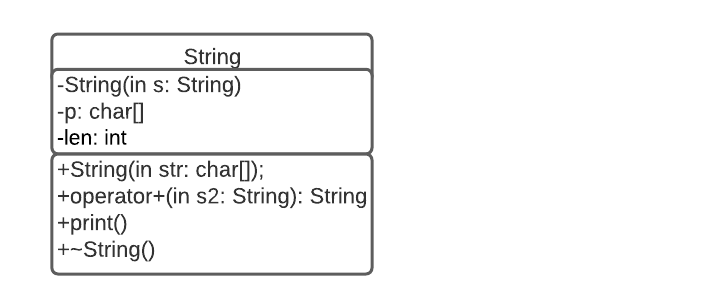
Определите класс "строка" с двумя член-данными: **char \*p** и **int len,** где **char\*p**- указатель на строку, а **int len** - длина строки.

Для данного класса определить:

* один конструктур и деструктор( предусмотреть выдачу сообщения об их работе);
* переопределить операцию "+" для сложения объектов класса;
* определить член функцию, выводящую строку на экран.

Напишите демонстрационную версию программы, которая создает 2 объекта класса "строка" (первый объект содержит строку с фамилией, второй объект содержит имя), выводит строки этих объектов на экран, складывает эти объекты и выводит на экран результирующую строку обоих.

**Структурная схема класса**



**Исходный текст программы**

class String { private:

String(String &s); //запрещаем копирование

char \*p;

int len;

public: String(const char \*str);

String operator+(const String &s2);

void print();

~String(); };

#include "String.h"

#include <cstring>

#include <iostream>

// **реализация**

String::String(const char \*str) {

std::cout << "String counstructor is working!" << std::endl;

len = strlen(str);

p = new char[len + 1];

strcpy(p, str); }

String::String(String &s) : String(s.p) { }

String String::operator+(const String &s2) {

char \*temp = new char[len + s2.len + 1];

strcpy(temp, p);

strcat(temp, s2.p);

String result(temp);

delete[] temp;

return result; }

void String::print() { std::cout << p << std::endl; }

String::~String() {

std::cout << "String destructor is working!" << std::endl;

delete[] p; }

#include "String.h"

int main() {

String s1("Pavel ");

String s2("Alexandrov");

s1.print();

s2.print();

(s1 + s2).print(); }

**Результат работы программы**

String counstructor is working!

String counstructor is working!

Pavel

Alexandrov

String counstructor is working!

String counstructor is working!

String destructor is working!

Pavel Alexandrov

String destructor is working!

String destructor is working!

String destructor is working!

**Вывод**

* Перегрузка нужна для того, чтобы функцию можно было вызывать для объектов разных типов данных.
* Перегрузка конструкторов помогает по-разному настраивать объект перед созданием.
* Перегрузка операций улучшает качество кода.