



МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Факультет 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 211-7210

Салов Д.К.

(Фамилия И.О.)

Дата, подпись 24.05.22

(Дата)

(Подпись)

Проверил: \_\_\_\_\_

(Фамилия И.О., степень, звание)

(Оценка)

Дата, подпись \_\_\_\_\_

(Дата)

(Подпись)

Замечания: \_\_\_\_\_

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

Москва

2021

**Цель:** Получить практические навыки в создании классов и их последующем использовании.

**Задания:**

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.

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.

6. В игре бридж каждому из игроков раздают 13 карт, таким образом колода расходуется полностью. Модифицируйте программу `CARDARRAY` 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.

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()` (заметим, что здесь название функции начинается с символа подчеркивания — заголовочные файлы `STDLIB.H` или `MATH.H`) для преобразования новой строки к числу типа `long double`. Предполагаем, что денежное значение не может быть отрицательным. Напишите функцию `main()` для проверки метода `mstold()`, которая несколько раз получает денежную строку от пользователя и выводит соответствующее число типа `long double`.

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.

```
safearray sal: //описываем массив int temp = 12345:
//описываем целое sal.putel (7, temp ); //помещаем
значениetemp 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 `safearray` похожей на работу встроенных массивов.

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» (LIFO), 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 `pop()` 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.

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. Поле 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
```

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.

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 `bMoney` 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. Вот как может выглядеть определение класса:

```
class bMoney { private: long
double money; public:
bMoney();
bMoney(char s[]);
void madd(bMoney m1, bMoney m2);
void
getmoney(); void putmoney();
```

```
}
```

## Код:

```
#include <iostream>
#include <iomanip>
#include <string>
#include <sstream>
#include <queue>

using namespace std;

//+++++++4 работает
int maxint(int num[], int n)
{
    int max = 0, index = 0;
    for (int j = 0; j < n; ++j)
    {
        if (num[j] > max)
        {
            max = num[j];
            index = j;
        }
    }
    return index;
}

void fun4() {
    int maxint(int[], int);
    int arr[1000];
    int n{ 0 };
    int maxindex, maxnum;
    char ans;
    do
    {
        cout << "\nEnter: "; cin >> arr[n++];
        cout << "\nContinue? (y/n)"; cin >> ans;
    } while (ans != 'n');

    maxindex = maxint(arr, n);
    maxnum = arr[maxint(arr, n)];
    cout << "\nThe greatest element: " << maxnum << endl;
    cout << "Index: " << maxindex << endl;
}

//+++++++5 работает
class Fraction {
private:
    int numerator;
    int denominator;
public:
    Fraction() : numerator(1), denominator(1) {}
    void input() {
        std::cout << "Enter fraction: \n";
        std::cin >> numerator >> denominator;
    }
    void f_add(Fraction first, Fraction second) {
        numerator = first.numerator * second.denominator + first.denominator *
second.numerator;
        denominator = first.denominator * second.denominator;
    }
    void f_mult(Fraction value1, int n) {
        denominator = value1.denominator * n;
    }
    void output() {
```

```

        std::cout << "\nAnswer is: " << numerator << "/" << denominator << "\n";
    }
};

void fun5() {
    Fraction arr[100];
    Fraction buf, sum;
    char ch;
    int n{ 0 };
    do {
        arr[n++].input();
        std::cout << "Continue'? (y/n)\n";
        std::cin >> ch;
    } while (ch != 'n');
    for (int j = 0; j < n; j += 2) {
        buf.f_add(arr[j], arr[j + 1]);
        sum.f_add(buf, sum);
    }
    sum.f_mult(sum, n + 1);
    sum.output();
}

//+++++++6 работает

enum Suit { clubs, diamonds, hearts, spades };
//from 2 to 10 are integers without names
const int jack = 11;
const int queen = 12;
const int king = 13;
const int ace = 14;
////////////////////////////////////
class card
{
private:
    int number;
    Suit suit;          //clubs, diamonds, hearts, spades
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;
    }
}

```



```

    }
}

void fun6() {
    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 << "\nOrdered deck:\n";
    for (j = 0; j < 52; j++)
    {
        deck[j].display();
        cout << " ";
        if (!(j + 1) % 13) //newline every 13 cards
            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;
    }
    cout << "\nNew deck:\n";

    card deck1[13], deck2[13], deck3[13], deck4[13];
    for (j = 0; j < 13; j++)
    {
        int i = 0;
        deck1[i] = deck[j];
        deck1[i].display();
        cout << " ";
        i++;
    }
    cout << endl;
    for (j = 13; j < 26; j++)
    {
        int i = 0;
        deck2[i] = deck[j];
        deck2[i].display();
        cout << " ";
        i++;
    }
    cout << endl;
    for (j = 26; j < 39; j++)
    {
        int i = 0;
        deck3[i] = deck[j];
        deck3[i].display();
        cout << " ";
        i++;
    }
    cout << endl;
    for (j = 39; j < 52; j++)
    {
        int i = 0;
        deck4[i] = deck[j];
        deck4[i].display();
    }
}

```

```

        cout << " ";
        i++;
    }
    cout << endl;
}

//+++++++7 работает
class Money
{
private:
    long double for_returning;
    string capital;
public:
    Money() :for_returning(0), capital("nothing")
    {}
    long double mstold()
    {
        int x = 0;
        std::cout << "\nEnter the string: $";
        std::cin >> capital;
        int dot_position = capital.find('.');

        if (dot_position == -1) {
            capital += ".00";
        }

        for_returning = stod(capital);

        int lenght = capital.length();
        for (int j = dot_position; j > 0; j--)
        {
            x++;
            if (!(x % 3))
            {
                capital.insert(j - 1, ",");
            }
        }
        if (!(dot_position % 3))
        {
            capital.erase(0, 1);
        }
        for (int j = dot_position; j < lenght; j++)
        {
            int max_lenght = dot_position + 19;
            if (j > max_lenght)
            {
                capital.erase(j, 1);
            }
        }
        return for_returning;
    }
    void display()
    {
        std::cout << "Your string: $" << capital << std::endl;
    }
};

void fun7()
{
    Money money;
    money.mstold();
    money.display();
}

```

```

//+++++++8 работает
const int LIMIT = 100; //do not rename
class Safearray
{
private:
    int array[LIMIT];
public:
    void putel(int n, int temp) //do not rename
    {
        if ((n < 0) or (n > LIMIT - 1)) //test
        {
            cout << "Achtung!" << endl;;
            system("pause");
            exit(1);
        }
        array[n] = temp;
    }
    int getel(int n) //do not rename
    {
        if ((n < 0) or (n > LIMIT - 1)) //test
        {
            cout << "Achtung!" << endl;;
            system("pause");
            exit(1);
        }
        return array[n];
    }
};

void fun8()
{
    Safearray sal;
    int temp, n;
    cout << "temp:\n";
    cin >> temp;
    cout << "n:\n";
    cin >> n;
    sal.putel(n, temp);
    temp = sal.getel(7);
    cout << "New temp: " << temp << endl;
}

```

```

//+++++++9 работает
#include <iostream>
using namespace std;

class Queue
{
private:
    enum { MAX = 10 }; // немного нестандартный синтаксис
    int st[MAX];       // стек в виде массива
    int head;
    int tail;          // вершина стека
public:

    Queue()            // конструктор
    {
        head = tail = 0;
    }
    void put(int var) // поместить в стек
    {
        st[tail++] = var;
    }
}

```

```

    }
    int get()          // ВЗЯТЬ ИЗ СТЕКА
    {
        return st[head++];
    }
};

void fun9() //не трогать
{
    Queue s1;

    s1.put(11);
    s1.put(22);
    cout << "1: " << s1.get() << endl;
    cout << "2: " << s1.get() << endl;

    s1.put(33);
    s1.put(44);
    s1.put(55);
    s1.put(66);
    cout << "3: " << s1.get() << endl;
    cout << "4: " << s1.get() << endl;
    cout << "5: " << s1.get() << endl;
    cout << "6: " << s1.get() << endl;
    cin.get();
}

//+++++++10 работает
class Matrix
{
private:
    int array[10][10];
public:
    int a, b;
    const int LIMIT = 10;
    Matrix(int r, int k) : a(r), b(k)
    {}
    void putel(int r, int k, int temp)
    {
        if ((r < 0) or (r > LIMIT - 1) or (k > LIMIT - 1) or (k < 0))
        {
            cout << "Achtung!" << endl;
            system("pause");
            exit(1);
        }
        array[r][k] = temp;
    }
    int getel(int r, int k)
    {
        if ((r < 0) or (r > LIMIT - 1) or (k > LIMIT - 1) or (k < 0))
        {
            cout << "Achtung!" << endl;
            system("pause");
            exit(1);
        }
        return array[r][k];
    }
};

void fun10()
{
    Matrix random_name(3, 4);
    int temp = 12345;
    random_name.putel(7, 4, temp);
    temp = random_name.getel(7, 4);
    cout << temp << endl;
}

```

```

}
//+++++++11 работает
class money
{
private:
    string sum1;
    string sum2;
    long double capital;
public:
    void getString()
    {
        long double constexpr limit = 9999999999999999.00; //limit
        cout << "Enter the amount: \n";
        cin >> capital;
        //test
        if (capital > limit) {
            cout << "Error! The gold vault is overflowing";
            exit(1);
        }
    }
    void ldtoms()
    {
        stringstream stream;
        stream << fixed << setprecision(2) << capital;
        stream >> sum1;
    }
    void strings()
    {
        string token("$");
        sum2 = sum1;
        sum2 = token + sum2;
        for (int i = sum2.size() - 6; i > 1; i -= 3)
            sum2.insert(i, " ");
    }
    void display()
    {
        cout << sum2 << "\n";
    }
};

void fun11()
{
    money babki;
    char ch = 'y';
    do
    {
        babki.getString();
        babki.ldtoms();
        babki.strings();
        babki.display();
        cout << "Continue? (y/n)\n";
        cin >> ch;
    } while (ch != 'n');
}
//+++++++12 работает
class bMoney
{
private:
    string capital;
    long double number;
public:
    bMoney() : number(0)
    {}
    void getString()

```

```

{
    cout << "Enter the amount (dollar sign in the end)" << endl;
    getline(cin, capital, '$');

    int wlen = capital.length();
    int n = 0;
    string num;

    for (int j = 0; j < wlen; j++)
        if (capital[j] != ',' && capital[j] != '$')
            num.push_back(capital[j]);
    number = stold(num);
}
void add(bMoney mon1, bMoney mon2)
{
    number = mon1.number + mon2.number;
}
void display()const
{
    cout << setiosflags(ios::fixed)
        << setiosflags(ios::showpoint)
        << setprecision(2)
        << "\n" << number << '$' << endl;
}
};

void fun12()
{
    bMoney money1, money2, money_sum;
    char ch = 'y';
    do
    {
        money1.getString();
        money2.getString();
        money_sum.add(money1, money2);
        money_sum.display();
        cout << "Continue? (y/n)\n";
        cin >> ch;
    } while (ch != 'n');
}

```

```

int main()
{
    setlocale(LC_ALL, "Russian");
    int nomer;
    cout << "\nВведите номер задачи\n";
    cin >> nomer;
    switch (nomer) {
    case 4:
        fun4();
        break;
    case 5:
        fun5();
        break;
    case 6:
        fun6();
        break;
    case 7:
        fun7();
        break;
    case 8:
        fun8();

```

```
        break;
    case 9:
        fun9();
        break;
    case 10:
        fun10();
        break;
    case 11:
        fun11();
        break;
    case 12:
        fun12();
        break;
}

return 0;
}
```