681.3×5(07)
М545

№ 2845-1

МИНИСТЕРСТВО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ТАГАНРОГСКИЙ ГОСУДАРСТВЕННЫЙ РАДИОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

КАФЕДРА ТЕОРЕТИЧЕСКИХ ОСНОВ РАДИОТЕХНИКИ

МЕТОДИЧЕСКИЕ УКАЗАНИЯ К ЛАБОРАТОРНЫМ РАБОТАМ ПО КУРСУ

ИНФОРМАТИКИ
(Часть 1)

Для студентов радиотехнических специальностей всех форм обучения

РТФ ТАГАНРОГ 2010

УДК 681.3×5(07.07)

Составитель: М.Н. Максимов

Методические указания к лабораторным работам по курсу
“Информатика” (Часть 1). Таганрог: Изд-во ТРТУ,
2000. 30 с.

Приведены методические указания к лабораторным работам
1-11 по курсу “Информатика”.

Методические указания предназначены для студентов радиотехнических специальностей всех форм обучения.

Библиогр.: 8 назв.

Рецензент А.Л. Черниховский, канд. техн. наук, доцент кафедры ТОР ТРТУ.

Максимов Михаил Николаевич

Введение

Настоящее руководство предназначено для студентов радиотехнических специальностей безотрывных форм обучения. Руководство содержит описание лабораторных работ 1-11.

Основные теоретические положения, необходимые для выполнения лабораторных работ, приведены в работах 1-11.

Общие требования к содержанию отчетов по лабораторным работам

1. Титульный лист отчета должен содержать название, цель лабораторной работы, группу и фамилию студента, выполнившую её, и фамилию преподавателя, проверившего отчет.
2. Выполненное домашнее задание.
3. Содержание этого пункта отчета определяется требованиями, приведенными в пункте "Содержание отчета" выполняемой лабораторной работы.
4. Ответы на контрольные вопросы.

Лабораторная работа №1
Операции над основными типами данных языка С++

1. Цель работы: Изучение операций над основными типами данных в языке С++.

2. Домашнее задание

2.1 Привести в отчете диапазон значений целых и вещественных констант и соответствующих им типов данных.

2.2 Нарисовать в отчете схему подготовки исполняемой программы.

2.3 Привести в отчете таблицу приоритетов операций.

3. Лабораторное задание.

3.1 Набрать программу №1, приведенную ниже.

//Программа №1

```
#include "stdafx.h"
#include <iostream>
using namespace std;
void _tmain(int argc, _TCHAR* argv[])
{
	cout<<"\nsizeof(int) = " << sizeof(int);
	cout<<"\nsizeof(short) = " << sizeof(short);
	cout<<"\nsizeof(long) = " << sizeof(long);
	cout<<"\nsizeof(float) = " << sizeof(float);
	cout<<"\nsizeof(double) = " << sizeof(double);
	cout<<"\nsizeof(char) = " << sizeof(char);
	cout<<"\nsizeof('a') = " << sizeof('a');
	cout<<"\nsizeof(1) = " << sizeof(1);
	cout<<"\nsizeof(1L) = " << sizeof(1L);
	cout<<"\nsizeof(1U) = " << sizeof(1U);
	cout<<"\nsizeof(1.) = " << sizeof(1.);
	cout<<"\nsizeof(1.F) = " << sizeof(1.F);
	cout<<"\nsizeof(1.L) = " << sizeof(1.L);
	cout<< "\n (4<<2)  =  " <<(4<<2);
	cout<<"\n (5>>1)  =  " << (5>>1);
	cout<<"\n (6&5)  =  " << (6&5);
	cout<<"\n (6|5)  =  " << (6|5);
	cout<<"\n (3<5)  =  " << (3<5);
	cout<<"\n (3>5) =  " << (3>5);
	cout<<"\n (3==5) =  " << (3==5);
	cout<<"\n (3!=5) =  " << (3!=5);
	cout<<"\n Press key to continue";
	getchar();
```

```
        cout<<"\n (3!=5 || 3==5)  =  " << (3!=5 || 3==5);
        cout<<"\n (3+4>5 && 3+5 > 4  && 4+5>3)  =" << (3+4>5 && 3+5 > 4  && 4+5>3);

        int k;
        cout<<"\n (k=35/4)  = " << (k=35/4);                    // Тест
        cout<<"\n (k/=1+2+2)  =  " << (k/=1+2+2);
        cout<<"\n (k*=5-2)  =  " << (k*=5-2);
        cout<<"\n (k%=3+2)  =  " << (k%=3+2);
        cout<<"\n (k+=21/3)  =  " << (k+=21/3);
        cout<<"\n (k-=6-6/2)  =  " << (k-=6-6/2);
        cout<<"\n (k<<=2)  =  " <<  (k<<=2);
        cout<<"\n (k>>=6-5)  =  " << (k>>=6-5);
        cout<<"\n (k&=9+4)  =  " << (k&=9+4);
        cout<<"\n (k|=8-2)  =  " << (k|=8-2);
        getchar();
}
```

3.2 Отладить программу №1.
3.3 Проанализировать результаты работы программы.
3.4 Ответить на контрольные вопросы.
3.5 Написать отчет.

4. Содержание отчета.
4.1 Титульный лист. Домашнее задание
4.2 Текст программы №1 и результаты её выполнения.
4.3 Краткое пояснение против результатов выполнения каждой операции.

5. Контрольные вопросы.
5.1 Какие типы данных вы знаете, сколько байт занимает каждый тип данных, в каких диапазонах могут изменяются значения этих типов данных?
5.2 Поясните, что такое вещественная, целая, символьная и строковая константа. Какое максимальное целая константа может использоваться в программе на языке С++. Приведите примеры констант различного типа.
5.3 Какие управляющие символы могут быть использованы в строковой константе, поясните их назначение, приведите примеры их использования.
5.4 Расскажите об этапы подготовки исполняемой программы.
5.5 Идентификатор, правило записи идентификаторов в языке С++?
5.6 Операции, типы операций. Привести примеры операций определенных над множеством только целых типов данных.
5.7 Чем отличаются друг от друга операции & от &&, а также | от ||.
5.8 Раскройте смысл операции %, *=, +=, /=.
5.9 В этом выражении 8 операций (3+4>5 && 3+5 > 4 && 4+5>3) . Помете цифрами, в какой последовательности будут выполнятся эти 8 операций. Найдите вручную чему равно значение выражения (3+4>(5 && 3)+5 > 4 && 4+5>3). Объясните почему.
5.10 Что такое приоритеты и ассоциативность операций.
5.11 В строке программы, напротив которой написан комментарий //Тест, вместо константы 35 подставьте константу заданную вам преподавателем, и рассчитайте в ручную значение всех выражений определенных ниже.

Лабораторная работа №2
Явное и неявное преобразование типов в языке С++
Определение, описание и вызов функций в языке С++.

1. Цель работы: Изучение операции преобразования типа в языке С++, а также определения, описания и вызова функции пользователем.

2. Домашнее задание
2.1 Сформулировать и записать правила преобразования типов при выполнении арифметических операций и операции присваивания. Привести примеры.

2.2 Сформулировать и записать правило передачи данных в функцию по значению через аппарат фактических и формальных параметров. Привести пример.

3. Лабораторное задание.

3.1 Набрать программы №1 и №2.

// Программа №1 Преобразование типов.

```
#include "stdafx.h"
#include <iostream>
#include <math.h>
using namespace std;

void main (){
      long k = 123456789;
      float g = (float)k;
      cout<<"\n\n k = "<<k;
      cout<<"\n  g= "<<g;
      k = (long)g;
      cout<<"\n k = "<<k;                //Объяснить результат
      g = (float) 2.222222e+2;
      int m = (int)g;
      cout<<"\n\n g = " <<g;
      cout<<"\n m = "<<m;
      g = (float)m;
      cout<<"\n g = "<<g;
      int a=1, b = 2;
      double c = 1;
      c = a/b*c+ a*c/b;
      cout<<"\n c = "<<c;                        //Объяснить результат
      cout<<"\n 1/2*exp(1) = "<<1/2*exp(1.);    //Объяснить результат
      getchar();
}
```

```
// Программа №2 Функции, определенные пользователем
#include "stdafx.h"
#include <iostream>
using namespace std;

extern double k;                                    //Описание переменной k
void main (){
      double a = 2, b = 3;//Описание, определение и инициализация переменных a и b.
      k = 1.3*k/((a+b)*(a-b));
      cout<< "\n k = "<<k;
      getchar();
}
double k=1;  //Определение переменной k
```

3.2 Отладить программы №1 и №2.

3.3 Модифицировать программу №2 определив функции, реализующие арифметические операции сложения (+), вычитания (-), умножения (*), деления (/). Имена функциям придумать самостоятельно. Вставить в строку программы k = 1.3*k/((a+b)*(a-b)); вместо операции и её операндов соответствующий вызов функции.

```
//Пример определения и вызова функции, реализующей операцию сложения +:
#include "stdafx.h"
#include <iostream>
using namespace std;

double sum(double a, double b);              //Описание функции sum( )
extern double k;
void main (){
```

```
	k = 1;
	double a = 2, b = 3;
	k = 1.3*k/(sum(a,b)*(a-b)); //Строка программы содержит вызов функции sum( )
	cout<< "\n k = "<<k;
	getchar();
}
double sum(double a, double b){                //Определение функции  sum( )
	return a+b;
}
double k;
```

3.4 Проанализировать результаты работы программ.

3.5 Написать программу №3, вычисляющую значение функции $S(t) = 1.2e^{-0.1t}\sin(2\pi * 1000t)$ при t = 0, 0.1 и 0.4.

3.6 Ответить на контрольные вопросы.

3.7 Написать отчет.

4. Содержание отчета

4.1 Титульный лист. Домашнее задание.

4.2 Текст и результаты работы программы №1.

4.3 Текст модифицированной программы №2, с функциями, реализующими операции - , *, /.

4.4 Текст программы №3.

5. Контрольные вопросы

5.1 Что такое описание, определение и вызов функции (пояснить на примере функций определенных вами в лабораторном задании)? Для чего необходимо описание функции, можно ли его опустить?

5.2 Что такое формальные и фактические параметры? Поясните способ передачи данных в функцию через параметры и через глобальную переменную?

5.3 Поясните, с помощью какого оператора функция возвращает значение в точку вызова. Какой тип этого значения?

5.4 Что такое описание и определение переменной? Приведите примеры описания и определения переменных? В чем их принципиальная разница?

5.5 Что такое инициализация переменной? Чем инициализация переменной отличается от присваивания переменной значения.

5.6 Свойства переменных различного типа: объем занимаемый в памяти, диапазон значений, количество значащих цифр, множество операций, определенных над ними, область видимости и область существования.

5.7 Какие переменные называются локальными, глобальными, статическими? Приведите примеры. Может ли локальная переменная быть статической?

5.8 Приведите примеры явного и неявного преобразования типов при выполнении арифметических операций и операции присваивания. Объясните результаты работы программы №1.

5.9 Как использовать библиотечные функции? Что такое заголовочный файл, и что он содержит. Где содержится определение и описание библиотечных функций, использованных в лабораторной работе?

5.10 Напишите функцию, вычисляющую модуль, и функцию, вычисляющую, аргумент комплексного числа.

Лабораторная работа №3
Операторы языка С++. Библиотечные функций языка С++

1. Цель работы: Получение навыков работы с операторами языка С++. Изучение одного из алгоритмов сортировки данных – пузырьковая сортировка.

2. Домашнее задание

2.1 Изучить операторы языка С++ и их изображение на блок схеме [2].

2.2 Нарисовать блок схему алгоритма программ №1

2.3 Нарисуйте блок схему алгоритма, реализующего пузырьковую сортировку.

3. Лабораторное задание

3.1 Набрать и отладить программу №1, которая вычисляет площадь фигуры, изображённой на рис. 1

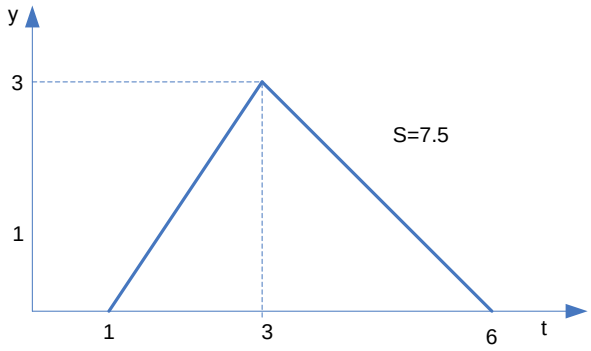

Рис. 1 Исследуемая фигура

```
   //Программа №1
#include "stdafx.h"
#include <iostream>
using namespace std;

void main (){
      int N;
      cout<<"\n Input N = ";
      cin>>N;
      double a = 1, b = 6 ;
      double h = (b-a)/N, t = a, S1 = 0,S2 = 0, y=0;
      //Находим площадь фигуры методом прямоугольников
      for(int i = 0; i < N; i++, t+=h){
            if(t>=a && t<=3) {y = 1.5*t-1.5; S1 += h*y; }
            if(t>=3 && t<=b) y = -1*t+6, S1 += h*y;
      }
      double y_last = 0;
      t = a, y=0;
      // Находим площадь фигуры методом трапеций
      for(int i = 0; i < N+1; i++, t+=h){
            if(t>=a && t<=3) y = 1.5*t-1.5, S2 += h*(y+y_last)/2;
            if(t>3 && t<=b) y = -1*t+6, S2 += h*(y+y_last)/2;
            y_last = y;
      }
      cout<<"\n  S1= "<<S1;
      cout<<"\n  S2= "<<S2;
      getchar();
      getchar();
}
```

3.2 Рассчитать площадь нарисованной преподавателем для Вас фигуры методом прямоугольников и трапеций.
3.3 Набрать и отладить программу №2

//Программа №2 (пузырьковая сортировка)

```
#include "stdafx.h"
#include <iostream>
#include <stdio.h>
#include <stdlib.h>
#include <iostream>
using namespace std;

const int  size = 100;
float fData[size];
```

```
void BubbleSort(float fArrayToSort[],int iNumberOfElements);
void main()
{
      srand(1); // Инициализация генератора случайных чисел
      int index;
      // Заполняем исходный массив набором случайных чисел
      for (index = 0; index < size; ++index)
            fData[index] = rand() % 100;
      BubbleSort(fData, size);
      for (index = 0; index < size; index++)
            cout<<"\n fData["<<index<< "] = "<<fData[index];
      printf("\n");
      getchar();
}
//
//---------------------------------------------------
void BubbleSort(float fArrayToSort[],float iNumberOfElements)//синтаксическая ошибка
{
   int i,j;
   float fTemp;
   for ( i = 0; 1 < iNumberOfElements - 1; i++) // Логическая ошибка
         for (j = iNumberOfElements - 1; j > i; j--){
               if (fArrayToSort[j] < fArrayToSort[j-1]){
                     // Меняем местами fArrayToSort[j] и fArrayToSort[j-1];
                        fTemp = fArrayToSort[j];
                        fArrayToSort[j] = fArrayToSort[j-1];
                        fArrayToSort[j-1] = fTemp;
                  }
            }
}
```

3.4 Используя Help, выписать описание всех стандартных функций, используемых в программах №1 и №2.
3.5 Найти и исправить в программе две ошибки. Модифицируйте программу №2 так, чтобы она находила и выводила на экран минимальное, максимальное и среднеарифметическое значение массива сортируемых чисел.
3.6 Написать отчет.

4. Содержание отчета

4.1 Титульный лист. Домашнее задание.
4.2 Текст модифицированной программы №1 и №2.
4.3 Описания всех стандартных функций, которые были использованы в программе №1, а также краткое пояснение их назначения.

5. Контрольные вопросы

5.1 Напишите программу, вычисляющую такое значение n, при котором значение предела $\lim_{n\to\infty}(1+1/n)^n$ вычислялось бы с точностью до 3 знака после запятой.
5.2 Напишите программу сортирующую в алфавитном порядке десять произвольных букв, введенных с клавиатуры.
5.3 Напишите программу, сортирующую по возрастанию пять действительных чисел, введенных с клавиатуры.
5.4 Напишите программу, вычисляющую факториал числа.
5.5 Напишите программу вычисляющую корни квадратного уравнения $a_0x^2+a_1x+a_2=0$ (коэффициенты a_0, a_1, a_2 задаются пользователем с клавиатуры).
5.6 Напишите программу вычисляющую значение функции $y(x) = a_0x^2+a_1x+a_2$ (коэффициенты a_0, a_1, a_2 задаются пользователем с клавиатуры).
5.7 Напишите программу вычисляющую значение функции $f(t) = 4\pi\sin(2\pi t)+ 4\pi/3\sin(6\pi t) +4\pi/5\sin(10\pi t)$.

5.8 Используя описание библиотечных функций, объясните какие типы параметров они принимают и какое значение возвращают.
5.9 Объясните, что такое пустой и составной операторы. Приведите примеры.
5.10 Поясните как работают операторы цикла. Приведите примеры.
5.11 Поясните как работают условные операторы. Приведите примеры.
5.12 Какие операторы передачи управления вы знаете. Приведите примеры.
5.13 Кратко пояснить назначение библиотечных функций, используемых в ваших программах.

Лабораторная работа №4
Конвертор текста из кодировки MS Windows в кодировку MS-DOS

1. Цель работы: Приобретение навыка работы со статическими массивами и указателями, а также освоение стандартных функций создания, открытия, закрытия, записи и чтения из файла.

2. Домашнее задание

1. Запишите последовательность кодов, которым представлена строка "Здравствуй мир!" в MS-DOS и MS Windows кодировках.
2. Для перекодировки символа 'A' русского алфавита из кодировки MS-DOS в кодировку MS Windows необходимо выполнить следующие операторы (unsigned char p = 'A'; p= p+64;), что нужно сделать, чтобы перекодировать символ 'Ё'?

3. Лабораторное задание

3.1 Набрать программу №1 и программу №2

//Программа №1

```
//В свойствах проекта в опциях по C/C++->Препроцессор->Определения препроцессора; добавить
_CRT_SECURE_NO_WARNINGS
#include "stdafx.h"
#include <iostream>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <fcntl.h>
#include <sys\stat.h>
#include <io.h>
using namespace std;
int main(void){
	setlocale(LC_ALL, "rus");
	int handle;
	char msg[] = "Hello world";
	if ((handle = _open("C:\\RXX\\TEST.txt", O_CREAT | O_TEXT| O_RDWR, S_IWRITE | S_IREAD)) ==
-1)   {
		perror("Error:");
		getchar();
		return 1;
	}
	_write(handle, msg, strlen(msg)); //msg == &msg[0]
	_close(handle);
	cout<<"\n Hello world - write to file";
	getchar();
	return 0;
}
```

//Программа №2

```
#include "stdafx.h"
#include <stdio.h>
#include <io.h>
#include <malloc.h>
```

```
#include <fcntl.h>
#include <process.h>
#include <sys\stat.h>
#include <iostream>
using namespace std;
int main(void){

   int handle, bytes;
   if ((handle =_open("C:\\RXX\\TEST.txt", O_RDWR | O_BINARY, S_IWRITE | S_IREAD)) == -1){
      printf("Error Opening File\n");
         getchar();
         exit(1);
   }
   long longFile = _lseek(handle,0,SEEK_END);
   _lseek(handle,0,SEEK_SET);
   unsigned char* buf = (unsigned char*)malloc(longFile+1);
   if ((bytes = _read(handle, buf, longFile)) == -1) {
      printf("Read Failed.\n");
         free(buf);
         getchar();
      exit(1);
   }
   else {
      printf("Read: %d bytes read.\n", bytes);
//    buf[longFile]='\0';
      cout<<buf;
   }
   free(buf);
   getchar();
   return 0;
}
```

3.2 Отладить программы №1и №2. (Программа №1 является примером создания нового файла Test.txt и записи в него текстовой строки "Hello world". Программа № 2 является примером чтения данных из файла Test.txt в буфер.)

3.3 Модифицировать программу №2 так, чтобы она читала данные из файла и записывала их в этот же файл в обратном порядке, т.е. если в файле содержится строка "Hello world", то после выполнения Вашей программы в этом файле должна содержатся строка "dlrow olleH"

3.4 Написать программу, которая бы перекодировала текстовый файл из кодировки MS-DOS в кодировку MS Windows. (Таблица кодов приведена в [1] на стр. 488-493).

3.5 Ответить устно на контрольные вопросы.

3.6 Написать отчет.

4. Содержание отчета

4.1 Титульный лист. Домашнее задание.

4.2 В отчете привести тексты программ и описание всех использованных библиотечных функций, а также кратко пояснить их назначение.

5. Контрольные вопросы

5.1 Как определить указатель? Какое множество операций определено над указателями? Приведите примеры операций с указателями.

5.2 Чем является переменна s, определенная как int b; int& s = b;? Какие операции определены над s.

5.3 Запишите описание функции lseek() и кратко поясните её назначение.

5.4 Для массива определенного как double Ar[3][4][5], чем является переменная Ar, Ar[2], Ar[1][3], ***Ar, *(*(*(Ar+1)+2)+1).

5.5 Поясните значение операций разыменования * и получения адреса &. Приведите примеры использования этих операций.

5.6 Напишите функцию, конвертирующую содержимое строки, состоящей из прописных букв в строчные и наоборот.

5.7 Напишите функцию, которая подсчитывает количество гласных и согласных в строке.

5.8 Напишите функцию, которая подсчитывает количество букв и цифр в строке.
5.9 Напишите функцию, которая подсчитывает число прописных и строчных букв в строке.
5.10 Напишите функцию, которая сравнивает две строки, и если они совпадают, то возвращает единицу (истину) в противном случае ноль (ложь).
5.11Напишите функцию, которая сортирует строку по алфавиту.
5.12Напишите функцию, которая шифрует текстовый файл путем замены значения символа с помощью выражения sym = F(sym), где F( ) - функция (например, значение символа C заменяется на C=C^0xFF).

Лабораторная работа №5
Способы передачи данных в функции языка C++

1. Цель работы: Изучение способов передачи параметров в функцию. Рекурсивные функции.

2. Домашнее задание
2.1 Приведите описание функций принимающие параметры по значению, по ссылке и по указателю. Поясните, какая разница между этими способами передачи параметров в функцию. Напишите, что будет выведено на экран следующей программой.

```
#include <iostream.h>
int f1(int b) {b=b+1; return b;}//
int& f2(int& b) {b=b+1; return b;}
int* f3(int* b) { *b=*b+1;return b;}
void main(){
int a =1; cout<<"\n a = "<<a;
cout<<"\n f1(a) = "<<f1(a); cout<<"\t a = "<<a; // f1(a) =  2  a = 1;
cout<<"\n f2(a) = "<<f2(a); cout<<"\t a = "<<a;// f2(a) =  2  a = 2;
cout<<"\n *f3(&a) = "<<*f3(&a); cout<<"\t a = "<<a; // *f3(&a) =  3  a = 3;
}
```

2.2 Переделайте следующую функцию, вычисляющую факториал с помощью цикла на рекурсивный вариант:

```
double factorial(unsigned value)
{
  unsigned i = 1;
  double result = 1;
  for (; i <= value; i++)   result *= i;
  return result;
}
```

2.3. Согласно номеру вашего варианта рассчитайте значение интеграла от:

1. $F(x) = \sin(x)+4*\cos(2*x)$ от 0 до $\pi/4$
2. $F(x) = 7*x^2+5*x+3$ от 3 до 6
3. $F(x) = -7*x^3+3*\cos(3*x) –x$ от 0 до $\pi/2$
4. $F(x) = 5*\cos(3*x)-12*\sin(3*x)$ от $\pi/4$ до π
5. $F(x) = 0.01*(1-x^2)$ от 0.01 до 0.5
6. $F(x) = 1/(1-x)$ от 2 до 5
7. $F(x) = \sin(x)+(x^2+4)$ от $-\pi/4$ до $\pi/4$
8. $F(x) = 7*x^3-5*x+7$ от 1 до 2
9. $F(x) = \sin(x)+\cos(4x)$ от 0 до π
10. $F(x) = 0.1*x^2(1-x^3)$ от -3 до 3

3. Лабораторное задание
3.1 Наберите программу, использующую рекурсивный алгоритм Хаара, для сортировки массива случайных чисел.

```
#include "stdafx.h"
# include <stdio.h>
# include <stdlib.h>
#include <time.h>
# define DIMENSION 5000
void QuickSort(int* array, int First, int Last)
```

```
{
   int Temp, LowerBoundary, UpperBoundary, Separator;
   LowerBoundary = First;
   UpperBoundary = Last;
   Separator = array[(First + Last) / 2];
   do
   {
      while (array[LowerBoundary] < Separator) LowerBoundary++;
      while (array[UpperBoundary] > Separator) UpperBoundary--;
      if (LowerBoundary <= UpperBoundary)
      {
         Temp = array[LowerBoundary];
         array[LowerBoundary++] = array[UpperBoundary];
         array[UpperBoundary--] = Temp;
      }
   } while (LowerBoundary <= UpperBoundary);
   if (First < UpperBoundary) QuickSort(array, First, UpperBoundary);
   if (LowerBoundary < Last) QuickSort(array, LowerBoundary, Last);
}
void main()
{
      time_t ftime, stime;
   int i = 0;
      int* ar = (int*) malloc(sizeof(int)*DIMENSION);
   for (; i < DIMENSION; ar[i++] = rand()%1000) ;
      time(&ftime); // время начала сортировки
   QuickSort(ar, 0, DIMENSION -1);
      time(&stime); //время окончания сортировки
   printf("\n\n");
//   for (i = 0; i < DIMENSION; printf("\n%d", ar[i++])) ;
   printf("\n stime - ftime = %d", stime - ftime);
   getchar();
   free(ar);
}
```

3.2 Оцените время сортировки массивов из 5000, 50000, 500000 элементов алгоритмом Хаара и методом пузырьковой сортировки. Данные измерений занесите в таблицу. Сделайте выводы.

3.3. Напишите программу, вычисляющую значение интеграла от функции, заданной в пункте 2.3, методом прямоугольников и методом трапеций. Сравните результат численного интегрирования, с результатами, полученными вами вручную.

3.4 Нарисуйте блок схему, алгоритма сортировки Хаара.

3.5 Наберите программу, рисующую на экране геометрические фигуры.

```
// В свойствах проекта в опциях по C/C++ ->Библиотека времени выполнения; установить /MTd
//В свойствах проекта Компоновщик->Ввод->Дополнительные зависимости;  поставить на первое место
uafxcwd.lib

#include "stdafx.h"
#include "afxwin.h"
#include "iostream"
using namespace std ;
void main(){
       HWND hwnd;
       hwnd=FindWindow(_T("ConsoleWindowClass"),_T("C:\\Windows\\system32\\cmd.exe"));
       if (hwnd!=NULL){
              HDC hdc;
              hdc=GetWindowDC(hwnd);
              POINT pt[3];
```

```
		CPoint pt1(100,100), pt2(200,200);
		pt[0].x = 0;pt[0].y=0; pt[1].x = 100;pt[1].y=100; pt[2].x = 50;pt[2].y=100;
		if (hdc!=0){
			CPen pen(PS_SOLID,4,RGB(255,0,0));
			SelectObject(hdc,pen);
			Ellipse(hdc,90,100,120,150);// рисуем эллипс
			Arc(hdc,pt1.x,pt1.y,pt2.x,pt2.y,100,200,100,100);//рисуем круг
			Polyline(hdc,pt,3 ); //рисуем ломаную
			getchar();
			for(int i=0; i < 100; i++){ // перемещение по экрану
				CPen pen1(PS_SOLID,4,RGB(0,255,0));
				SelectObject(hdc,pen1);
				Arc(hdc,pt1.x,pt1.y,pt2.x,pt2.y,100,200,100,200);//рисуем круг
				Sleep(24);//задержка на 24мс
				CPen pen2(PS_SOLID,4,RGB(0,0,0));
				SelectObject(hdc,pen2);
				Arc(hdc,pt1.x,pt1.y,pt2.x,pt2.y,100,200,100,200);//стираем круг
				pt1.x+=1; pt1.y+=1;pt2.x+=1;pt2.y+=1; //Меняем координаты круга
			}
			ReleaseDC(hwnd, hdc);
		}
		else cout << "Error DC Window" << endl;
	}
		else cout << "Error Find Window" << endl;
}
```

3.6. Наберите программу, рисующую на экране геометрические фигуры.
3.7. Получите рисунок у преподавателя и напишите программу рисующую его на экране.
3.8. Ответьте устно на контрольные вопросы.
3.9. Напишите отчет.

4. Содержание отчета
4.1 Титульный лист. Домашнее задание..
4.2 Результаты выполнения пунктов 3.2, 3.3, 3.4 и 3.7.

5. Контрольные вопросы
5.1 Чем отличается передача параметров в функцию по значению и по ссылке?
5.2 Каким образом программа может получать данные из командной строки?
5.3 Как можно из функции main получить доступ к переменным окружения?
5.4 Может ли функция возвращать массив? Принимать массив по значению?
5.5 Может ли аргументом функции быть другая функция? Указатель на другую функцию?
5.6 Приведите примеры использования указателя на функцию.
5.7 Как объявляется функция с переменным числом параметров? Приведите примеры.
5.8 Каким образом можно определить, сколько параметров ввел пользователь в командной строке?
5.9 Разработайте рекурсивный алгоритм расчета определителя квадратной матрицы.
5.10 Объясните алгоритм быстрой сортировки, используемый программой из пункта 2.2. лабораторного задания.
5.11 Почему при работе с рекурсивными функциями часто переполняется стек?
5.12. Покажите действие алгоритма быстрой сортировки на примере следующего массива:
{6,2,1,3,4,5,8,7,0}.
5.13. Всегда можно ли итерационный алгоритм заменить рекурсивным? Рекурсивный итерационным?

Лабораторная работа №6
Многомерные массивы, массивы динамической памяти.

1. Цель работы: Приобретение навыков работы с динамическими массивами памяти.

2. Домашнее задание
2.1 Изучите материал по темам: массивы и указатели, многомерные массивы, массивы указателей, динамические

массивы.

2.2 Приведите примеры описания, определения и инициализации статического одномерного и многомерного массивов, массива указателей, динамического массива.

3. Лабораторное задание.

3.1 Набрать программу № 1

```
// Программа №1
#include "stdafx.h"
#include <iostream>
using namespace std;
void main(){
	int n;
	cout<<"\n Input n =";
	cin>>n;
	double **matr;
	matr = new double* [n];
	if(matr == NULL) {
		cout<<"\n Не создан динамический массив";
		return;
	}
	for(int i = 0; i < n ; i++){
		matr[i] = new double [n];
		if(matr[i]== NULL) {
			cout<<"\n Не создан динамический массив";
			return;
		}
		for(int j = 0; j < n;j++)
			if(i != j) matr [i][j] = 0;
			else matr[i][j] = 1;
	}
	for(int i = 0; i < n; i++){
		cout<<"\n string "<<":"<<i;
		for(int j = 0;j< n; j++)
			cout<<"\t"<<matr[i][j];
	}
	for(int i=0; i < n; i++) delete matr[i];
	delete[]matr;
	getchar();getchar();
}
```

3.2 Отладить программу №1 (Эта программа является примером создания двухмерного динамического массива.)

3.3 Написать функцию, реализующие перемножение, сложение, вычитание двух матриц. Эти функции должны принимать указатели (или ссылки) на матрицы и возвращать указатель на результирующую матрицу.

3.4 Написать функцию транспонирования матрицы.

3.5 Написать алгоритм и функцию решения системы линейных алгебраических уравнений методом Гаусса.(*)

3.6 Ответить на контрольные вопросы.

3.6 Написать отчет.

4. Содержание отчета

4.1 Титульный лист. Домашнее задание.

4.2 Тексты программ, написанных при выполнении 3.3 и 3.4 пунктов лабораторного задания.

5. Контрольные вопросы

5.1 Операторы и функции динамического выделения памяти new, delete, malloc(), free(). Привести примеры использования этих операторов и функций.

5.2 В чем разница между динамическим массивом и статическим массивом памяти.

5.3 Напишите функцию, принимающую в качестве параметров указатели (или ссылки) на две строки и возвращающую указатель на новую строку, являющуюся результатом слияния этих строк.
5.4 Приведите примеры передачи в функцию как параметра динамического и статического массива памяти.
5.5 Напишите функцию, принимающую указатель (или ссылку) на матрицу в качестве параметра и возвращающую указатель на копию этой матрицы, повернутую по часовой стрелки.
5.6 Напишите функцию, принимающую указатель (или ссылку) на матрицу в качестве параметра и возвращающую указатель на копию этой матрицы, повернутую против часовой стрелки.
5.7 Напишите функцию, принимающую указатель (или ссылку) на матрицу в качестве параметра и возвращающую указатель на копию этой матрицы, транспонированную относительно не главной диагонали.
5.8 Напишите функцию, принимающую указатель (или ссылку) на матрицу в качестве параметра и возвращающую указатель на копию этой матрицы, повернутую зеркально вниз.
5.9 Напишите функцию, принимающую указатель (или ссылку) на матрицу в качестве параметра и возвращающую указатель на копию этой матрицы, повернутую зеркально вправо.
5.10 Перечислите способы передачи параметров в функцию. Приведите примеры.

Лабораторная работа №7
Линейные списки и структурированные данные.

1. Цель работы: Получение навыка работы со структурами и линейными списками.

2. Домашнее задание

1. Разработать алгоритм и пояснить на рисунках последовательность действий, которую необходимо выполнить для того, чтобы вставить объект в начало, в конец, в середину или удалить из двухсвязного списка.

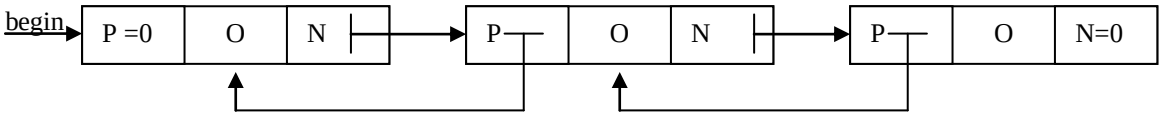

Рис 1 Двухсвязный список

P(prior) - указатель на предыдущий элемент списка; N(next) - указатель на следующий элемент списка; begin - указатель на начало списка; O - объект или указатель на объект, вставленный в список.

2. Изучить текст программы №1.

Лабораторное задание

1. Набрать программу №1

//Программа №1

```
#include "stdafx.h"
#include<string.h>
#include<iostream>
using namespace std;
struct card {   //Определение структурного типа для книги
                        char *author; // Ф.И.О. автора
                        char *title;  // Заголовок книги
                        char *city;   // Место издания
                        char *firm;   // Издательство
                        int year;      // Год издания
                        int pages;     // Количество страниц
                        };
//Функция печати сведений о книге:
void printbook(card& car)
 { static int count = 0;
      cout<<"\n"<< ++count <<". "<<car.author;
      cout<<"  "<<car.title<<".- "<<car.city;
      cout<<": "<<car.firm<<", ";
      cout<<"\n"<<car.year<<".- "<<car.pages<<" с.";
```

```
}
struct record {   //Структурный тип для элемента списка (1)
                                card book;
                                record *prior;
                                record *next;
                              };
//Исходные данные о книгах:
card books[] = { //Инициализация массива структур:     (2)
  { "Wiener R.S.", "Turbo C",
                                                              "M",    "ST",1991,
384},
  { "Stroustrup B.","Langvige C",
                                                            "kiev","DiaSoft",1993, 560},
  { "Turbo C++.", "For programm",

"M","INTKV",1991,394},
  { "Limppman S.B.","C++ for new",

"M","GELION",1993,496}

};
void main()
{ record *begin = NULL, //Указатель начала списка          (3)
                  *last = NULL,   //Указатель на очередную запись
                  *list;                  //Указатель на элементы списка
// n-количество записей в списке:
int n = sizeof(books)/sizeof(books[0]);
// Цикл обработки исходных записей о книгах:
for (int i=0;i<n; i++)
      {//Создать новую запись(элемент списка):              (4)
       last = new(record);
       //Занести сведения о книге в новую запись:
       (*last) .book.author = books[i].author;
       (*last) .book.title = books[i].title;
       last->book.city = books[i].city;
       last->book.firm = books[i].firm;
       last->book.year = books[i].year;
       last->book.pages = books[i].pages;
       //Включить запись в список(установить связи):
       if (begin == NULL) //Списка ещё нет                  (5)
             {last->prior = NULL;
              begin = last;
              last->next = NULL;
             }
       else
             { //Список уже существует
              list = begin;
              //Цикл просмотра цикла - поиск места для
              //новой записи:
              while (list)                                  //(6)
                   {if (strcmp(last->book.author,
                                       list->book.author) < 0 )
                              {//Вставить новую запись перед list:
                                   if (begin == list)
                                          {//Начало списка: (7)
                                          last->prior = NULL;
                                          begin = last;
```

```
                                        }
                                      else
                                        {//Вставить между записями: (8)
                                            list->prior->next = last;
                                            last->prior = list->prior;
                                        }
                                      list->prior = last;
                                      last->next  = list;
                                      //Выйти из цикла просмотра списка:
                                      break;
                                   }
                                if (list->next == NULL)
                                     {//Включить запись в конец цикла: (9)
                                     last->next = NULL;
                                     last->prior = list;
                                     list->next = last;
                                     //Выйти из цикла просмотра списка:
                                     break;
                                   }
                              //Перейти к следующему элементу списка:
                              list = list->next;
                             }//Конец цикла просмотра списка
                              //(Поиск места для новой записи)
              }               //Включение записи выполнено
  }                   //Конец цикла обработки исходных данных

      //Печать в алфавитном порядке библиографического списка:
      list = begin;                                                  // (10)
      cout<<"\n";
      while (list) {
            printbook(list->book);
            list = list->next;
      }
      getchar();
}
```

2. Отладить программу №1 (Программа является примером использования двухсвязного линейного списка.)
3. Написать функции вставки в начало, в середину и в конец списка объекта типа card, а также функции удаления и доступа (по индексу) к объекту типа card в списке.
4. Модифицировать программу №1 так, чтобы можно было с клавиатуры вводить и удалять записи в списке.
5. Модифицировать функции списка так, чтобы ими можно было пользоваться для организации списка произвольного типа объектов *.
6. Написать отчет.

Содержание отчета.

1. Титульный лист. Домашнее задание
2. Тексты функций написанных при выполнении пунктов 3,4 и 5 лабораторного задания.

Контрольные вопросы

1. Что такое стек, очередь, дек?
2. Чем отличается указатель типа void* от указателей другого типа type*?
3. Расскажите, что такое структура и объединение? Приведите примеры определения, инициализации и описания структуры и объединения.
4. Как получить доступ к полю структуры и объединения? Какое множество операций определено над структурой/объединением и полями структуры/объединения?

5. Приведите примеры использования оператора typedef со стандартными типами данных, со структурами и объединениями.
6. Битовые поля структур и объединений, назначение, способ определения, множество операций, определенных над битовыми полями.
7. Написать функции реализующие стек целых чисел.
8. Написать функции реализующие дек вещественных чисел.
9. Написать функции реализующие очередь символьных переменных.
10. В чем состоят преимущества и недостатки связанных списков перед массивами? Приведите примеры.
11. Запишите синтаксис доступа к полям данных структурированных типов данных через имя объекта этого типа и указатель на объект этого типа.
12. Напишите функцию сортировки объектов типа card, вставленных в список, по названию книги или году издания. (Для этого необходимо создать динамический массив указателей M на объекты типа card в списке и с помощью функции qsort() отсортировать массив M по соответствующему критерию, при этом последовательность объектов в списке остается неизменной).

Лабораторная работа №8
Типы данных, определённые программистом

Цель работы: Получение навыков в создании новых типов данных и определении множества операций над ними.

Домашнее задание.
Изучить материал по темам: конструкторы, деструкторы и доступность компонентов класса; компонентные данные и компонентные функции; друзья классов; расширение действия стандартных операций; шаблоны классов.

Лабораторное задание
1. Набрать и отладить программу №1

//Программа №1
```
#include "stdafx.h"
#include <string.h>
#include <iostream>
#include <MATH.H>
#include <stdio.h>
#include <stdlib.h>
using namespace std;

//                                                          class Matr
//-------------------------------------------------------------------------
template <class T>
class Matr
  {
  long line;
  long col;
  T ** matr;

  public:
  virtual    ~Matr();                  //Деструктор
     Matr(long l, long c);          // Конструктор
     Matr():line(0), col(0),matr(NULL){} //Конструктор по умолчанию
     Matr(Matr<T>& A);                  //Конструктор копии

  T *             operator[](long i){return matr[i];}
      template <class T>
  friend Matr<T>  operator*( Matr<T>& A, Matr<T>& B);
  const  Matr<T>& operator=(const Matr<T>& A);
      template <class T>
  friend Matr<T>  operator*(double K, const Matr<T>& A);
```

```
  Matr<T>         operator+(const Matr<T>& A);
  Matr<T>         operator-(const Matr<T>& A);
  void  display();
  };

//                                                    Matr<T>::Matr()
//---------------------------------------------------------------------
template <class T>
Matr<T>::Matr(long l1, long col1)
   {
   line = l1;
   col  = col1;
   matr = new T*[line];
   if(matr == NULL) cout<< "Out of memory";
   for (int i = 0; i < line; i++){
      matr[i] = new T[col];
      if(matr[i] == NULL) cout<<"Out of memory";
   }
   }
//                                    Matr<T>::Matr(Matr<T>& A)
//---------------------------------------------------------------------
template <class T>
Matr<T>::Matr(Matr<T>& A):line(A.line), col(A.col)
   {
   matr = new T*[line];
   if(matr == NULL) cout<< "Out of memory";
   for (int i = 0; i < line; i++){
      matr[i] = new T[col];
      if(matr[i] == NULL) cout<<"Out of memory";
   }
   for(long j = 0; j<line; j++){
      for( int    i = 0; i<col; i++){
       matr[j][i] = A.matr[j][i];
       }
      }
   }
//                                                   Matr<T>::~Matr()
//---------------------------------------------------------------------
template <class T>
Matr<T>::~Matr()
   {
   for (int i = 0; i < line; i++) delete matr[i];
   delete matr;
   }
//                                    void  display(const Matr<T>& A)
//---------------------------------------------------------------------
template <class T>
void Matr<T>::display(){
  cout<<"\n";
  for(int i = 0; i<line;i++){
      cout<<"\n";
      for(int j = 0; j < col; j++)
       cout<<matr[i][j]<<"\t";
  }
}
//                                              Matr<T>::operator*()
//---------------------------------------------------------------------
```

```
template <class T>
Matr<T> operator*( Matr<T>& A,
                   Matr<T>& B)
   {
   if(!(A.col == B.line)) cout<<"\n A*B A.col != B.line";
   Matr<T> arMatr(A.line, B.col);
   long l1   = A.line;
   long col1 = A.col;
   long col2 = B.col;
   for(long i = 0; i < l1; i++){
      for(long j = 0; j < col2; j++){
           arMatr[i][j] = 0;
            for(long k = 0; k < col1; k++) {
             arMatr[i][j]+=A[i][k]*B[k][j];
           }
       }
      }
   return arMatr;
   }
//                                                    Matr::operator=()
//-------------------------------------------------------------------
template <class T>
const Matr<T>& Matr<T>::operator=(const Matr<T>& A)
   {
   if(this == &A) return *this;
   line = A.line;
   col  = A.col;
   for(long i = 0; i<A.line; i++){
       for(long j = 0; j<A.col; j++){
         matr[i][j] = A.matr[i][j];
         }
       }
   return *this;
   }
//                                              Matr<T>::operator*()
//-------------------------------------------------------------------
template <class T>
Matr<T> operator*(double K, const Matr<T>& A)
   {
   Matr<T> M(A.line, A.col);
   for(long i = 0; i<A.line; i++){
       for(long j = 0; j<A.col; j++){
             M.matr[i][j] = K * A.matr[i][j];
        }
   }
   return M;
   }
//                                              Matr<T>::operator+()
//-------------------------------------------------------------------
template <class T>
Matr<T> Matr<T>::operator+(const Matr<T>& A)
    {
    if(line != A.line || col != A.col) {
        cout<<"\n A != B";
        Matr<T> M(0,0);
        return M;
      }
```

```
   Matr<T> M(A.line, A.col);
   for(long i = 0; i<A.line; i++){
       for(long j = 0; j<A.col; j++){
             M.matr[i][j] = matr[i][j] + A.matr[i][j];
             }
       }
   return M;
   }
//                                                Matr<T>::operator-()
//-------------------------------------------------------------------------
template <class T>
Matr<T> Matr<T>::operator-(const Matr<T>& A)
    {
    if(line != A.line) {
      cout<<"\n - no A.line = B.line";
        Matr<T> M(0,0);
      return M;
      }
    if(col != A.col)  {
      cout<<"\n - no A.line = B.line";
        Matr<T> M(0,0);
      return  M;
      }
   Matr<T> M(A.line, A.col);
   for(long i = 0; i<A.line; i++){
       for(long j = 0; j<A.col; j++){
            M.matr[i][j] = matr[i][j] - A.matr[i][j];
            }
       }
   return M;
   }
//                                                              TMatr()
//-------------------------------------------------------------------------
template <class T>
Matr<T> TMatr(Matr<T>& M){
    Matr<T> TM(M.col, M.line);
    for(int i = 0; i < M.line; i++)
            for(int j = 0; j < M.col; j++)
            TM[j][i] = M[i][j];
    return TM;
}

void main(){
Matr<double> A(2,2), B(2,2);
A[0][0]=A[0][1]=A[1][0]=A[1][1] = 1;
B[0][0]=B[0][1]=B[1][0]=B[1][1] = 2;
A.display();
B.display();
A=(2.5*A-A+B)*B;
A.display();
getchar();
}
```

2. В каждую компонентную функцию вставить строку программы, печатающую, что именно эта компонентная функция вызвана. (В конструктор, например, вставить строку cout<<"\n Это конструктор.";) и объяснить последовательность вызовов компонентных функций при выполнении main().

3. До определить множество операций над матрицами, т.е. написать компонентные функции, реализующие операции ==, !=, > , < , A * const, cout<<A. Считать, что матрица A > матрицы B, если max $|a_{ij}|$ > max $|b_{ij}|$; i = 1,…,N; j = 1, …, M.
4. Определить новый тип данных, комплексное число, и множество операций над ним.
5. Написать отчет.

Содержание отчета

5. Титульный лист отчета должен содержать название, цель лабораторной работы, группу и фамилию студента, выполнившего её, и фамилию преподавателя, проверившего отчет.
6. Выполненное домашнее задание.
7. Тексты программ, написанных при выполнении 3 и 4 пунктов лабораторного задания.

Контрольные вопросы.

1. Для чего используется конструктор копии? В каких случаях наличие конструктора копии в классе обязательно, для правильной работы программы, а в каких нет?
2. Как работает компонентная функция operator= ? В каких случаях наличие компонентной функции operator= в классе обязательно, для правильной работы программы, а в каких нет?
3. Конструктор преобразования типа. Приведите примеры явного и неявного вызова конструктора преобразования типа для класса комплексное число.
4. Перегрузка операций. Запишите множество перегружаемых и не перегружаемых операций языка C++. Расскажите о свойствах перегруженных операций.
5. Напишите класс vector (вектор) и определите над ним множество операций { +, -, ==, =, !=, скалярное произведение}.
6. Напишите класс str (строка) и определите над ним множество операций { +, -, ==, =, !=, <, >}.
7. Напишите класс круг и определите над ним множество операций {==, !=, <,> } (по площади).
8. Напишите класс квадрат и определите над ним множество операций {==, !=, <,> } (по площади).
9. Напишите класс список, обеспечивающий: вставку, удаление, доступ к элементу в списке.
10. Деструктор, его назначение и сигнатура.
11. Добавьте в класс Matr статическую компонентную функцию и статическое поле данных, в котором хранится количество объектов типа Matr.
12. Напишите компонентные функции, обеспечивающие сохранение и восстановление объекта типа Matr из файла.
13. В чем состоит идея инкапсуляции данных? В чем её основное достоинство? С помощью каких средств языка C++ она реализуется в программе? Приведите примеры класса, написанного с инкапсулированными данными и нет.

Лабораторная работа N 9
Работа с иерархическими структурами (бинарные деревья)

Цель работы: Получение навыка работы с иерархическими структурами.

Домашнее задание

1.1. Тщательно изучите листинг программ №1. Нарисуйте блок-схему функции iTakeOut.
1.2. Составьте произвольный список из учеников вашей группы с вашей фамилией в начале списка. Нарисуйте дерево, использую составленный список.

Лабораторное задание

2.1. Наберите программу №1 (комментарии можно не набирать).

```
#include "stdafx.h"
#include <stdlib.h>
# include <string.h>
# include <stdio.h>
# include <malloc.h>
# include <conio.h>


# define STAFF struct sStaffType
STAFF   // Учебно-вспомогательный персонал
```

```
{
   int iYearsOfService;   // Время работы (лет)
   float fHourlyWage;     // Почасовая оплата
};
# define STUDENT struct sStudentType
STUDENT
{
   float fGradePtAverage;   // Средний рейтинг
   int iLevel;              // Год обучения
};
# define PROFESSOR struct sProfType
PROFESSOR
{
   int iDepartmentNumber;      // Номер кафедры
   float fAnnualSalary;        // Годовая зарплата
};
# define NODE_TYPE enum eNodeType
typedef NODE_TYPE {student, professor, staff};
# define TREE struct sTree
TREE
{
   char sLastName[15];   // Фамилия
   char sFirstName[15];  // Имя
   int iAge;             // Возраст
   TREE *Left, *Right;   // Указатели на левый и правый листья (ветви)
   NODE_TYPE tag;        // описатель типа узла - студент или профессор или УВП
   union
   {
        STUDENT student;
        PROFESSOR professor;
        STAFF staff;
   } uNodeTag;                 // Обьединение, содержащее информацию по
};                             // студенту или сотруднику университета
extern void Insert(TREE **root, TREE *item);  // Вставить в дерево новый элемент item
extern void Display(TREE *root);  // Показать содержимое дерева
extern int iIsPresent(TREE *root, TREE *item); // Содержится ли информация item в
дереве?
extern int iTakeOut(TREE **root, TREE *item);  // Удалить элемент item из дерева
extern void Destroy(TREE *root); // Уничтожить дерево
static TREE* CreateNode(TREE* item) // Создать элемент item
{
   TREE* node;
   node = (TREE*) malloc(sizeof(TREE));
   *node = *item;
   return node;
}
void Destroy(TREE* root)  // Уничтожить дерево
{
   if (root)    // Обратите особое внимание на рекурсивную работу этой функции
    {
        Destroy(root->Left);
        Destroy(root->Right);
        free(root); // Освободить память, которая была выделена для узла дерева
    }
   root = 0;
}
int iTakeOut(TREE** root, TREE* item) // Удалить элемент item из дерева
```

```
{
   TREE *previous = 0,    // Предыдущий узел дерева
          *present = *root,  // Текущий узел дерева
          *replace,  // Вспомогательные узлы,
          *s,        // используемые для перемещения элементов
          *parent;   // дерева после удаления найденного узла
          int iFound = 0;
   while (present && !iFound) // Пока не будет найден элемент item
   {
         if(strcmp(item->sLastName, present->sLastName) == 0)
            iFound = 1;  // Информация по человеку с таким именем и фамилией есть в
дереве
         else
          {
            previous = present;
            // Если ASCII представление фамилии из item меньше ASCII кода фамилии
            // из текущего узла дерева (present), то перейти к просмотру левого
            // узла (листа) относительно present, иначе - правого
            if(strcmp(item->sLastName, present->sLastName) < 0)
                 present = present->Left;
            else
                 present = present->Right;
          }
   }
   if (iFound) // если item присутствует в дереве
   {
         if (present->Left == 0)  // Если найденный элемент не имеет ветви слева
            replace = present->Right;
         else
         {
            if (present->Right == 0) // Если найденный элемент не имеет ветви справа
                 replace = present->Left;
            else  // Если удаляемый элемент имеет и левую и правую ветвь (листья)
            {
                  parent = present;
                  replace = present->Right;
                  s = replace->Left;
                  // Теперь необходимо подвинуть все элементы ветви, чтобы избежать
разрыва
                  // дерева при удалении найденного элемента (present)
                  while (s != 0) // Пока не будет достигнут крайний левый лист в
рассматриваемой ветви
                  {  // Спускаемся вниз дерева по левой ветви
                        parent = replace;
                        replace = s;
                        s = replace->Left;
                  }
                  if (parent != present) // Есть левая ветвь от правой ветви от
найденного элемента
                  {
                        parent->Left = replace->Right;   // Правую подветвь
переносимого элемента сделать левой подветвью предыдущего
                        replace->Right = present->Right; // Переместить элемент на
место удаляемого
                  }
                  replace->Left = present->Left; // Переместить левую ветвь
            }
```

```
          }
          if (previous == 0) // Элемент лежит сразу же за корнем дерева
             *root = replace;
          else
             if(present == previous->Left) // Предыдущий спуск был по левой ветви
                  previous->Left = replace;
             else                          // Предыдущий спуск был по правой ветви
                  previous->Right = replace;
          free (present);  // Удалить найденный элемент
    }
    return iFound; // 1 - если элемент был удален, 0 - если такого элемента в дереве
не было
}
void Insert(TREE **root, TREE *item ) // Вставить элемент item в дерево
{
    TREE *parent = 0,
    // current (текущий) указатель на дерево указывает на его вершину (корень)
    *current = *root;
    TREE *new_node;   // Новый узел
    int iFound = 0;
    while (current && !iFound) // Пока элемент item не найден
    {
         if (strcmp(item->sLastName, current->sLastName) == 0) iFound = 1;
         else
         {
            parent = current;
            if (strcmp(item->sLastName, current->sLastName) < 0)
                 current = current->Left; // перемещаться по левой ветви
            else
                 current = current->Right; // перемещаться по правой ветви
         }
    }
    if (iFound == 0)
     {
         if (parent == 0) // в дереве нет еще элементов - создаем его
         {
            *root = CreateNode(item); // создать узел
            (*root)->Left = (*root)->Right = 0;
         }
         else // Вставить узел в дерево
         {
            new_node = CreateNode(item);
            new_node->Left = new_node->Right = 0;
            if (strcmp(item->sLastName, parent->sLastName) < 0)
                 parent->Left = new_node;
            else
                 parent->Right = new_node;
         }
     }
}
void Display(TREE *root)  // Показать дерево
{
    if(root)  // Обратите внимание также на рекурсивный обход дерева
    {
         Display(root->Left);// показать вначале левую ветвь (лист) дерева
         printf("\n%s, %s", root->sLastName, root->sFirstName);
         printf("\n Old - %d", root->iAge);
```

```
        switch(root->tag)  // Обратите внимание на использование в
        {                  // конструкции switch элементов перечислимого
           case student:   // (enum) типа
             printf("\nReyting: %.2f",
                             root->uNodeTag.student.fGradePtAverage);
             printf("\nKurs: %d\n", root->uNodeTag.student.iLevel);
           break;
           case professor:
             printf("\nNumber of kafedra: %d",
                            root->uNodeTag.professor.iDepartmentNumber);
             printf("\nYear selary: %.2f\n",
                           root->uNodeTag.professor.fAnnualSalary);
           break;
           case staff:
             printf("\n Time of work(year): %d",
                           root->uNodeTag.staff.iYearsOfService);
             printf("\nSelary of oure: %.2f\n",
                            root->uNodeTag.staff.fHourlyWage);
         }
        Display(root->Right); // Вывести информацию о содержимом правого узла
    }
}
int iIsPresent(TREE *root, TREE *item)
{
   TREE *current = root;  // Устанавливаем указатель на вершину (корень) дерева
   int iFound = 0;
   while (current && !iFound) // пока элемент item не найден
    {
        if (strcmp(item->sLastName, current->sLastName) == 0) iFound = 1;
        else
        {  // Если ASCII код фамилии из item меньше ASCII кода из текущего узла
(current)
           if (strcmp(item->sLastName, current->sLastName) < 0)
                current = current->Left;   // то перейти к рассмотрению левого узла
           else
                current = current->Right; // иначе перейти к рассмотрению правого
узла
        }
    }
   return iFound;  // Если не найден - 0, если найден - 1
}
TREE* sMyTree;
void main()
{
   // Выделяем память на три узла дерева
   TREE* item1 = (TREE*) malloc(sizeof(TREE));
   TREE* item2 = (TREE*) malloc(sizeof(TREE));
   TREE* item3 = (TREE*) malloc(sizeof(TREE));
   // Инициализация первого элемента
   strcpy(item1->sLastName, "Fyfikov");
   strcpy(item1->sFirstName, "Ziberman");
   item1->iAge = 32;
   item1->tag = staff;
   item1->uNodeTag.staff.iYearsOfService = 3;
   item1->uNodeTag.staff.fHourlyWage = 5.25;
   // Вставить элемент в дерево
   Insert(&sMyTree, item1);
```

```
    strcpy(item2->sLastName, "Vibigalo");
    strcpy(item2->sFirstName, "Ivanov");
    item2->iAge = 56;
    item2->tag = professor;
    item2->uNodeTag.professor.iDepartmentNumber = 7;
    item2->uNodeTag.professor.fAnnualSalary = 15321.0;
    Insert(&sMyTree, item2);
    strcpy(item3->sLastName, "Sidorov");
    strcpy(item3->sFirstName, "Antonov");
    item3->iAge = 18;
    item3->tag = student;
    item3->uNodeTag.student.iLevel = 1;
    item3->uNodeTag.student.fGradePtAverage = 0.75;
    Insert(&sMyTree, item3);
    Display(sMyTree); // Показать дерево
    getchar();
    if(iIsPresent(sMyTree, item2))
       printf("\n 2- element out of tree\n");
    else
       printf("\n element out of tree\n");
    getchar();
    iTakeOut(&sMyTree, item1);
    Display(sMyTree);
    getchar();
    iTakeOut(&sMyTree, item2);
    Display(sMyTree);
    getchar();
    iTakeOut(&sMyTree, item3);
    Display(sMyTree);
    getchar();
    printf("\n");
}
```

2.3. Запустите программу и проанализируйте результаты ее работы.
2.4. Измените программу таким образом, чтобы в дерево можно было бы вставлять информацию о студентах колледжа.
2.5 Измените программу так, чтобы в дерево можно было вставлять однофамильцев.
2.6 Модифицируйте класс sTree так, чтобы его объекты можно было вставлять в контейнер set библиотеки STL. Продемонстрируйте примеры вставки.

Содержание отчета

1. Титульный лист отчета должен содержать название, цель лабораторной работы, группу и фамилию студента, выполнившего её, и фамилию преподавателя, проверившего отчет.
2. Выполненное домашнее задание.
3. Тексты программ, написанных при выполнении 2.4, 2.5 и 2.6 пунктов лабораторного задания.

3. Вопросы к защите

3.1. Использую библиотеку STL, напишите класс вектор целых чисел.
3.2. Использую библиотеку STL, напишите класс очередь вещественных чисел.
3.3. Использую библиотеку STL, напишите класс список символьных переменных.
3.4. Использую библиотеку STL, напишите класс множество символьных переменных.
3.5. Использую библиотеку STL, напишите класс мультимножество символьных переменных.
3.6. Нарисуйте пример двоичного дерева поиска и покажите, как изменяется его структура при удалении из него элемента.

3.7. Нарисуйте пример двоичного дерева поиска и покажите, как изменяется его структура при вставке в него элемента.
3.8. Напишите функцию сохранения и восстановления дерева из файла.
3.9. Напишите функцию, позволяющую найти и отредактировать содержимое любого узла в дереве.
3.10. Напишите функцию, выводящую на экран информацию о узлах дерева удовлетворяющих заданному с клавиатуры пользователем диапазону (например, вывести все узлы начинающиеся с букв от B до E).
3.11. Модифицируйте программу так, чтобы фамилия и имя в узле дерева заносились не в статический, а динамический массив памяти.

Лабораторная работа N 10
Наследование классов в языке C++

Цель работы: Получение навыка работы с наследованием классов в языке C++

Домашнее задание

1.1. Тщательно изучите листинг программ №1. Постройте графы иерархии классов.
1.2 Письменно поясните термины инкапсуляция и интерфейс, наследование интерфейса и наследование реализации, композиция и агрегация в языке C++. С помощью каких средств языка они реализованы.

Лабораторное задание

2.1. Наберите программу №1.

```
// Figura.cpp:
#include "stdafx.h"
#include "afxwin.h"
#include "iostream"
using namespace std ;

class Figure{
	static HWND hwnd;
protected:
	static HDC hdc;
public:
	Figure(){/*cout<<"\n Figure()";*/}
	void show(){}
	void hide(){}
	void move(int x, int y){}
	static void
InitGraphic(){hwnd=FindWindow(_T("ConsoleWindowClass"),_T("C:\\Windows\\system32\\cmd.exe"));hdc=
GetWindowDC(hwnd);}
	static void CloseGraphic(){ReleaseDC(hwnd, hdc); CloseHandle(hwnd);}
	~Figure(){/*cout<<"\t ~Figure()";*/}
};
HWND Figure::hwnd = 0;
HDC Figure::hdc = 0;

class Square: public Figure {
	POINT pt[5];
public:
	Square(POINT* p){
		for(int i =0 ; i <5; i++){pt[i].x = p[i].x;pt[i].y = p[i].y;}
	}
	void show(){
		CPen pen(PS_SOLID,2,RGB(255,0,0));
		SelectObject(hdc,pen);
		Polyline(hdc,pt,5 );
	}
	void hide(){
		CPen pen(PS_SOLID,2,RGB(0,0,0));
```

```
                SelectObject(hdc,pen);
                Polyline(hdc,pt,5 );
        }
        void move(int x, int y){for(int i = 0; i<5;i++){ pt[i].x+=x;pt[i].y+=y;} }
        ~Square(){/*cout<<"\t ~Square()";*/}
};

class ClsEllipse: public Figure {


public:
        CPoint pt1,pt2;
        ClsEllipse(){/*cout<<"\t ClsEllipse()";*/
                pt1.x=100;              pt1.y=100;
                pt2.x=200;              pt2.y=200;
        }
        void show(){
                CPen pen(PS_SOLID,2,RGB(0,255,0));
                SelectObject(hdc,pen);
                Arc(hdc,pt1.x,pt1.y,pt2.x,pt2.y,100,200,0,100);
        }
        void hide(){
                CPen pen(PS_SOLID,2,RGB(0,0,0));
                SelectObject(hdc,pen);
                Arc(hdc,pt1.x,pt1.y,pt2.x,pt2.y,100,200,0,100);
        }
        void move(int x, int y){ pt1.x+=x,pt1.y+=y,pt2.x+=x,pt2.y+=y; }
        ~ClsEllipse(){/*cout<<"\t ~ClsEllipse()";*/}
};
//Включение объектов
class MyObject{
        Square sq1, sq2; //Композиция (агрегирование по значению)
        ClsEllipse& elp; //Агрегация (агрегирование по ссылке)
public:
        MyObject(const Square& p1,const Square& p2,ClsEllipse& el):sq1(p1),sq2(p2),
elp(el){/*cout<<"\t MyObject()";*/}
        void show(){sq1.show(); sq2.show();elp.show();}
        void move(int x, int y){sq1.move(x,y); sq2.move(x,y);elp.move(x,y);}
        void hide(){sq1.hide(); sq2.hide(); elp.hide();}
        ~MyObject(){/*cout<<"\n ~MyObject()";*/}
};
//Множественное наследование
class Heir: Square, ClsEllipse{
public:
        Heir(POINT *p):Square(p),ClsEllipse(){/*cout<<"\t Heir()";*/ }
        void show(){Square::show(); ClsEllipse::show();}
        void move(int x, int y){Square::move(x,y); ClsEllipse::move(x,y);}
        void hide(){Square::hide(); ClsEllipse::hide();}
        ~Heir(){/*cout<<"\n ~Heir()";*/}
};

void ShowMyObject(MyObject obj){
        for(int i = 0 ; i <100 ; i++){obj.show();  Sleep(24);  obj.hide();  obj.move(4,0);}
}
void main(){
        POINT pt1[5];
        pt1[0].x = 40;pt1[0].y=40;
        pt1[1].x = 40;pt1[1].y=140;
```

```
	pt1[2].x = 140;pt1[2].y=140;
	pt1[3].x = 140;pt1[3].y=40;
	pt1[4].x = 40;pt1[4].y=40;

	Figure::InitGraphic();
	{
		Square sq1(pt1);	ClsEllipse elp;
		for(int i = 0 ; i <100 ; i++){	sq1.show();elp.show();  Sleep(24);  sq1.hide();
elp.hide();  sq1.move(1,1); elp.move(2,2);}
	}
	ClsEllipse elp;
	Square sq2(pt1);
	sq2.move(20,20);
	MyObject obj(pt1, sq2, elp);
	getchar();
	ShowMyObject(obj);
	{
		Heir hr(pt1);
		getchar();
		for(int i = 0 ; i <100 ; i++){hr.show();  Sleep(24);  hr.hide();  hr.move(0,3);}
	}
	Figure::CloseGraphic();
}
```

2.2. Определите классы линия, треугольник, трапеция.
2.3. Получите у преподавателя рисунок и, используя разработанные классы, нарисуйте его на экране, применив множественное наследование и включение.
2.4 Переопределить один из классов с помощью указателей.

Содержание отчета

3.1 Титульный лист отчета должен содержать название, цель лабораторной работы, группу и фамилию студента, выполнившего её, и фамилию преподавателя, проверившего отчет.
3.2 Выполненное домашнее задание.
3.3 Тексты программ, написанные при выполнении 2.2, 2.3 и 2.4 пунктов лабораторного задания.

4. Вопросы к защите

4.1. Расскажите о методах декомпозиции системы.
4.2. Приведите примеры иерархичных систем.
4.3. Расскажите о переопределении данных и методов базового класса в производном классе.
4.4. Раскройте понятия инкапсуляция и интерфейс класса. С помощью каких средств языка они реализованы.
4.5. Реализуйте класс Square, используя динамическую память.
4.6. Реализуйте класс ClsEllipse, используя динамическую память.
4.7. Реализуйте класс MyObject, используя динамическую память.
4.8. Напишите классы абитуриент и производный от него студент. Поясните механизм наследования.
4.9. Напишите классы университет и факультет и на их примере поясните, что такое композиция.
4.10. Напишите классы преподаватель и факультет и на их примере поясните, что такое агрегация.

Лабораторная работа N 11
Абстрактные классы и виртуальные функции C++

Цель работы: Получение навыка работы с виртуальными функциями и абстрактными классами в языке C++

Домашнее задание

1.1. Тщательно изучите листинг программ №1. Постройте граф иерархии классов.

1.2 Поясните письменно термины: инкапсуляция, интерфейс, наследование, полиморфизм, агрегирование, делегирование в отношении классов языка С++. С помощью каких средств языка они реализованы.

Лабораторное задание
2.1. Наберите программу №1.

```
// В свойствах проекта в опциях по C/C++ ->Библиотека времени выполнения; установить /MTd
//В свойствах проекта Компоновщик->Ввод->Дополнительные зависимости; поставьте  на первое место
uafxcwd.lib
#include "stdafx.h"
#include "afxwin.h"
#include "iostream"
using namespace std ;

HWND hwnd = 0;
HDC  hdc = 0;
void InitGraphic(){
system("mode con cols=168 lines=55");     system("pause >> void");
hwnd=FindWindow(_T("ConsoleWindowClass"),_T("C:\\Windows\\system32\\cmd.exe"));hdc=GetWindowDC(hw
nd);
}
void CloseGraphic(){ReleaseDC(hwnd, hdc); CloseHandle(hwnd);}

//------------------------------------------------------------------------- IFigure
class IFigure{ //интерфейсный класс
protected:
	int fMove; //0 - фигура двигается; 1 - фигура мигает на месте; 2 - фигура стоит на месте;
	int fClr; //0 - фигура цвет не меняет; 1 - фигура меняет цвет
public:
	IFigure(): fMove(0), fClr(0){/*cout<<"\n IFigure()";*/}
	virtual void show()=0;
	virtual void hide()=0;
	virtual void move(int x, int y)=0;
};

//-------------------------------------------------------------------------Square
class Square:virtual public IFigure {
	POINT pt[5];
	COLORREF color;
public:
	Square(POINT* p): color(RGB(255,0,0)){ for(int i =0 ; i <5; i++) pt[i] = p[i]; }
	void SetColor(COLORREF cl){color = cl;}
	void show(){
		CPen pen(PS_SOLID,2,color);
		SelectObject(hdc,pen);
		Polyline(hdc,pt,5 );
	}
	void hide(){
		CPen pen;
		pen.CreatePen(PS_SOLID,2,RGB(0,0,0));
		SelectObject(hdc,pen);
		Polyline(hdc,pt,5 );
	}
	void move(int x, int y){for(int i = 0; i<5;i++){ pt[i].x+=x;pt[i].y+=y;} }
	virtual ~Square(){/*cout<<"\t ~Square()";*/}
};
//-------------------------------------------------------------------------ClsEllipse
class ClsEllipse: virtual  public IFigure {
```

```
	CPoint pt1,pt2;
public:
	ClsEllipse():pt1(100,100),pt2(200,200) {}
	virtual void show() {
		CPen pen(PS_SOLID,2,RGB(0,255,0));
		SelectObject(hdc,pen);
		Arc(hdc,pt1.x,pt1.y,pt2.x,pt2.y,100,200,0,100);
	}
	virtual void hide() {
		CPen pen(PS_SOLID,2,RGB(0,0,0));
		SelectObject(hdc,pen);
		Arc(hdc,pt1.x,pt1.y,pt2.x,pt2.y,100,200,0,100);
	}
	virtual void move(int x, int y)  { pt1.x+=x,pt1.y+=y,pt2.x+=x,pt2.y+=y; }
	virtual ~ClsEllipse(){/*cout<<"\t ~ClsEllipse()";*/}
};
//-------------------------------------------------------------------------Rectan
class Rectan: public IFigure {
	Square* pSq;
public:
	//дописать operator=
	virtual void show(){pSq->show();}//Делегирование
	virtual void move(int x, int y){pSq->move(x,y);}//Делегирование
	virtual void hide(){pSq->hide();}//Делегирование
	void SetColor(COLORREF cl){pSq->SetColor(cl);}
	Rectan (Square& p){pSq = new Square(p);}
	virtual ~Rectan(){delete pSq;}
};
//-------------------------------------------------------------------------DrowTxt
class DrowTxt{
	CString str;
public:
	DrowTxt(CString s):str(s){}
	void show(){
		CDC* pCDC = CDC::FromHandle(hdc);
		pCDC->SetTextColor(RGB(255,0,0));
		pCDC->SetBkColor(RGB(0,0,0));
		pCDC->TextOutW(300,100,str); pCDC->TextOutW(0,0," ");
	}
};

//-------------------------------------------------------------------------Heir
class Heir: public  Square, public ClsEllipse{ //Виртуальный базовый класс
public:                                         //(Множественное наследование)
	Heir(POINT *p):Square(p),ClsEllipse(){/*cout<<"\t Heir()";*/ }
	void show(){Square::show(); ClsEllipse::show();}
	void move(int x, int y){Square::move(x,y); ClsEllipse::move(x,y);}
	void hide(){Square::hide(); ClsEllipse::hide();}
	virtual ~Heir(){/*cout<<"\n ~Heir()";*/}
};
 //-------------------------------------------------------------------------RecordPlayer
class RecordPlayer{ //Чтобы воспользоваться классом, объекты должны поддерживать интерфейс
IFigure
	IFigure**pFig;//Массив указателей IFigure*
	int n; //Текущее количество указателей в массиве
	int N; //Размерность массива
public:
	void Insert(IFigure* pF){if (n<N) pFig[n++] =pF; }
```

```
	RecordPlayer(int Nfig): N(Nfig), n(0) {  pFig = new IFigure*[N]; }
	virtual void show(){  for(int i = 0; i < n; i++) pFig[i]->show();  }//Полиморфизм
	virtual void hide(){  for(int i = 0; i < n; i++) pFig[i]->hide();  }//Полиморфизм
	virtual void move(int x, int y){  for(int i = 0; i < n; i++) pFig[i]->move(x,y);
}//Полиморфизм
	void PlayMyObject(int x, int y){  for(int i = 0; i <150 ; i++){show();Sleep(24);hide();
move(x,y);} show();}
	virtual ~RecordPlayer(){delete []pFig;}
};

void main(){

	POINT pt1[5];
	pt1[0].x = 40;pt1[0].y=40;
	pt1[1].x = 40;pt1[1].y=140;
	pt1[2].x = 140;pt1[2].y=140;
	pt1[3].x = 140;pt1[3].y=40;
	pt1[4].x = 40;pt1[4].y=40;

	InitGraphic();

	DrowTxt dtxt("Привет");
	dtxt.show();
	getchar();

	Heir hr(pt1);
	for(int i = 0 ; i <100 ; i++){hr.show();  Sleep(24);  hr.hide();  hr.move(0,3);}
	getchar();

	ClsEllipse elp;
	Square sq1(pt1), sq2(pt1), sq3(pt1);
	sq1.SetColor(RGB(255,255,0));  sq2.SetColor(RGB(0,255,0));
	sq3.SetColor(RGB(0,0,255));     hr.SetColor(RGB(0,255,255));
	sq2.move(20,20);     sq3.move(40,30);     hr.move(0,-150);
	Rectan rec(sq3);
	RecordPlayer RPlayer(5);
	RPlayer.Insert(&elp);
	RPlayer.Insert(&sq1);
	RPlayer.Insert(&sq2);
	RPlayer.Insert(&rec);
	RPlayer.Insert(&hr);
	RPlayer.PlayMyObject(3,0);

	getchar();
	CloseGraphic();
}
```

2.2 Доопределите класс Rectan.

2.3 Модифицируйте класс ClsEllipse так, чтобы параметры эллипса можно было задавать через конструктор, и объекты этого класса могли менять свой цвет.

2.4 Модифицируйте класс DrowTxt так, чтобы он мог быть использован классом RecordPlayer для воспроизведения на экране.

2.5 Напишите класс «текст в прямоугольнике» так, чтобы он мог быть использован классом RecordPlayer для воспроизведения на экране.

2.6 Модифицируйте программу так, чтобы объекты на экране с помощью класса RecordPlayer могли двигаться, оставаться неподвижными, мигать, изменять свой цвет.

2.7 Получите рисунок у преподавателя и напишите программу, рисующую его на экране с помощью класса RecordPlayer.

Содержание отчета

3.1 Титульный лист отчета должен содержать название, цель лабораторной работы, группу и фамилию студента, выполнившего её, и фамилию преподавателя, проверившего отчет.
3.2 Выполненное домашнее задание.
3.3 Тексты классов, написанных при выполнении 2.2 – 2.7 пунктов лабораторного задания.

4. Вопросы к защите

4.1. Расскажите о виртуальных функциях и полиморфизме.
4.2. Что такое абстрактные классы, для чего они нужны?
4.3. Расскажите о переопределении данных и методов базового класса в производном классе.
4.4. Раскройте понятия инкапсуляция и интерфейс класса. С помощью каких средств языка они реализованы.
4.5. Реализуйте класс круг, вписанный в квадрат, и выведите его на экран с помощью класса RecordPlayer.
4.6. Реализуйте класс ClsEllipse, используя динамическую память.
4.7. Реализуйте класс Square, используя динамическую память.
4.8. Напишите класс «магазин», который работает с объектами абстрактного класса «товар». Определите 2-3 класса, производных от класса «товар». Продемонстрируйте примеры полиморфного поведения.
4.9. Напишите класс «вагон», который работает с объектами абстрактного класса «груз». Определите 2-3 класса, производных от класса «груз». Продемонстрируйте примеры полиморфного поведения.

Лабораторная работа N 12
Библиотека STL

Цель работы: Получение навыка работы с библиотекой STL

Домашнее задание

1.1. Тщательно изучите листинг программ №1. Опишите, что делает класс CountedPtr.
1.2 Письменно в отчёте приведите все компонентные функции одной из коллекций из библиотеки STL.

Лабораторное задание

2.1. Наберите программу №1.

```
#include "stdafx.h"
#include <iostream>
#include <list>
#include <deque>
# include <set>
#include <algorithm>
using namespace std;

/* Класс, обеспечивающий семантику подсчёта ссылок.
*  Объект, на который ссылается указатель, автоматически
*   уничтожается при удалении последнего экземпляра CountedPtr
*   для данного объекта.
*/
template <class T>
class CountedPtr {
private:
	T* ptr;		// Указатель на значение
	long* count;  // Количество владельцев (общие данные)
	public:
	// Инициализация объекта существующим указателем
	// - указатель p должен быть получен в результате вызова new
	explicit CountedPtr(T* p = 0)
		: ptr(p), count(new long(1)) {
	}
```

```
	// Копирующий указатель (увеличивает счётчик владельцев)
	CountedPtr(const CountedPtr<T>& p) throw()
		: ptr(p.ptr), count(p.count) {
		++*count;
	}
	// Деструктор (уничтожает объект, если владелец был последним)
	~CountedPtr() throw() {
		dispose();
	}
	// Присваивание (перевод указателя на новый объект)
	CountedPtr<T>& operator= (const CountedPtr<T>& p) throw() {
		if (this != &p) {
			dispose();
			ptr = p.ptr;
			count = p.count;
			++*count;
		}
		return *this;
	}

	// Доступ к объекту, на который ссылается указатель
	T& operator*() const throw() {
		return *ptr;
	}
	T* operator->() const throw() {
		return ptr;
	}
private:
	void dispose() {
		if (--*count == 0) {
			delete count;
			delete ptr;
		}
	}
};


struct comp {
	int* Re, *Im;
	comp() { Re = new int; Im = new int; ;*Re = 0; *Im = 0; }
	comp(int r, int i) { Re = new int; Im = new int;*Re = r; *Im = i; }
	~comp() { delete Re; delete Im; }
	const comp& operator=(const comp& T) { // оператор функция=
		*Re = *T.Re; *Im = *T.Im; return *this;
	}
	const comp& operator-() { // оператор функция-
		*Re = -*Re; *Im = -*Im; return *this;
	}
	comp(comp& T) {      Re = new int;     Im = new int;  *Re = *T.Re; *Im = *T.Im;}
	const comp operator*(const comp& T) { // оператор функция*
		comp Rez(0,0);
		*Rez.Re =*Re *  *T.Re - *Im *  *T.Im;
		*Rez.Im = *Re *  *T.Im + *Im *  *T.Re;
		 return Rez;
	}
	double modComp()const {return sqrt(*Re**Re + *Im**Im); }
	bool operator<(const comp T) { // оператор функция<
		if( modComp() < T.modComp()) return 0;//x<x всегда ложно
```

```
		return 1;
	}
	void comp::display() const
	{
		cout << "\n Re = " << *Re << "\t Im = " << *Im;
	}

};

bool operator< (const CountedPtr<comp> p1, const CountedPtr<comp> p2) {
	if (*p1 < *p2) return 0;
	return 1;
}



void printCountedPtr(CountedPtr<comp> elem)
{
	(*elem).display();
}

int main()
{

	// Три разные коллекции
	typedef CountedPtr<comp> IntPtr;
	deque<IntPtr> coll1;
	list<IntPtr> coll2;
	set<IntPtr> coll3;

	/* Вставка общих объектов в коллекции*/

	for (int i = 0; i<5; ++i) {
		IntPtr ptr(new comp(i,i));
		coll1.push_back(ptr); //coll1.push_back(comp(i,i));
		coll2.push_front(ptr);
		coll3.insert(ptr);
	}

	// Вывод содержимого коллекций
	cout<<"\n deque";
	for_each(coll1.begin(),coll1.end(),	printCountedPtr);
	cout << "\n list";
	for_each(coll2.begin(), coll2.end(), printCountedPtr);
	cout << "\n set";
	for_each(coll3.begin(), coll3.end(), printCountedPtr);
	cout << endl << endl;
	/* Модификация значений в разных коллекциях
	* - возведение в квадрат значения в coll1
	* - изменение знака первого значения в coll2
	*/
	*coll1[1] = *coll1[1] * *coll1[1];
//	*coll2.front() = - (*coll2.front());

	// Повторный вывод содержимого коллекций
	cout << "\n\n deque";
	for_each(coll1.begin(), coll1.end(), printCountedPtr);
	cout << "\n list";
```

```
        for_each(coll2.begin(), coll2.end(), printCountedPtr);
        cout << "\n set";
        for_each(coll3.begin(), coll3.end(), printCountedPtr);
        cout << endl;
}
```

2.2. Упростите программу. Перейдите от ссылочной семантики к семантике по значению (класс CountedPtr не нужен). Сравните данные, выводимые на экран, упрощённой программой и исходной.
2.3. Определите свой собственный класс (например: запись в базе данных, строка, вектор, квадрат, круг) и сформируйте коллекции из объектов вашего класса.

Содержание отчета

3.1 Титульный лист отчета должен содержать название, цель лабораторной работы, группу и фамилию студента, выполнившего её, и фамилию преподавателя, проверившего отчет.
3.2 Выполненное домашнее задание.
3.3 Тексты программ, написанные при выполнении 2.2 и 2.3 пунктов лабораторного задания.

4. Вопросы к защите

4.1. Какие типы коллекций вы знаете?
4.2. Сравните коллекции между собой. Какие у каждой из них имеются достоинства и недостатки?
4.3. Что такое итератор? Какие типы итераторов вы знаете?
4.4. Напишите программу реализующую список элементов типа квадрат.
4.5. Напишите программу, реализующую вектор комплексных чисел.
4.6. Напишите программу, реализующую набор строк.
4.7. Напишите программу, реализующую дек элементов типа эллипс.
4.8. Напишите класс «круг» вписанный в «квадрат», используя классы ClsEllipse и Square.
4.9. Напишите класс «текст» вписанный в «круг», используя классы ClsEllipse и DrowTxt.
4.10. Напишите класс «магазин», который работает с объектами абстрактного класса «товар». Определите 2-3 класса, производных от класса «товар». Продемонстрируйте примеры полиморфного поведения.
4.11. Напишите класс «вагон», который работает с объектами абстрактного класса «груз». Определите 2-3 класса, производных от класса «груз». Продемонстрируйте примеры полиморфного поведения.

Литература

1. Подбельский В. В. Стандартный Си++ [Текст] : учеб. пособие для студ. вузов. - М. : Финансы и статистика, 2008. - 687 с. : ил.. - Библиогр.: с. 667-669 (31 назв.). - ISBN 978-5-279-03243-3
2. Максимов М. Н. Язык Си++ как инструмент для моделирования радиотехнических цепей и сигналов [Текст] : курс лекций : для студ. радиотехн. спец. всех форм обуч.. Ч. 1 / ТТИ ЮФУ, РТФ, Каф. ТОР. - Таганрог : Изд-во ТРТУ, 2007. - 159 с.. - Библиогр.: с. 159 (5 назв.)
3. Библиотека STL