**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**



**МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**Факультет 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 функции.

**Выполнил(а): студент(ка) группы 211-7210**

**Салов Д.К.**

(Фамилия И.О.)

**Дата, подпись** 14.06.22

(Дата) (Подпись)

**Проверил: \_\_*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_***

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

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

(Дата) (Подпись)

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

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Москва**

**2021**

**Цель:**  Получить практические навыки в создании и использовании

виртуальных функций.**Задания:**

**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.

#include <iostream>

#include <iomanip>

#include <string>

#include <sstream>

#include <string\_view>

using namespace std;

class publication {

private:

string title;

float price;

public:

void putdata() {

cout << "Enter a name: ";

cin >> title;

cout << "Enter a price: ";

cin >> price;

}

void getdata() const {

cout << "\nName " << title;

cout << "\nCost: " << price << "\n";

}

};

class book : public publication {

private:

int pages;

public:

void putdata() {

publication::putdata();

cout << "Enter number of pages: ";

cin >> pages;

}

void getdata() const {

publication::getdata();

cout << "Number of pages: " << pages << "\n";

}

};

class tape : public publication {

private:

float playing\_time;

public:

void putdata() {

publication::putdata();

cout << "Enter playing time: ";

cin >> playing\_time;

}

void getdata() const {

publication::getdata();

cout << "Time in secs: " << playing\_time << "\n";

}

};

int main()

{

int n = 0;

publication\* ptr[100];

char ch;

do {

cout << "Choose book or tape (b/t): ";

cin >> ch;

if (ch == 'b')

ptr[n] = new book;

if (ch == 't')

ptr[n] = new tape;

ptr[n++]->putdata();

cout << "(Continue? n/y): ";

cin >> ch;

} while (ch != 'n');

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

ptr[i]->getdata();

}

}

**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.

#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

   };

//--------------------------------------------------------------

                                  //multiply d1 by d2

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;

   }

#include <iostream>

#include <iomanip>

#include <string>

#include <sstream>

#include <string\_view>

using namespace std;

class Distance {

private:

int feet;

float inches;

public:

Distance() : feet(0), inches(0.0) {}

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

Distance(float fltfeet)

{

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

inches = 12 \* (fltfeet - feet);

}

void showdist() const

{

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

}

friend float operator\* (Distance, Distance);

};

float operator\* (Distance d1, Distance d2) {

return (d1.feet + d1.inches / 12) \* (d2.feet + d2.inches / 12);

}

int main()

{

Distance dist1 = 2.5;

Distance dist2 = 1.25;

float dist;

cout << "First distance = ";

dist1.showdist();

cout << "\nSecond distance = ";

dist2.showdist();

dist = dist1 \* dist2;

cout << "\n\nFirst distance \* Second distance = " << dist << " kv";

dist = dist2 \* dist1;

cout << "\nSecond distance \* First distance = " << dist << " kv";

dist = 7.5 \* dist1;

cout << "\n\n7.5 \* First distance = " << dist << " kv";

dist = dist1 \* 7.5;

cout << "\nFirst distance \* 7.5 = " << dist << " kv\n";

}

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.

// clarray.cpp

// creates array class #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()                  //destructor

         { delete[] ptr; }       int& operator [] (int j)  //overloaded subscript operator          { return \*(ptr+j); }

   };

//////////////////////////////////////////////////////////////// int main()

   {

   const int ASIZE = 10;        //size of array

   Array arr(ASIZE);            //make an array

   for(int j=0; j<ASIZE; j++)   //fill it with squares       arr[j] = j\*j;

   for(j=0; j<ASIZE; j++)       //display its contents       cout << arr[j] << ' ';    cout << endl;    return 0;

   }

#include <iostream>

#include <iomanip>

#include <string>

#include <sstream>

#include <string\_view>

using namespace std;

class Array

{

private:

int\* ptr;

int size;

public:

Array() : ptr(0), size(0) {}

Array(int s)

{

size = s;

ptr = new int[s];

}

Array(Array& arr)

{

size = arr.size;

ptr = new int[size];

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

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

}

~Array() {

delete[] ptr;

}

int& operator[] (int j)

{

return \*(ptr + j);

}

Array& operator= (Array& arr)

{

if (this == &arr) return \*this;

delete[] ptr;

size = arr.size;

ptr = new int[size];

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

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

return \*this;

}

void display()

{

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

cout << \*(ptr + i) << ' ';

}

};

int main()

{

const int ASIZE = 10;

Array arr1(ASIZE), arr3(ASIZE);

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

arr1[j] = j \* j;

cout << "First array = ";

arr1.display();

cout << "\n";

Array arr2(arr1); //!!!

cout << "Second array = ";

arr2.display();

cout << "\n";

arr3 = arr1;

cout << "Third array = ";

arr3.display();

}

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.

#include <iostream>

#include <iomanip>

#include <string>

#include <sstream>

#include <string\_view>

using namespace std;

class publication

{

private:

string title;

float price;

public:

virtual void putdata()

{

cout << "Enter a name: ";

cin >> title;

cout << "Enter a price: ";

cin >> price;

}

virtual void getdata() const

{

cout << "\nName " << title;

cout << "\nCost: " << price << "\n";

}

virtual bool isOversize() const = 0;

};

class book : public publication

{

private:

int pages;

public:

void putdata()

{

publication::putdata();

cout << "Enter count of pages: ";

cin >> pages;

}

void getdata() const

{

publication::getdata();

cout << "Count of pages: " << pages << "\n";

}

bool isOversize() const

{

if (pages > 800)

return true;

else

return false;

}

};

class tape5 : public publication

{

private:

float playing\_time;

public:

void putdata()

{

publication::putdata();

cout << "Enter playing time: ";

cin >> playing\_time;

}

void getdata() const

{

publication::getdata();

cout << "Time in secs: " << playing\_time << "\n";

}

bool isOversize() const

{

if (playing\_time > 90)

return true;

else

return false;

}

};

int main()

{

int n = 0;

publication\* ptr[100];

char ch;

do

{

cout << "Choose book or tape (b/t): ";

cin >> ch;

if (ch == 'b')

ptr[n] = new book;

if (ch == 't')

ptr[n] = new tape5;

ptr[n++]->putdata();

cout << "(Continue? n/y): ";

cin >> ch;

} while (ch != 'n');

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

{

ptr[i]->getdata();

if (ptr[i]->isOversize())

cout << "Oversize";

}

}

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.

#include <iostream>

#include <iomanip>

#include <string>

#include <sstream>

#include <string\_view>

using namespace std;

class bMoney

{

private:

string strMon;

long double capital;

public:

bMoney() : capital(0)

{}

bMoney(long double num)

{

capital = num;

}

operator long double() const

{

long double num = capital;

return num;

}

void mstold()

{

cout << "Enter the amount in decimal point. End with a dollar sign" << endl;

getline(cin, strMon, '$');

int wlen = strMon.length();

int n = 0;

string num;

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

if (strMon[j] != ',' && strMon[j] != '$')

num.push\_back(strMon[j]); ;

capital = stold(num);

}

bMoney operator + (bMoney mon2)

{

return capital + mon2.capital;

}

bMoney operator - (bMoney mon2)

{

return capital - mon2.capital;

}

bMoney operator \* (bMoney mon2)

{

return capital \* mon2;

}

bMoney operator / (bMoney mon2)

{

return capital / mon2.capital;

}

bMoney operator / (long double mon2)

{

return capital / mon2;

}

friend bMoney operator\* (long double, bMoney);

friend long double operator/ (long double, bMoney);

void output()const

{

cout << setiosflags(ios::fixed)

<< setiosflags(ios::showpoint)

<< setprecision(2)

<< "\n" << capital << '$' << endl;

}

};

bMoney operator\* (long double value, bMoney mvalue)

{

return bMoney(value \* mvalue.capital);

}

long double operator/ (long double value, bMoney mvalue)

{

return (value / mvalue.capital);

}

int main()

{

bMoney val1, val2, result;

long double amount = 0;

do

{

val1.mstold();

val2.mstold();

result = val1 + val2;

result.output();

result = val1 - val2;

result.output();

amount = val2;

result = val1 \* val2;

result.output();

result = val1 / val2;

result.output();

result = val1 / amount;

result.output();

cout << "Continue'? (y/n)\n";

cin.get();

} while (cin.get() != 'n');

}

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:

mo2 = round(mol):

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.

#include <iostream>

#include <iomanip>

#include <string>

#include <sstream>

#include <string\_view>

using namespace std;

class bMoney

{

private:

string strMon;

long double capital;

public:

bMoney() : capital(0) {}

bMoney(long double num)

{

capital = num;

}

operator long double() const

{

long double num = capital;

return num;

}

void mstold()

{

cout << "Enter the amount in decimal point. End with a dollar sign" << endl;

getline(cin, strMon, '$');

int wlen = strMon.length();

int n = 0;

string num;

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

if (strMon[j] != ',' && strMon[j] != '$')

num.push\_back(strMon[j]); ;

capital = stold(num);

}

bMoney operator + (bMoney mon2)

{

return capital + mon2.capital;

}

bMoney operator - (bMoney mon2)

{

return capital - mon2.capital;

}

bMoney operator \* (bMoney mon2)

{

return capital \* mon2;

}

bMoney operator / (bMoney mon2)

{

return capital / mon2.capital;

}

bMoney operator / (long double mon2)

{

return capital / mon2;

}

friend bMoney operator\* (long double, bMoney);

friend long double operator/ (long double, bMoney);

friend bMoney round(bMoney);

void output()

{

cout << setiosflags(ios::fixed)

<< setiosflags(ios::showpoint)

<< setprecision(2)

<< "\n" << capital << '$' << endl;

}

};

int main()

{

bMoney val1, val2;

val1.mstold();

val1.output();

val2 = round(val1);

val2.output();

cout << endl;

}

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.

*Пример сценария для реализации*

class token {    public:

                        virtual float getnumber() = 0;                        virtual char getoperator = 0;             };

            class operator : public token {                        private:

char oper;

                        public:

                                    operator(char);                                    char getoperator();              float getnumber();

};

class number : public token {              private:                       float fnum;              public:

                        number(float);                         float getnumber();                 char getoperator();

};

token \* atoken[100];

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.

#include <iostream>

using namespace std;

class Token {

public:

virtual float getNumber() const = 0;

virtual char getOperator() const = 0;

};

class Operator : public Token {

private:

char oper;

public:

Operator(char op) : oper(op) {}

char getOperator() const {

return oper;

}

float getNumber() const {

return 0.0f;

}

};

class Number : public Token {

private:

float fnum;

public:

Number(float n) : fnum(n) {}

float getNumber() const {

return fnum;

}

char getOperator() const {

return '0';

}

};

class Stack {

private:

Token\* atoken[100];

int head;

public:

Stack() {

head = 0;

}

void push(Token\* var) {

atoken[++head] = var;

}

Token\* pop() {

return atoken[head--];

}

int gettop() const {

return head;

}

bool isNumber() const {

Number\* pNum;

if (pNum = dynamic\_cast<Number\*>(atoken[head]))

return true;

else

return false;

}

};

int main() {

Stack stack;

Number value1(4.7), value2(7.378), value3(27.29), value4(8.0), value5(6.11);

Operator plus('+'), mul('\*'), minus('-'), div('/');

stack.push(&value4); stack.push(&div); stack.push(&value5);

stack.push(&mul); stack.push(&value3); stack.push(&plus);

stack.push(&value1); stack.push(&plus); stack.push(&value2);

while (stack.gettop() > 0) {

if (stack.isNumber())

cout << stack.pop()->getNumber();

else

cout << stack.pop()->getOperator();

if (stack.gettop() > 0)

cout << ' ';

}

return 0;

}

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.

// horse.cpp

// models a horse race

#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

   {

   private:

      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

   {

   private:

      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; }

///////////////////////////////////////////////////////////////// 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;

   track theTrack(length, total);   //create the track    theTrack.run();                  //run the race    return 0;    }  //end main()

ТУТ ЧТО-ТО НЕ ТАК!

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.

#include <iostream>

using namespace std;

struct link {

int data;

link\* next;

};

class linklist {

private:

link\* first;

public:

linklist() {

first = NULL;

}

linklist(linklist&);

~linklist();

void additem(int d);

void display();

linklist& operator= (linklist&);

};

linklist::linklist(linklist& li) {

first = NULL;

link\* current = li.first;

while (current) {

additem(current->data);

current = current->next;

}

}

void linklist::additem(int d) {

link\* newlink = new link;

newlink->data = d;

newlink->next = first;

first = newlink;

}

void linklist::display() {

link\* current = first;

while (current) {

cout << current->data << ' ';

current = current->next;

}

cout << endl;

}

linklist::~linklist() {

link\* current = first;

while (current) {

first = current->next;

delete current;

cout << "Elemet has deleted." << endl;

current = first;

}

}

linklist& linklist::operator= (linklist& li) {

if (this == &li)

return \*this;

link\* current = first;

while (current) {

first = current->next;

delete current;

current = first;

}

current = li.first;

while (current) {

additem(current->data);

current = current->next;

}

return \*this;

}

int main() {

linklist\* list1 = new linklist;

list1->additem(25);

list1->additem(36);

list1->additem(49);

list1->additem(64);

cout << "list1: ";

list1->display();

linklist list2(\*list1);

linklist list3;

list3 = \*list1;

delete list1;

cout << "list2: ";

list2.display();

cout << "list3: ";

list3.display();

return 0;

}

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. Это потребует выражений, подобных следующим:

Number\* ptrN = new Number Cans): s.push(ptrN);

И

Operator\* ptrO = new Operator(ch); s.push(ptrO);

// parse.cpp

// evaluates arithmetic expressions composed of 1-digit numbers

#include <iostream>

#include <cstring>                   //for strlen(), etc using namespace std; const int LEN = 80;    //length of expressions, in characters const int MAX = 40;    //size of stack

//////////////////////////////////////////////////////////////// class Stack

   {

   private:

      char st[MAX];                  //stack: array of chars       int top;                       //number of top of stack public:

      Stack()                        //constructor

         { top = 0; }       void push(char var)            //put char on stack

         { st[++top] = var; }       char pop()                     //take char off stack

         { return st[top--]; }       int gettop()                   //get top of stack

         { return top; }

   };

//////////////////////////////////////////////////////////////// class express                        //expression class

   {

   private:

      Stack s;                       //stack for analysis       char\* pStr;                    //pointer to input string       int len;                       //length of input string    public:       express(char\* ptr)             //constructor

         {

         pStr = ptr;                 //set pointer to string          len = strlen(pStr);         //set length

         }

      void parse();                  //parse the input string       int solve();                   //evaluate the stack

   };

//-------------------------------------------------------------- void express::parse()                //add items to stack

   {

   char ch;                          //char from input string    char lastval;                     //last value    char lastop;                      //last operator

   for(int j=0; j<len; j++)          //for each input character

      {

      ch = pStr[j];                  //get next character

      if(ch>='0' && ch<='9')         //if it's a digit,

         s.push(ch-'0');             //save numerical value

                                     //if it's operator

      else if(ch=='+' || ch=='-' || ch=='\*' || ch=='/')

         {

         if(s.gettop()==1)           //if it's first operator             s.push(ch);              //put on stack          else                        //not first operator

            {

            lastval = s.pop();       //get previous digit

            lastop = s.pop();        //get previous operator             //if this is \* or / AND last operator was + or -             if( (ch=='\*' || ch=='/') &&

                (lastop=='+' || lastop=='-') )

               {

               s.push(lastop);       //restore last two pops

               s.push(lastval);

               }             else                     //in all other cases

               {

               switch(lastop)        //do last operation                   {                  //push result on stack                   case '+': s.push(s.pop() + lastval); break;                   case '-': s.push(s.pop() - lastval); break;                   case '\*': s.push(s.pop() \* lastval); break;                   case '/': s.push(s.pop() / lastval); break;                   default:  cout << "\nUnknown oper"; exit(1);

                  }  //end switch

               }  //end else, in all other cases

            s.push(ch);              //put current op on stack

            }  //end else, not first operator          }  //end else if, it's an operator       else                           //not a known character

         { cout << "\nUnknown input character"; exit(1); }

      }  //end for

   }  //end parse()

//-------------------------------------------------------------- int express::solve()                 //remove items from stack

   {

   char lastval;                     //previous value

   while(s.gettop() > 1)

      {

      lastval = s.pop();             //get previous value       switch( s.pop() )              //get previous operator          {                           //do operation, push answer          case '+': s.push(s.pop() + lastval); break;          case '-': s.push(s.pop() - lastval); break;          case '\*': s.push(s.pop() \* lastval); break;          case '/': s.push(s.pop() / lastval); break;          default:  cout << "\nUnknown operator"; exit(1);

         }  //end switch       }  //end while

   return int( s.pop() );            //last item on stack is ans

   }  //end solve()

//////////////////////////////////////////////////////////////// int main()

{    char ans;                         //'y' or 'n'

   char string[LEN];                 //input string from user

   cout << "\nEnter an arithmetic expression"

           "\nof the form 2+3\*4/3-2."

           "\nNo number may have more than one digit."            "\nDon't use any spaces or parentheses.";    do {       cout << "\nEnter expresssion: ";       cin >> string;                        //input from user       express\* eptr = new express(string);  //make expression

      eptr->parse();                        //parse it       cout << "\nThe numerical value is: "             << eptr->solve();                //solve it       delete eptr;                          //delete expression       cout << "\nDo another (Enter y or n)? ";       cin >> ans;       } while(ans == 'y');    return 0;

   }