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 № 6

Дисциплина: ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ

Тема: Массивы и строки

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**

Напишите функцию reversit(), которая переворачивает строку (массив типа char). Используйте цикл for, который меняет местами первый и последний символы, затем следующие и т. д. до предпоследнего. Строка должна передаваться в функцию reversit() как аргумент.

Напишите программу для выполнения функции reversit(). Программа должна принимать строку от пользователя, вызывать функцию reversit(), а затем выводить полученный результат. Используйте метод ввода, который позволяет использовать внутренние пробелы. Протестируйте программу на примере фразы «Аргентина манит негра».

**Код:**

#include <iostream>

#include <cstring>//для strlen()

using namespace std;

int main()

{

setlocale(LC\_ALL, "");

void reversit(char[]);

const int MAX = 80;

char str[MAX];

cout << "\nВведите строку: ";

cin.get(str, MAX);

reversit(str);

cout << "Перевернутая строка: ";

cout << str << endl;

return 0;

}

void reversit(char s[])

{

int len = strlen(s);

for (int j = 0; j < len / 2; j++)

{

char temp = s[j];

s[j] = s[len - j - 1];

s[len - j - 1] = temp;

}

}

**Вывод:**  


**Задание 2**

Создайте класс employee, который содержит имя (объект класса string) и номер (тина long) служащего. Включите в него метод getdata(), предназначенный для получения данных от пользователя и помещения их в объект, и метод putdata(), для вывода данных. Предполагаем, что имя может иметь внутренние пробелы.

Напишите функцию main(), использующую этот класс. Вам нужно будет создать массив типа employee, а затем предложить пользователю ввести данные до 100 служащих. Наконец, вам нужно будет вывести данные всех служащих.

**Код:**

#include <iostream>

#include <string>

using namespace std;

class employee

{

private:

string name;

long number;

public:

void getdata()

{

cout << "\nВведите имя:"; cin >> name;

cout << "Введите номер:"; cin >> number;

}

void putdata()

{

cout << "\n Имя:" << name;

cout << "\n Номер:" << number;

}

};

int main()

{

setlocale(LC\_ALL, "");

employee emparr[100];

int n = 0;

char ch;

do {

cout << "\nВведите данные о работнике с номером " << n + 1;

emparr[n++].getdata();

cout << "Продолжить (y/n)?"; cin >> ch;

} while (ch != 'n');

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

{

cout << "\nНомер работника " << j + 1;

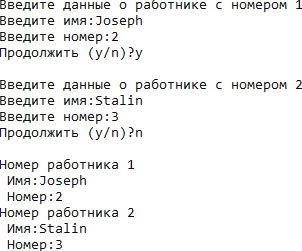
emparr[j].putdata();

}

cout << endl;

return 0;

}

**Вывод:**  


**Задание 3**

Напишите программу, вычисляющую среднее значение 100 интервалов, введенных пользователем. Создайте массив объектов класса Distance, как это было сделано в примере ENGLARAY этой главы. Для вычисления среднего значения вы можете позаимствовать метод add\_dist() из примера EN6LC0N главы 5. Вам также понадобится метод, который выделяет целую часть из значения Distance. Вот одна из возможностей:

**Код:**

#include <iostream>

using namespace std;

class Distance //класс английских расстояний

{

private:

int feet;

float inches;

public:

Distance() //конструктор (без аргументов)

{

feet = 0; inches = 0;

}

Distance(int ft, float in)//конструктор (2 аргумента)

{

feet = ft; inches = in;

}

void getdist() //получить расстояние от пользователя

{

cout << "\nВведите футы:"; cin >> feet;

cout << "Введите дюймы:"; cin >> inches;

}

void showdist() //вывод расстояния

{

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

}

void add\_dist(Distance, Distance);//объявления

void div\_dist(Distance, int);

};

//сложение расстояний d2 и d3

void Distance::add\_dist(Distance d2, Distance d3)

{

inches = d2.inches + d3.inches; //сложить дюймы

feet = 0; //(для возможного переноса)

if (inches >= 12.0) //если сумма превышает 12.0,

{//уменьшить число дюймов

inches -= 12.0; //на 12.0 и

feet++; //увеличить число футов

} //на 1

feet += d2.feet + d3.feet; //сложить футы

}

//деление объекта Distance на целое число

void Distance::div\_dist(Distance d2, int divisor)

{

float fltfeet = d2.feet + d2.inches / 12.0;//преобразовать в формат float

fltfeet /= divisor; //выполнить деление

feet = int(fltfeet); //получить футовую часть

inches = (fltfeet - feet)\*12.0; //получить дюймовую часть

}

int main()

{

setlocale(LC\_ALL, "");

Distance distarr[100]; //массив из 100 объектов типа Distance

Distance total(0, 0.0), average;// прочие расстояния

int count = 0; //считает количество введенных значений

char ch; //признак ответа пользователя

do {

cout << "\nВведите расстояние "; //получить значения

distarr[count++].getdist(); //от оператора,занести

cout << "\nПродолжить (y/n)?"; //их в массив

cin >> ch;

} while (ch != 'n');

for (int j = 0; j < count; j++) //сложить все расстояния

total.add\_dist(total, distarr[j]); //в total

average.div\_dist(total, count); //разделить на число

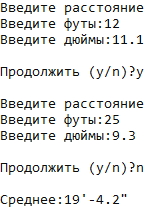
cout << "\nСреднее:"; //вывести среднее значение

average.showdist();

cout << endl;

return 0;

}

**Вывод:**  


**Задание 4**

Начните с программы, которая позволяет пользователю вводить целые числа, а затем сохранять их в массиве типа int. Напишите функцию maxint(), которая, обрабатывая элементы массива один за другим, находит наибольший. Функция должна принимать в качестве аргумента адрес массива и количество элементов в нем, а возвращать индекс наибольшего элемента. Программа должна вызвать эту функцию, а затем вывести наибольший элемент и его индекс.

**Код:**

#include <iostream>

using namespace std;

int intmax(int arr[], int size)

{

int max = 0;

for (int i = 1; i < size; i++)

if (arr[i] > arr[max]) max = i;

return max;

}

int main()

{

setlocale(LC\_ALL, "");

int arr[100];

int count = 0;

char ch;

do {

cout << "Введите целое число ";

cin >> arr[count++];

cout << "Продолжить (y/n)?";

cin >> ch;

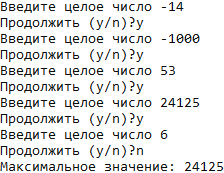
} while (ch != 'n');

int max = intmax(arr, count);

cout << "Максимальное значение: " << arr[max];

return 0;

}

**Вывод:**  


**Задание 5**

Начните с класса fraction из упражнений 11 и 12 лабораторной работы 5. Напишите функцию main(), которая получает случайные дробные числа от пользователя, сохраняет их в массиве типа fraction, вычисляет среднее значение и выводит результат.

**Код:**

#include <iostream>

#include "fraction.h"

using namespace std;

int main()

{

setlocale(LC\_ALL, "");

fraction arr[100];

fraction total(0, 1);

int count = 0;

char ch;

string str;

do {

cout << "Введите дробь ";

cin >> str;

arr[count++].parse(str);

cout << "Продолжить (y/n)?";

cin >> ch;

} while (ch != 'n');

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

total.sum(arr[j]);

total.mul(fraction(1, count));

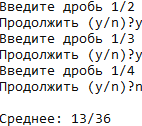
cout << "\nСреднее: ";

total.display();

cout << endl;

return 0;

}

**Вывод:**  


**Задание 6**

В игре бридж каждому из игроков раздают 13 карт, таким образом колода расходуется полностью. Модифицируйте программу CARDARAY этой главы так, чтобы после перемешивания колоды она делилась на четыре части по 13 карт каждая. Каждая из четырех групп карт затем должна быть выведена.

**Код:**

#include <iostream>

#include <cstdlib>

#include <ctime>

using namespace std;

enum Suit { clubs, diamonds, hearts, spades };

const int jack = 11;

const int queen = 12;

const int king = 13;

const int ace = 14;

class card

{

private:

int number;

Suit suit;

public:card()

{ }

void set(int n, Suit s)

{

suit = s; number = n;

}

void display();

};

void card::display()

{

if (number >= 2 && number <= 10)

cout << number;

else

{

switch (number)

{

case jack: cout << 'J'; break;

case queen: cout << 'Q'; break;

case king: cout << 'K'; break;

case ace: cout << 'A'; break;

}

}

switch (suit)

{

case clubs: cout << static\_cast<char>(5); break;

case diamonds: cout << static\_cast<char>(4); break;

case hearts: cout << static\_cast<char>(3); break;

case spades: cout << static\_cast<char>(6); break;

}

}

int main()

{

setlocale(LC\_ALL, "");

card deck[52];

int j;

cout << endl;

for (j = 0; j < 52; j++)

{

int num = (j % 13) + 2;

Suit su = Suit(j / 13);

deck[j].set(num, su);

}

cout << "Исходная колода:\n";

for (j = 0; j < 52; j++)

{

deck[j].display();

cout << " ";

if (!((j + 1) % 13))

cout << endl;

}

srand(time(NULL));

for (j = 0; j < 52; j++)

{

int k = rand() % 52;

card temp = deck[j];

deck[j] = deck[k];

deck[k] = temp;

}

card newdesk[4][13];

for (int i = 0, j = 0; j < 52; j++)

{

int buf = (j + 1) % 13;

newdesk[i][buf] = deck[j];

if (!buf) i++;

}

cout << "\nПеремешанная колода:\n";

for (int i = 0; i < 4; i++)

{

cout << "Колода №" << (i + 1) << ":\n";

for (card c : newdesk[i])

{

c.display();

cout << ' ';

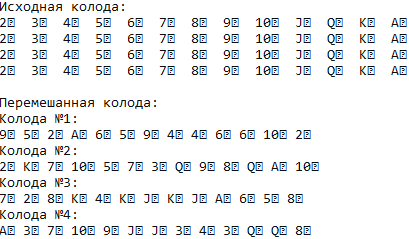
}

cout << endl;

}

return 0;

}

**Вывод:**  


**Задание 7**

Одним из недостатков C++ является отсутствие для бизнес-программ встроенного типа для денежных значений, такого, как $173 698 001,32. Такой денежный тип должен иметь возможность для хранения числа с фиксированной десятичной точкой точностью около 17 знаков, которого было бы достаточно для хранения национального долга в долларах и центах. К счастью, встроенный тип C++ long double имеет точность 19 цифр, поэтому мы можем использовать его как базисный для класса money, даже используя плавающую точку. Однако нам нужно будет добавить возможность ввода и вывода денежных значений с предшествующим им знаком доллара и разделенными запятыми группы по три числа: так проще читать большие числа. Первым делом при разработке такого класса напишем метод mstold(), который принимает денежную строку, то есть строку, представляющую собой некоторое количество денег типа “$123.456.789.00”

в качестве аргумента и возвращает эквивалентное ее значению число типа long double.

Вам нужно будет обработать денежную строку как массив символов и, просматривая ее символ за символом, скопировать из нее только цифры (0-9) и десятичную точку в другую строку. Игнорируется все остальное, включая знак доллара и запятые. Затем вы можете использовать библиотечную функцию \_atold() для преобразования новой строки к числу типа long double. Напишите функцию main() для проверки метода mstold(), которая несколько раз получает денежную строку от пользователя и выводит соответствующее число типа long double.

**Код:**

#include <iostream>

#include <string>

using namespace std;

class money

{

private:

long double \_sum;

public:

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());

}

};

int main()

{

setlocale(LC\_ALL, "");

money m;

string str;

char ch;

do

{

cout << "Введите сумму:\n";

cin >> str;

cout << "Long double: " << m.mstold(str) << endl;

cout << "Продолжить(y/n)? ";

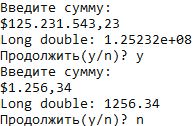
cin >> ch;

}

while (ch != 'n');

return 0;

}

**Вывод:**  


**Задание 8**

Другим недостатком C++ является отсутствие автоматической проверки индексов массива на соответствие их границам массива (это делает действия с массивами быстрыми, но менее надежными). Мы можем использовать класс для создания надежного массива, который проверяет индексы при любой попытке доступа к массиву. Напишите класс safearay, который использует массив типа int фиксированного размера (назовем его LIMIT) в качестве своей единственной переменной. В классе будет два метода. Первый, putel(), принимает индекс и значение типа int как аргументы и вставляет это значение в массив по заданному индексу. Второй, getel(), принимает индекс как аргумент и возвращает значение типа int, содержащееся в элементе с этим индексом.

safearay sal: // описываем массив

int temp = 12345: // описываем целое

sal.putel (7, temp ); // помещаем значение temp в массив

temp = sal.getel(7.4):// получаем значение из массива

Оба метода должны проверять индекс аргумента, чтобы быть уверенными, что он не меньше 0 и не больше, чем LIMIT—1. Вы можете использовать этот массив без опаски, что запись будет произведена в другие части памяти.

Использование методов для доступа к элементам массива не выглядит так наглядно, как использование операции [ ]. В главе 8 мы увидим, как перегрузить эту операцию, чтобы сделать работу нашего класса safearay похожей на работу встроенных массивов.

**Код:**

#include <iostream>

using namespace std;

const int Limit = 100;

class safearray

{

private:

int \_arr[Limit];

public:

void putel(int index, int el)

{

if (index >= 0 && index <= Limit - 1)

{

\_arr[index] = el;

cout << "Успешно присвоен элемент " << el << endl;

}

else

{

cout << "\nИндекс " << index << " вне границы массива";

}

}

int getel(int index)

{

if (index >= 0 && index <= Limit - 1)

{

return \_arr[index];

}

cout << "\nИндекс " << index << " вне границы массива";

exit(-1);

}

};

int main()

{

setlocale(LC\_ALL, "");

safearray sal;

int temp = 12345;

sal.putel(7, temp);

temp = sal.getel(7);

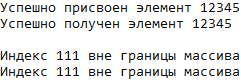
cout << "Успешно получен элемент " << temp << endl;

sal.putel(111, temp);

temp = sal.getel(111);

return 0;

}

**Вывод:**  


**Задание 9**

Очередь — это устройство для хранения данных, похожее на стек. Отличие в том, что в стеке последний сохраненный элемент будет первым извлеченным, тогда как в очереди первый сохраненный элемент будет первым извлеченным. То есть в стеке используется подход «последний вошел — первый вышел» (LJFO), а в очереди используется подход «первый вошел — первый вышел» (FIFO). Очередь похожа на простую очередь посетителей магазина: первый, кто встал в очередь, будет обслужен первым.

Перепишите программу STAKARAY из этой главы, включив в нее класс queue вместо класса stack. Кроме того, класс должен иметь два метода: один, называемый put(), для помещения элемента в очередь; и другой, называемый get(), для извлечения элемента из очереди. Эти методы эквивалентны методам push() и рор() класса stack.

Оба класса, stack и queue, используют массив для хранения данных. Однако вместо одного поля top типа int, как в классе stack, вам понадобятся два поля для очереди: одна, называемая head, указывающая на начало очереди; и вторая, tail, указывающая на конец очереди. Элементы помещаются в конец очереди (как посетители банка, становящиеся в очередь), а извлекаются из начала очереди. Конец очереди перемещается к началу по массиву по мере того, как элементы добавляются и извлекаются из очереди. Такие результаты добавляют сложности: если одна из двух переменных head или tail примут значение конца массива, то следует вернуться на начало. Таким образом, вам нужно выражение типа

if ( tail == MAX - 1 ) { tail = -1 };

для возврата переменной tail и похожее выражение для возврата переменной head. Массив, используемый в очереди, иногда называют круговым буфером, так как начало и конец очереди циркулируют по нему вместе с ее данными.

**Код:**

#include <iostream>

using namespace std;

const int MAX = 4;

class queue

{

private:

int \_arr[MAX];

public:

int head;

int tail;

queue()

{

head = 0;

tail = 0;

}

int get()

{

if (head == tail)

{

cout << "Очередь пуста\n";

exit(0);

}

int res = \_arr[tail];

tail = ++tail % MAX;

return res;

}

void put(int el)

{

if (tail == (head + 1) % MAX)

{

cout << "Очередь полна\n";

return;

}

\_arr[head] = el;

head = ++head % MAX;

cout << "Положен элемент " << el << endl;

}

};

int main()

{

setlocale(LC\_ALL, "");

queue q;

q.put(42);

q.put(442);

q.put(22);

q.put(12);

cout << "Взят элемент из очереди: " << q.get() << endl;

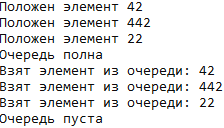
cout << "Взят элемент из очереди: " << q.get() << endl;

cout << "Взят элемент из очереди: " << q.get() << endl;

cout << "Взят элемент из очереди: " << q.get() << endl;

return 0;

}

**Вывод:**  


**Задание 10**

Матрица — это двумерный массив. Создайте класс matrix, который предоставляет те же меры безопасности, как и класс из задания 7, то есть осуществляет проверку индексов массива на вхождение их в границы массива. Полем класса matrix будет массив 10 на 10. Конструктор должен позволять программисту определить реальный размер массива (допустим, сделать его меньше, чем 10 на 10). Методам, предназначенным для доступа к членам матрицы, теперь нужны два индекса: по одному для каждой размерности массива. Вот фрагмент функции main(), которая работает с таким классом:

matrix ml(3,4); // Описываем матрицу

int temp = 12345; // Описываем целое

ml.putel(4, 4, temp); // Помещаем значение

temp = ml.getel(7,4); // Получаем значение из матрицы

**Код:**

#include <iostream>

using namespace std;

const int Limit = 10;

class matrix

{

private:

int \_arr[Limit][Limit];

int \_maxr;

int \_maxc;

public:

matrix() {}

matrix(int maxr, int maxc) : \_maxr(maxr), \_maxc(maxc)

{}

void putel(int ir, int ic, int el)

{

if ((ir >= 0 && ir <= \_maxr - 1) && (ic >= 0 && ic <= \_maxc - 1))

{

\_arr[ir][ic] = el;

cout << "Успешно присвоен элемент " << el << endl;

}

else

{

cout << "\nИндекс " << ir << "-" << ic << " вне границы массива";

}

}

int getel(int ir, int ic)

{

if ((ir >= 0 && ir <= \_maxr - 1) && (ic >= 0 && ic <= \_maxc - 1))

{

return \_arr[ir][ic];

}

cout << "\nИндекс " << ir << "-" << ic << " вне границы массива";

exit(-1);

}

};

int main()

{

setlocale(LC\_ALL, "");

matrix sal(5, 5);

int temp = 12345;

sal.putel(2, 4, temp);

temp = sal.getel(2, 4);

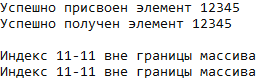
cout << "Успешно получен элемент " << temp << endl;

sal.putel(11, 11, temp);

temp = sal.getel(11, 11);

return 0;

}

**Вывод:**  


**Задание 11**

Вернемся к обсуждению денежных строк из упражнения 6. Напишите метод ldtoms() для преобразования числа типа long double в денежную строку, представляющую это число. Для начала вам нужно проверить, что значение long double не очень большое. Мы предполагаем, что вы не будете пытаться преобразовать число, больше чем 9 999 999 999 999 990.00. Затем преобразуем long double в строку (без знака доллара и запятых), хранящуюся в памяти, используя объект ostrstream, как рассматривалось ранее в этой главе. Получившаяся отформатированная строка может быть помещена в буфер, называющийся ustring.

Затем вам нужно будет создать другую строку, начинающуюся со знака доллара, далее копируем цифру за цифрой из строки ustring, начиная слева и вставляя запятые через каждые три цифры. Также вам нужно подавлять нули в начале строки. Например, вы должны вывести $3 124.95, а не $0 000 000 000 003 124.95. Не забудьте закончить строку нулевым символом '\0'.

Напишите функцию main() для тестирования этой функции путем многократного ввода пользователем чисел типа long double и вывода результата в виде денежной строки.

**Код:**

#include <iostream>

#include <string>

#include <sstream>

using namespace std;

class money

{

private:

long double \_sum;

public:

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;

}

};

int main()

{

setlocale(LC\_ALL, "");

money m;

long double s;

char ch;

do

{

cout << "Введите long double:\n";

cin >> s;

cout << "Денежная сумма: " << m.ldtoms(s) << endl;

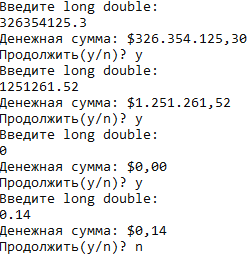
cout << "Продолжить(y/n)? ";

cin >> ch;

} while (ch != 'n');

return 0;

}

**Вывод:**  


**Задание 12**

Создайте класс bMoney. Он должен хранить денежные значения как long double. Используйте метод mstold() для преобразования денежной строки, введенной пользователем, в long double, и метод ldtoms() для преобразования числа типа long double в денежную строку для вывода (см. задания 6 и 10). Вы можете вызывать для ввода и вывода методы getmoney()и putmoney(). Напишите другой метод класса для сложения двух объектов типа ЬМопеу и назовите его madd(). Сложение этих объектов легко произвести: просто сложите переменную типа long double одного объекта с такой же переменной другого объекта. Напишите функцию main(), которая просит пользователя несколько раз ввести денежную строку, а затем выводит сумму значений этих строк.

**Код:**

#include <iostream>

#include <string>

#include <sstream>

using namespace std;

class money

{

private:

long double \_sum;

public:

money() {}

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 денежную сумму:\n";

cin >> str;

\_sum = mstold(str);

}

void putmoney()

{

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

}

void add(money m)

{

\_sum += m.\_sum;

}

};

int main()

{

setlocale(LC\_ALL, "");

money res(0), m;

long double s;

char ch;

do

{

m.getmoney();

res.add(m);

cout << "Продолжить(y/n)? ";

cin >> ch;

} while (ch != 'n');

res.putmoney();

return 0;

}

**Вывод:**  
