МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«ВЛАДИВОСТОКСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
(ФГБОУ ВО «ВВГУ»)

ИНСТИТУТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И АНАЛИЗА ДАННЫХ
КАФЕДРА ИНОРМАЦИОННЫХ ТЕХНОЛОГИЙ И СИСТЕМ

# ОТЧЕТ
# ПО ЛАБОРАТОРНОЙ РАБОТЕ №6

по дисциплине
«Информатика и программирование»

Студент
гр. БИН-25-2 _____ Шумейко И.А

Ассистент
Преподавателя _____ Водяницкий М.В

Владивосток 2025

# Задание

Задание 1

Написать функцию, которая конвертирует время из одной величины в другую.

На вход подается:

1. число (величина времени)
2. исходная единица измерения
3. единица измерения, в которую нужно перевести

Функция должна вернуть конвертированное значение.

Задание 2

Пользователь делает вклад в банке в размере `a` рублей сроком на `n` лет

Процент по вкладу зависит от суммы и срока.

Зависимость от суммы:

Каждые 10 000 рублей увеличивают ставку на 0.3%

но суммарное увеличение не может превышать 5%

минимальный вклад - 30 000 рублей

Зависимость от срока:

1. первые 3 года - 3%
2. от 4 до 6 лет - 5%
3. более 6 лет - 2%

Необходимо написать функцию, которая рассчитывает прибыль пользователя без учета первоначально вложенной суммы.

Используется сложный процент: каждый год процент начисляется на текущую сумму вклада.

На вход подаются: сумма вклада и количество лет. Результат: сумма прибыли (не весь вклад, а только заработанные проценты).

Задание 3

Написать функцию для вывода всех простых чисел в заданном диапазоне. Нужно учитывать некорректные данные (например, начало больше конца или диапазон без простых чисел).

На вход подаются два числа: начало и конец диапазона (включительно). На выходе - список всех простых чисел или сообщение об ошибке.

(Формат вывода списка простых чисел может быть любым удобным: в строку через пробел, в несколько строк и т. п.).

Задание 4

Реализовать функцию сложения двух матриц.

При сложении двух матриц получается новая матрица того же размера, где каждый элемент - это сумма элементов с тем же индексом из двух исходных матриц

Ограничения:

1. складывать можно только матрицы одинакового размера
2. размер матрицы должен быть строго больше 2 (например, 3×3, 4×4 и т.д.)
3. при нарушении условий нужно вывести сообщение об ошибке

На вход подаются:

1. размер матрицы `n` (для квадратной матрицы `n × n`)
2. элементы первой матрицы (по строкам, через пробел)
3. элементы второй матрицы в таком же формате

Задание 5

Написать функцию, которая определяет, является ли строка палиндромом. Палиндром - это строка, которая читается одинаково слева направо и справа налево (обычно без учета пробелов, регистра и знаков препинания - эти правила нужно явно задать в своей реализации)

На вход подается строка. На выходе:

1. «Да», если это палиндром
2. «Нет», если это не палиндром

# Содержание

# 1 Выполнение работы

## 1.1 Задание 1

Код для данного задания представлен на рисунке 1.

```
double TimeTrans(double time,int32_t modificator){
   if(modificator < 0){
      for(;modificator != 0;++modificator){
         time *= 60.;
      }
   }else if(modificator > 0){
      for(;modificator != 0;--modificator){
         time /= 60.;
      }
   }

   return time;
}

void F1(){
   std::unordered_map<uint8_t,int32_t> times{{'s',1},{'m',2},{'h',3}};

   std::cout << "Введите текст в формате \"время единица_измерения
единица_измерения\":\n";

   double num;
   uint8_t ch1,ch2;
   std::cin >> num >> ch1 >> ch2;

   std::cout << TimeTrans(num,times.at(ch2) - times.at(ch1)) << ' ' << ch2;
}
```

Рисунок 1 — код для задания 1

Для выполнения задания создадим карту times, где ключ — обозначение единицы времени, а значение какие-то числа, разница между которыми показывает какое количество раз нужно умножить или разделить время time на 60, чтобы получилось число в соответствующей единице измерения. И так считываем из ввода число и два символа, находим разницу между числами по ключам в times, и вызываем функцию TimeTrans, которая соответствующее количество раз делит или умножает время на 60.

## 1.2 Задание 2

Код для данного задания представлен на рисунке 2.

```
double SumPersent(double init_sum, int32_t years){
  double persent = 1.;
  persent += init_sum / 10000 * 0.003;
  if(years <= 3){
    persent += 0.03;
  }else if(years > 3 && years <= 6){
    persent += 0.05;
  }else{
    persent += 0.02;
  }

  for(;years != 0; --years){
    init_sum *= persent;
  }

  return init_sum;
}

void F2(){
  std::cout<< "Введите сумму и срок:\n";
  double sum; int32_t years;
  std::cin >> sum >> years;
  if(sum < 30000){
    throw std::logic_error("Минимальная сумма 30000");
  }else if(years < 1){
    throw std::logic_error("Нереальный срок");
  }

  std::cout << SumPersent(sum,years) - sum;
}
```

Рисунок 2 — код для задания 2

Для выполнения задания создадим функцию SumPerscent, где сначала прибавим к переменной persent сумму вклада деленную на 10 000 и умноженную на процент, который дается за каждые 10 000 рублей изначальной суммы. Затем используем конструкцию if-else и добавим дополнительный процент и вернем или изначальную сумму или умноженную на процент years раз. Затем примем на вход сумму и срок, ели сумма меньше 30 тысяч выведем ошибку,

если срок меньше года — ошибку. Если всё отлично — выведем значение из функции SumPercent.

### 1.3 Задание 3

Код для выполнения задания представлен на рисунке 3.

```
void F3(){
  std::cout << "Введите начало и конец и диапазона:";
  int32_t begin, end;
  std::cin >> begin >> end;
  if(end > begin){
    throw std::logic_error("начало больше конца");
  }else if(begin == end){
    throw std::logic_error("ёрмунганд");
  }
  std::vector<int32_t> i_vec;
  i_vec.reserve(static_cast<size_t>(begin - end));

  for(;begin != end;++begin){
    bool is_pure = true;
    for(int i = 2; i != begin;++i){
      if(begin % i == 0){
        is_pure = false;
        break;
      }
    }
    if(is_pure){
      i_vec.push_back(begin);
    }
  }
  if(i_vec.size() == 0){
    throw std::logic_error("диапазон без простых чисел");
  }else{
    for(auto& val: i_vec){
      std::cout << val << ' ';
    }
    std::cout << '\n';
  }
}
```

Рисунок 3 — код для задания 3

Для выполнения задания примем на ввод диапазон чисел, пройдемся циклом по каждому числу, при этом каждое число через цикл попытаемся разде-

лить на каждое, что меньше него, если получилось — число составное, значит не добавляем его в вектор i_vec, если число простое — добавляем. В конце, если вектор i_vec пустой пишем ошибку, иначе выводим все числа из вектора.

## 1.4 Задание 4

Код для выполнения задания представлен на рисунках 4.1, 4.2, 4.3.

```
class Matrix{
public:
  Matrix(std::vector<std::vector<double>> value):matrix_(std::move(value)){}
  Matrix(size_t M, size_t N){
    matrix_ = std::vector<std::vector<double>>(M,std::vector<double>(N,0.));
  }
  Matrix(const Matrix& mat){matrix_ = mat.matrix_;}
  Matrix(Matrix&& mat):matrix_(std::move(mat.matrix_)){}

  Matrix& operator= (Matrix&& mat){
    matrix_ = std::move(mat.matrix_);
    return *this;
  }

  Matrix& Swap(Matrix& mat){
    std::swap(matrix_,mat.matrix_);
    return *this;
  }

  Matrix& operator= (const Matrix& other){
    if(this != &other){
      Matrix temp(other);
      Swap(temp);
    }
    return *this;
  }

  bool IsAvaliableToAdd(const Matrix& other)const{
    return Size() == other.Size();
  }
```

Рисунок 4.1 — часть кода для задания 4

На рисунке 4.1 создан класс Matrix, который содержит вектор из векторов, которые содержат число double. Для этого класса определяем конструкторы из двух чисел, копирование и перемещения, а так же соответственные операторы и метод IsAvaliableToAdd для проверки возможности сложения.

```
  std::pair<size_t,size_t> Size()const{
     if(matrix_.size() != 0){
       return {matrix_.size(),matrix_.at(0).size()};
     }else{
       return {0,0};
     }
   }

   Matrix& operator+= (const Matrix& other){
     if(IsAvaliableToAdd(other)){
       for(size_t a = 0; a != other.matrix_.size();++a){
         for(size_t b = 0; b != other.matrix_.at(0).size();++b){
           matrix_[a][b] += other.matrix_[a][b];
         }
       }
     }

     return *this;
   }

   double& get(size_t n,size_t k){
     return matrix_[n][k];
   }
   const double& take(size_t n,size_t k)const{
     return matrix_.at(n).at(k);
   }

private:
   std::vector<std::vector<double>> matrix_;
};
Matrix operator+ (Matrix mat1, Matrix mat2){
   return mat1 += mat2;
}
```

Рисунок 4.2 — часть кода для задания 4

На рисунке 4.2 Для класса Matrix определяем метод Size для получения размера матрицы, оператор сложения двух матриц, методы get, который позволяет изменять число в матрице, метод take, который позволяет только посмотреть число находящееся под каким - то номером.

```
std::istream& operator>> (std::istream& is,Matrix& mat){
  for(size_t i = 0; i != mat.Size().first && is.good(); ++i){
    for(size_t b = 0; b != mat.Size().second && is.good();++b){
      if(is.eof()){
        std::cout << "Не все числа заданы для матрицы данного размера";
        return is;
      }
      if(is.bad()){
        std::cout << "Ошибка ввода одного или нескольких чисел";
        return is;
      }
      double value;
      is >> value;
      mat.get(i,b) = value;
    }

  }
  return is;
}

std::ostream& operator<< (std::ostream& os, const Matrix& mat){
  for(size_t a = 0; a != mat.Size().first; ++a){
    for(size_t b = 0; b != mat.Size().second; ++b){
      os << mat.take(a,b) << ' ';
    }
  }
  return os;
}

void F4(){
  std::cout << "Напишите размер матриц и ниже, рязделяя числа пробелами сами
матрицы:\n";
  size_t N;
  std::cin >> N;
  if(N < 2){
    std::cout << "Минимальный размер матрицы - 2";
    return;
  }
  Matrix mat1(N,N),mat2(N,N);
  std::cin >> mat1 >> mat2;

  std:: cout << mat1 + mat2;
}
```

Рисунок 4.3 — код для задания 4

На рисунке 4.3 определяем оператор ввода из потока, для считывания чисел из матрицы, а так же оператор вывода для вывода чисел из матрицы. Затем сохраняем размер двух матриц в переменной N, если N равна 2, то выводим ошибку, если нет, то принимаем на ввод матрицы и выводим сложения.

## 1.5 Задание 5

Код для выполнения задания представлен на рисунке 5.

```
void F5(){
  std::string str;
  std::wcout << L"Введите строчку:\n";
  std::getline(std::cin,str);
  for(auto it = str.begin(); it != str.end(); ++it){
    if(*it == ' '){
      str.erase(it);
    }
  }

  auto it1 = str.begin();
  auto it2 = std::prev(str.end());
  for(;it1 < it2;++it1,--it2){
    if(tolower(*it1) != tolower(*it2)){
      std::cout << "Нет";
      return;
    }
  }
  std::wcout << L"Да";
}
```

Рисунок 5 — код для задания 5

Для выполнения задания принимаем на ввод строчку, размер делим на два и сохраняем в n, создаем два string_view по строчке, у первого убираем n элементов с конца, у второй — с начала, затем циклом проходим по элементам string_view и сравниваем их, приводя к нижнему регистру.