ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ



МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

ВЫСШАЯ ШКОЛА ПЕЧАТИ И МЕДИАИНДУСТРИИ

*Институт Принтмедиа и информационных технологий*

*Кафедра Информатики и информационных технологий*

направление подготовки

09.03.02 «Информационные системы и технологии»,

ЛАБОРАТОРНАЯ РАБОТА № 10

Дисциплина: ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ

Тема: Виртуальные и дружественные функции

Выполнил(а): студент(ка) группы 181-722

Голиков Р.О.

(Фамилия И.О.)

Дата, подпись \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ *\_\_\_\_\_\_\_\_\_\_\_*

(Дата) (Подпись)

Проверил: Ктн Доцент Арсентьев Д.А.*\_\_\_\_\_\_\_\_\_\_*

(Фамилия И.О. степень, звание) (Оценка)

Дата, подпись \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ *\_\_\_\_\_\_\_\_\_\_\_*

(Дата) (Подпись)

Замечания: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Москва

2019

**Задание 1**

Пусть имеется та же издательская компания, которая описана в упражнении 1 лабораторной работы по наследованию, которая продает и книги, и аудио версии печатной продукции. Как и в том упражнении, создайте класс publication, хранящий название (фактически, строку) и цену (типа float) публикации. Создайте два порожденных класса: book, в котором происходит изменение счетчика страниц (типа int), и tape, в котором происходит изменение счетчика записанных на кассету минут. Каждый из классов должен иметь метод getdata(), запрашивающий информацию у пользователя, и putdata() для вывода данных на экран.

**Задание 2**

Напишите main(), где создавался бы массив указателей на класс publication. В цикле запрашивайте у пользователя данные о конкретной книге или кассете, используйте new для создания нового объекта book или tape. Сопоставляйте указатель в массиве с объектом. Когда пользователь закончит ввод исходных данных, выведите результат для всех введенных книг и кассет, используя цикл for и единственное выражение.

**Код:**

#include <iostream>

#include <string>

using namespace std;

class publication

{

private:

string title;

float price;

public:

virtual void getdata()

{

cout << "\nEnter title: "; cin >> title;

cout << "Enter price: "; cin >> price;

}

virtual void putdata()

{

cout << "\n\nTitle: " << title;

cout << "\nPrice: " << price;

}

};

class book : public publication

{

private:

int pages;

public:

void getdata()

{

publication::getdata();

cout << "Enter number of pages: "; cin >> pages;

}

void putdata()

{

publication::putdata();

cout << "\nPages: " << pages;

}

};

class tape : public publication

{

private:

float time;

public:

void getdata()

{

publication::getdata();

cout << "Enter playing time: "; cin >> time;

}

void putdata()

{

publication::putdata();

cout << "\nPlaying time: " << time;

}

};

int main()

{

publication\* pubarr[100]; //array of ptrs to pubs

int n = 0; //number of pubs in array

char choice; //user's choice

do {

cout << "\nEnter data for book or tape (b/t)? ";

cin >> choice;

if (choice == 'b') //make book object

pubarr[n] = new book; // put in array

else //make tape object

pubarr[n] = new tape; // put in array

pubarr[n++]->getdata(); //get data for object

cout << " Enter another (y/n)? "; //another pub?

cin >> choice;

} while (choice == 'y'); //cycle until not 'y'

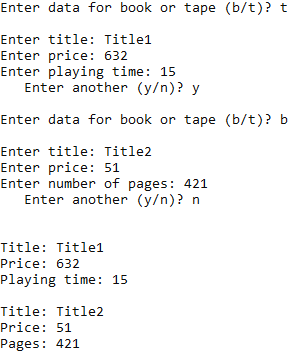
for (int j = 0; j < n; j++) //cycle thru all pubs

pubarr[j]->putdata(); //print data for pub

cout << endl;

return 0;

}

**Вывод:**  


**Задание 3**

В классе Distance, как показано в примерах ниже, создайте перегружаемую операцию умножения \*, чтобы можно было умножать два расстояния. Сделайте эту функцию дружественной, тогда можно будет использовать выражение типа Wdist = 7.5 \* dist2. Вам понадобится конструктор с одним аргументом для перевода величин из формата чисел с плавающей запятой в формат Distance. Напишите какой-либо main() на свое усмотрение для того, чтобы несколькими способами проверить работу этой перегружаемой операции.

**Код:**

#include <iostream>

using namespace std;

class Distance //English Distance class

{

private:

int feet;

float inches;

public:

Distance() //constructor (no args)

{

feet = 0; inches = 0.0;

}

Distance(float fltfeet) //constructor (one arg)

{ //feet is integer part

feet = static\_cast<int>(fltfeet);

inches = 12 \* (fltfeet - feet); //inches is what's left

} //constructor (two args)

Distance(int ft, float in) : feet(ft), inches(in)

{ }

void showdist() //display distance

{

cout << feet << "\'-" << inches << '\"';

}

friend Distance operator \* (Distance, Distance); //friend

};

Distance operator \* (Distance d1, Distance d2)

{

float fltfeet1 = d1.feet + d1.inches / 12; //convert to float

float fltfeet2 = d2.feet + d2.inches / 12;

float multfeet = fltfeet1 \* fltfeet2; //find the product

return Distance(multfeet); //return temp Distance

}

int main()

{

Distance dist1(3, 6.0); //make some distances

Distance dist2(2, 3.0);

Distance dist3;

dist3 = dist1 \* dist2; //multiplication

dist3 = 10.0 \* dist3; //mult and conversion

//display all distances

cout << "\ndist1 = "; dist1.showdist();

cout << "\ndist2 = "; dist2.showdist();

cout << "\ndist3 = "; dist3.showdist();

cout << endl;

return 0;

}

**Вывод:**  


**Задание 4**

Как уже говорилось, классы можно заставлять вести себя как массивы. Пример ниже показывает один из способов создания такого класса. Взяв за основу приведенную программу, добавьте перегружаемое присваивание и перегружаемый конструктор копирования к классу Array. Затем добавьте к main() выражение Array arr2(arrl); и arr3=arrl; для проверки того, что перегружаемые операции работают. Конструктор копирования должен создать новый объект Array со своим собственным местом в памяти, выделенным для хранения элементов массива. И конструктор копирования, и оператор присваивания должны копировать содержимое старого объекта класса Array в новый.

**Код:**

#include <iostream>

using namespace std;

class Array //models a normal C++ array

{

private:

int\* ptr; //pointer to Array contents

int size; //size of Array

public:

Array(int s) //one-argument constructor

{

size = s; //argument is size of Array

ptr = new int[s]; //make space for Array

}

Array(const Array &arr)

{

size = arr.size;

ptr = new int[size];

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

\*(ptr + i) = \*(arr.ptr + i);

}

~Array() //destructor

{

delete[] ptr;

}

int& operator [] (int j) //overloaded subscript operator

{

return \*(ptr + j);

}

Array& operator = (const Array &arr)

{

\*this = Array(arr);

return \*this;

}

};

int main()

{

const int ASIZE = 10;

Array arr1(ASIZE);

for (int j = 0; j < ASIZE; j++)

arr1[j] = j \* j;

Array arr2(arr1);

arr2[0] = 1; // Will change only arr2[0]

Array arr3 = arr1;

arr3[0] = 2; // Will change only arr3[0]

for (int j = 0; j < ASIZE; j++)

cout << arr1[j] << ' ';

cout << endl;

for (int j = 0; j < ASIZE; j++)

cout << arr2[j] << ' ';

cout << endl;

for (int j = 0; j < ASIZE; j++)

cout << arr3[j] << ' ';

cout << endl;

return 0;

}

**Вывод:**  


**Задание 5**

Взяв за основу программу из упражнения 1 этой лабораторной, добавьте метод типа bool, называющийся isOveersize(), к классам book и tape. Допустим, книга, в которой больше 800 страниц, или кассета со временем проигрывания более 90 минут, будут считаться объектами с превышением размера. К этой функции можно обращаться из main(), а результат ее работы выводить в виде строки «Превышение размера!» для соответствующих книг и кассет. Допустим, объекты классов book и tape должны быть доступны через указатели на них, хранящиеся в массиве типа publication.

**Код:**

#include <iostream>

#include <string>

using namespace std;

class publication

{

private:

string title;

float price;

public:

virtual void getdata()

{

cout << "\nEnter title: "; cin >> title;

cout << "Enter price: "; cin >> price;

}

virtual void putdata()

{

cout << "\n\nTitle: " << title;

cout << "\nPrice: " << price;

}

virtual bool isOveersize()

{

return false;

}

};

class book : public publication

{

private:

int pages;

public:

void getdata()

{

publication::getdata();

cout << "Enter number of pages: "; cin >> pages;

}

void putdata()

{

publication::putdata(); cout << "\nPages: " << pages;

}

bool isOveersize()

{

return pages > 800;

}

};

class tape : public publication

{

private:

float time;

public:

void getdata()

{

publication::getdata();

cout << "Enter playing time: "; cin >> time;

}

void putdata()

{

publication::putdata(); cout << "\nPlaying time: " << time;

}

bool isOveersize()

{

return time > 90;

}

};

int main()

{

publication\* pubarr[100]; //array of ptrs to pubs

int n = 0; //number of pubs in array

char choice; //user's choice

do {

cout << "\nEnter data for book or tape (b/t)? "; cin >> choice;

if (choice == 'b') pubarr[n] = new book;

else pubarr[n] = new tape; // put in array

do {

pubarr[n]->getdata();

if (pubarr[n]->isOveersize())

cout << "Oversized!\n";

else break;

} while (true);

n++;

cout << " Enter another (y/n)? "; //another pub?

cin >> choice;

} while (choice == 'y'); //cycle until not 'y'

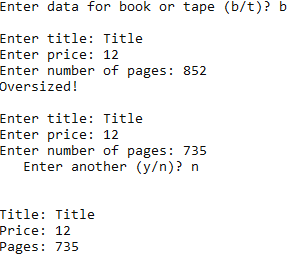
for (int j = 0; j < n; j++) //cycle thru all pubs

pubarr[j]->putdata(); //print data for pub

cout << endl;

return 0;

}

**Вывод:**  


**Задание 6**

Возьмите за основу программу из упражнения 8 лабораторной про перегрузку операций, где было перегружено пять арифметических операций для работы с денежным форматом. Добавьте два оператора, которые не были перегружены в том упражнении:

long double \* bМопеу //умножать число на деньги

long double / bМопеу //делить число на деньги

Эти операции требуют наличия дружественных функций, так как справа от оператора находится объект, а слева — обычное число. Убедитесь, что main() позволяет пользователю ввести две денежные строки и число с плавающей запятой, а затем корректно выполняет все семь арифметических действий с соответствующими парами значений.

**Код:**

#include <iostream>

#include <string>

#include <sstream>

using namespace std;

class money

{

private:

long double \_sum;

public:

money() {}

explicit money(long double sum) : \_sum(sum) {}

long double mstold(string sum)

{

string res = "";

for (int i = 0; i < sum.length(); i++)

{

if (isdigit(sum[i]) || sum[i] == ',')

res += sum[i];

}

return stold(res.c\_str());

}

string ldtoms(long double sum)

{

if (sum > 9999999999999990.00) return "";

ostringstream strs;

strs << fixed << sum;

string ustr = strs.str();

string newstr = "$";

int i, j;

for (i = 0; i < ustr.length() && (ustr[i] == '0' || ustr[i] == '.'); i++);

if (i > 1 && ustr[i - 1] == '.') i -= 2;

if (i == ustr.length()) return "$0,00";

for (j = ustr.length() - 1; j >= 0 && ustr[j] != '.'; j--);

j += 2;

for (; i <= j; i++)

{

if (i != 0 && i < j - 2 && (j - i + 1) % 3 == 0) newstr += '.';

newstr += ustr[i] == '.' ? ',' : ustr[i];

}

return newstr;

}

void getmoney()

{

string str;

cout << "Введите сумму в долларах:\n"; cin >> str;

\_sum = mstold(str);

}

void putmoney()

{

cout << "Денежная сумма равна: " << ldtoms(\_sum) << endl;

}

money operator+(money m)

{

return money(\_sum + m.\_sum);

}

money operator-(money m)

{

return money(\_sum - m.\_sum);

}

money operator\*(long double ld)

{

return money(\_sum \* ld);

}

long double operator/(money m)

{

return \_sum / m.\_sum;

}

money operator/(long double ld)

{

return money(\_sum / ld);

}

friend money operator \* (long double ld, money m);

friend money operator / (long double ld, money m);

};

money operator \* (long double ld, money m)

{

return money(m.\_sum \* ld);

}

money operator / (long double ld, money m)

{

return money(ld / m.\_sum);

}

int main()

{

setlocale(LC\_ALL, "");

money m1, m2;

long double s;

char ch;

do

{

m1.getmoney();

m2.getmoney();

cout << "Введите double: "; cin >> s;

cout << "m1 + m2: ";

(m1 + m2).putmoney();

cout << "m1 - m2: ";

(m1 - m2).putmoney();

cout << "m1 \* s: ";

(m1 \* s).putmoney();

cout << "m1 / m2: ";

cout << (m1 / m2) << endl;

cout << "m1 / s: ";

(m1 / s).putmoney();

cout << "s \* m1: ";

(s \* m1).putmoney();

cout << "s / m1: ";

(s / m1).putmoney();

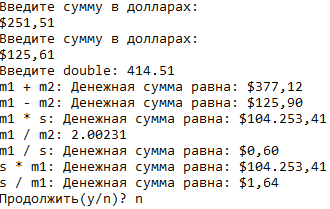
cout << "Продолжить(y/n)? ";

cin >> ch;

} while (ch != 'n');

return 0;

}

**Вывод:**  


**Задание 7**

Как и в предыдущем упражнении, возьмите за основу программу из упражнения 8 лабораторной по перегрузке операций. На этот раз от вас требуется добавить функцию, округляющую значение bМопеу до ближайшего доллара. Как известно, значения, не превышающие $0.49, округляются вниз, а числа от $0.50 и более округляются вверх. Можно использовать библиотечную функцию modfl(): Она разбивает переменную типа long double на целую и дробную части. Если дробная часть меньше 0.50, функция просто возвращает целую часть числа. В противном случае возвращается увеличенная на 1 целая часть. В main() проверьте работоспособность функции путем передачи в нее последовательно значений, одни из которых меньше $0.49, другие — больше $0.50.

**Код:**

#include <iostream>

#include <string>

#include <sstream>

using namespace std;

class money

{

private:

long double \_sum;

public:

money() {}

explicit money(long double sum) : \_sum(sum) {}

long double mstold(string sum)

{

string res = "";

for (int i = 0; i < sum.length(); i++)

{

if (isdigit(sum[i]) || sum[i] == ',')

res += sum[i];

}

return stold(res.c\_str());

}

string ldtoms(long double sum)

{

if (sum > 9999999999999990.00) return "";

ostringstream strs;

strs << fixed << sum;

string ustr = strs.str();

string newstr = "$";

int i, j;

for (i = 0; i < ustr.length() && (ustr[i] == '0' || ustr[i] == '.'); i++);

if (i > 1 && ustr[i - 1] == '.') i -= 2;

if (i == ustr.length()) return "$0,00";

for (j = ustr.length() - 1; j >= 0 && ustr[j] != '.'; j--);

j += 2;

for (; i <= j; i++)

{

if (i != 0 && i < j - 2 && (j - i + 1) % 3 == 0)

newstr += '.';

newstr += ustr[i] == '.' ? ',' : ustr[i];

}

return newstr;

}

// overloaded operators

// ...

// ...

void getmoney()

{

string str;

cout << "Введите сумму в долларах:\n";

cin >> str;

\_sum = mstold(str);

}

void putmoney()

{

cout << "Денежная сумма равна: " << ldtoms(\_sum) << endl;

}

friend money round(money m);

};

money round(money m)

{

long double temp = m.\_sum;

return money(modfl(m.\_sum, &temp) >= 0.5 ? temp + 1 : temp);

}

int main()

{

setlocale(LC\_ALL, "");

money m;

char ch;

do

{

m.getmoney();

round(m).putmoney();

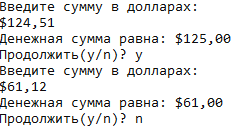
cout << "Продолжить(y/n)? ";

cin >> ch;

} while (ch != 'n');

return 0;

}

**Вывод:**  


**Задание 8**

Помните программу по вычислению математических выражений с дробями из предыдущей лабораторной работы? Попробуйте доработать ее, чтобы она могла вычислять значения математических выражений с рациональными числами, например типа float, а не только с одноразрядными числами:

3.14159 / 2.0 + 75.25 \* 3,333 + 6.02

Во-первых, нужно развить стек до такой степени, чтобы он мог хранить и операторы (типа char), и числа (типа float). Но как, спрашивается, можно хранить в стеке значения двух разных типов? Ведь стек — это, по сути дела, массив. Надо еще учесть, что типы char и float даже не совпадают по размеру! Даже указатели на разные типы данных (char\* и float\*) компилятор не позволит хранить в одном массиве, несмотря на то, что они одинакового размера. Единственный способ хранить в массиве два разных типа указателей — сделать эти типы наследниками одного и того же базового класса. При этом базовому классу даже нет нужды иметь какие-то собственные данные, это может быть абстрактный класс, из которого никакие объекты создаваться не будут.

Конструкторы могут хранить значения в порожденных классах обычным способом, но должна иметься специальная чистая виртуальная функция для того, чтобы извлечь эти значения.

Виртуальные функции базового класса должны быть реализованы во всех порожденных классах, в противном случае классы становятся абстрактными. Таким образом, классу Operand нужна функция getNumber(), несмотря на то, что она фиктивная. Классу Number нужна функция getOperand(), несмотря на то, что она тоже фиктивная.

Поработайте над этим каркасом, сделайте его реально работающей программой, добавив класс Stack, содержащий объекты класса Token, и функцию main(), в которой бы заносились в стек и извлекались из него разные арифметические операторы и числа в формате с плавающей запятой.

**Код:**

#include <iostream>

#include <cstring>

#include <string>

using namespace std;

const int LIMIT = 100;

const int MAX = 100;

class Token {

public:

virtual float getNumber() = 0;

virtual char getOperand() = 0;

};

class Operand : public Token {

private:

char \_oper;

public:

Operand(char oper) : \_oper(oper) {}

char getOperand()

{

return \_oper;

}

float getNumber()

{

return 0;

}

};

class Number : public Token {

private:

float \_fnum;

public:

Number(float fnum) : \_fnum(fnum) {}

float getNumber()

{

return \_fnum;

}

char getOperand()

{

return 0;

}

};

class Stack

{

private:

Token \*\_arr[LIMIT];

int \_next;

public:

Stack() : \_next(0) {}

void push(Token \*el)

{

if (\_next == LIMIT)

{

cout << "\nStack overflow\n";

return;

}

\_arr[\_next++] = el;

}

Token\* pop()

{

if (\_next == 0)

{

cout << "\nStack is empty\n"; exit(0);

}

return \_arr[--\_next];

}

int count()

{

return \_next;

}

};

class express

{

private:

Stack s;

char\* pStr;

int len;

public:

express(char\* ptr)

{

pStr = ptr;

len = strlen(pStr);

}

float operation(float f1, float f2, char op)

{

switch (op)

{

case '+': return f1 + f2; break;

case '-': return f1 - f2; break;

case '\*': return f1 \* f2; break;

case '/': return f1 / f2; break;

default: cout << "\nНеизвестный оператор" << op; exit(1);

}

}

bool isOperand(char ch)

{

return ch == '+' || ch == '-' || ch == '\*' || ch == '/';

}

void parse()

{

char lastop, temp[MAX];

float lastval, secval, buf;

int k, j;

for (j = 0, k = 0; j < len; j++)

{

if (isdigit(pStr[j]) || pStr[j] == ',')

{

temp[k++] = pStr[j];

}

else

{

temp[k] = '\0';

k = 0;

buf = stof(temp);

s.push(new Number(buf));

if (!isOperand(pStr[j]))

{

cout << "\nНеизвестный символ";

exit(1);

}

if (s.count() == 1)

s.push(new Operand(pStr[j]));

else

{

lastval = s.pop()->getNumber();

lastop = s.pop()->getOperand();

if ((pStr[j] == '\*' || pStr[j] == '/') && (lastop == '+' || lastop == '-'))

{

s.push(new Operand(lastop));

s.push(new Number(lastval));

}

else

{

secval = s.pop()->getNumber();

s.push(new Number(operation(secval, lastval, lastop)));

}

s.push(new Operand(pStr[j]));

}

}

}

temp[k] = '\0';

buf = stof(temp);

s.push(new Number(buf));

}

float solve()

{

float lastval, secval;

char oper, oper2;

while (s.count() > 2)

{

lastval = s.pop()->getNumber();

oper = s.pop()->getOperand();

secval = s.pop()->getNumber();

if (s.count() != 0)

{

oper2 = s.pop()->getOperand();

if (oper2 == '-')

{

secval \*= -1;

oper2 = '+';

}

s.push(new Operand(oper2));

}

s.push(new Number(operation(secval, lastval, oper)));

}

return s.pop()->getNumber();

}

};

int main()

{

setlocale(LC\_ALL, "");

char ans;

char string[MAX];

cout << "\nВведите арифметическое выражение в виде 2+3\*4/3-2"

"\nНе используйте пробелы и скобки";

do

{

cout << "\nВыражение: "; cin >> string;

express\* eptr = new express(string);

eptr->parse();

cout << "\nРезультат: "

<< eptr->solve();

delete eptr;

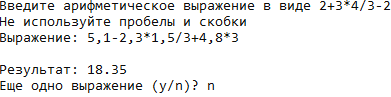
cout << "\nЕще одно выражение (y/n)? ";

cin >> ans;

} while (ans == 'y');

return 0;

}

**Вывод:**  


**Задание 9**

Пусть есть классы, данные ниже. Предположим, что любая лошадь, которая на скачках к середине дистанции находится впереди всех, становится практически непобедимой. Относительно класса лошадей создадим порожденный класс comhorse (для конкурентоспособной лошади). Перегрузим функцию horse\_tick() в этом классе таким образом, чтобы каждая лошадь могла проверять, является ли она ведущей и нет ли поблизости соперников (скажем, ближе, чем на 0,1 форлонг (1/80 часть мили или 20,1 м.)). Если есть, то ей следует немного ускориться. Может быть, не настолько, чтобы побеждать на всех скачках, но в достаточной мере для того, чтобы оставаться конкурентоспособной.

Как каждая лошадь узнает, где находятся остальные? Моделирующий ее объект должен иметь доступ к области памяти, в которой хранятся данные о соперниках. В программе это hArray. Будьте внимательны: вы создаете класс для передовой лошади, он должен быть наследником класса всех лошадей. Поэтому классу comhorse потребуется перегрузить hArray.

Вам может потребоваться создать еще один производный класс, comtrack, для отслеживания позиции лошади.

Можно непрерывно проверять, лидирует ли ваша лошадь, и если она впереди всех, но лишь ненамного, следует ее немного ускорить.

**Код:**

#include "msoftcon.h" //for console graphics

#include <iostream> //for I/O

#include <cstdlib> //for random()

#include <ctime> //for time()

using namespace std;

const int CPF = 5; //columns per furlong

const int maxHorses = 7; //maximum number of horses

class track; //for forward references

class horse

{

protected:

const track\* ptrTrack; //pointer to track

const int horse\_number; //this horse's number

float finish\_time; //this horse's finish time

float distance\_run; //distance run so far

public: //create the horse

horse(const int n, const track\* ptrT) :

horse\_number(n), ptrTrack(ptrT),

distance\_run(0.0) //haven't moved yet

{ }

~horse() //destroy the horse

{ /\*empty\*/

} //display the horse

void display\_horse(const float elapsed\_time);

}; //end class horse

class track

{

protected:

horse\* hArray[maxHorses]; //array of ptrs-to-horses

int total\_horses; //total number of horses

int horse\_count; //horses created so far

const float track\_length; //track length in furlongs

float elapsed\_time; //time since start of race

public:

track(float lenT, int nH); //2-arg constructor

~track(); //destructor

void display\_track(); //display track

void run(); //run the race

float get\_track\_len() const; //return total track length

}; //end class track

void horse::display\_horse(float elapsed\_time) //for each horse

{ //display horse & number

set\_cursor\_pos(1 + int(distance\_run \* CPF),

2 + horse\_number \* 2);

//horse 0 is blue

set\_color(static\_cast<color>(cBLUE + horse\_number));

//draw horse

char horse\_char = '0' + static\_cast<char>(horse\_number);

\_putch(' '); \_putch('\xDB'); \_putch(horse\_char); \_putch('\xDB');

//until finish,

if (distance\_run < ptrTrack->get\_track\_len() + 1.0 / CPF)

{

if (rand() % 3) //skip about 1 of 3 ticks

distance\_run += 0.2F; //advance 0.2 furlongs

finish\_time = elapsed\_time; //update finish time

}

else

{ //display finish time

int mins = int(finish\_time) / 60;

int secs = int(finish\_time) - mins \* 60;

cout << " Time=" << mins << ":" << secs;

}

} //end display\_horse()

track::track(float lenT, int nH) : //track constructor

track\_length(lenT), total\_horses(nH),

horse\_count(0), elapsed\_time(0.0)

{

init\_graphics(); //start graphics

total\_horses = //not more than 7 horses

(total\_horses > maxHorses) ? maxHorses : total\_horses;

for (int j = 0; j < total\_horses; j++) //make each horse

hArray[j] = new horse(horse\_count++, this);

time\_t aTime; //initialize random numbers

srand(static\_cast<unsigned>(time(&aTime)));

display\_track();

} //end track constructor

track::~track() //track destructor

{

for (int j = 0; j < total\_horses; j++) //delete each horse

delete hArray[j];

}

void track::display\_track()

{

clear\_screen(); //clear screen

//display track

for (int f = 0; f <= track\_length; f++) //for each furlong

for (int r = 1; r <= total\_horses \* 2 + 1; r++) //and screen row

{

set\_cursor\_pos(f\*CPF + 5, r);

if (f == 0 || f == track\_length)

cout << '\xDE'; //draw start or finish line

else

cout << '\xB3'; //draw furlong marker

}

} //end display\_track()

void track::run()

{

while (!\_kbhit())

{

elapsed\_time += 1.75; //update time

//update each horse

for (int j = 0; j < total\_horses; j++)

hArray[j]->display\_horse(elapsed\_time);

wait(500);

}

\_getch(); //eat the keystroke

cout << endl;

}

float track::get\_track\_len() const

{

return track\_length;

}

class comhorse : public horse

{

public:

comhorse(const int n, const track\* ptrT) : horse(n, ptrT) {}

float get\_distance()

{

return distance\_run;

}

void horse\_tick()

{

set\_cursor\_pos(1 + int(distance\_run\*CPF), 2 + horse\_number \* 2);

set\_color(static\_cast<color>(cBLUE + horse\_number));

char horse\_char = '0' + static\_cast<char>(horse\_number);

cout << ' ' << '\xDB' << horse\_char << '\xDB';

distance\_run += 0.2F;

}

};

class comtrack : public track

{

private:

comhorse \*hArray[maxHorses];

public:

comtrack(float lenT, int nH) : track(lenT, nH)

{

horse\_count = 0;

for (int j = 0; j < total\_horses; j++)

hArray[j] = new comhorse(horse\_count++, this);

}

void run()

{

while (!\_kbhit())

{

int horse\_leader = 0;

float delta = 0.0;

elapsed\_time += 1.75;

for (int j = 0; j < total\_horses; j++)

hArray[j]->display\_horse(elapsed\_time);

for (int j = 0; j < total\_horses; j++)

if (hArray[horse\_leader]->get\_distance() < hArray[j]->get\_distance())

horse\_leader = j;

for (int j = 0; j < total\_horses; j++)

if (hArray[horse\_leader]->get\_distance() - hArray[j]->get\_distance() > delta)

delta = hArray[horse\_leader]->get\_distance() - hArray[j]->get\_distance();

if ((hArray[horse\_leader]->get\_distance() > (track\_length + 1.0 / CPF) / 2) &&

(hArray[horse\_leader]->get\_distance() < (track\_length + 1.0 / CPF)) &&

(delta <= 2.2f))

{

hArray[horse\_leader]->horse\_tick();

}

wait(100);

}

\_getch();

cout << endl;

}

};

int main()

{

float length;

int total;

//get data from user

cout << "\nEnter track length (furlongs; 1 to 12): ";

cin >> length;

cout << "\nEnter number of horses (1 to 7): ";

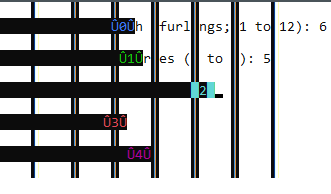
cin >> total;

comtrack theTrack(length, total); //create the track

theTrack.run(); //run the race

return 0;

} //end main()

**Вывод:**  




**Задание 10**

Упражнение 4 в главе про перегрузку операций включало в себя добавление к классу Linklist перегружаемого деструктора. Допустим, мы заполняем объект этого класса данными, а затем присваиваем один класс целиком другому, используя стандартный оператор присваивания:

list2 = listl;

Допустим, что впоследствии мы удалим объект класса Listl. Можем ли мы все еще использовать list2 для доступа к введенным данным? Увы, нет, так как при удалении listl все его ссылки были удалены. Единственное, что было известно объекту linklist про удаленный объект, это указатель на него. Но его удалили, указатель в List2 стал недееспособным, и все попытки получить доступ к данным приведут к получению мусора вместо данных, а в худшем случае — к зависанию программы.

Один из способов избежать этих проблем — перегрузить оператор присваивания, чтобы он вместе с объектом копировал бы все его ссылки. Но тогда придется пройти по всей цепочке, поочередно копируя все ссылки. Как отмечалось ранее, следует также перегружать конструктор копирования. Чтобы была возможность удалять объекты linklist в main(), можно создавать их с помощью указателя и new. В таком случае проще будет проверять работу новых операций. Не переживайте, если обнаружите, что в процессе копирования порядок следования данных изменился.

**Код:**

#include <iostream>

using namespace std;

struct link //one element of list

{

int data; //data item

link\* next; //pointer to next link

};

class linklist //a list of links

{

private:

link\* first; //pointer to first link

public:

linklist() //no-argument constructor

{

first = NULL;

}

linklist(const linklist &list) //no-argument constructor

{

if (list.first == NULL) return;

first = new link();

first->data = list.first->data;

link \*cur1 = first;

link \*cur2 = list.first;

while (cur2->next != NULL)

{

cur2 = cur2->next;

cur1->next = new link();

cur1 = cur1->next;

cur1->data = cur2->data;

}

}

linklist& operator=(const linklist &list)

{

\*this = linklist(list);

return \*this;

}

~linklist(); //destructor

void additem(int d); //add data item (one link)

void display(); //display all links

};

void linklist::additem(int d) //add data item

{

link\* newlink = new link; //make a new link

newlink->data = d; //give it data

newlink->next = first; //it points to next link

first = newlink; //now first points to this

}

void linklist::display() //display all links

{

link\* current = first; //set ptr to first link

while (current != NULL) //quit on last link

{

cout << " " << current->data; //print data

current = current->next; //move to next link

}

}

linklist::~linklist() //destructor

{

link\* current = first; //set ptr to first link

cout << "\nУдаление списка!\n";

while (current != NULL) //quit on last link

{

link\* temp = current; //save ptr to this link

current = current->next; //get ptr to next link

cout << temp->data << " ";

delete temp; //delete this link

}

}

int main()

{

setlocale(LC\_ALL, "");

linklist \*li = new linklist(); //make linked list

cout << "\nОбъявление нового списка:\n";

li->additem(25); //add four items to list

li->additem(36);

li->additem(49);

li->additem(64);

li->display(); //display entire list

linklist copy = \*li;

delete li;

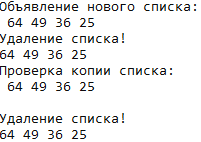
cout << "\nПроверка копии списка:\n";

copy.display();

cout << endl;

return 0;

}

**Вывод:**  


**Задание 11**

Выполните изменения в предложенной программе соответствии с упражнением 7. То есть заставьте программу анализировать выражения, содержащие числа в формате с плавающей запятой. Совместите классы, предложенные в упражнении 7, с алгоритмами из программы, приведённой ниже. Вам придется работать с указателями на символы вместо работы с самими символами

Реализовано в 8м задании