**Лекция 1**

Архитектура Фон-Неймана.

Архитектура компьютера определяется процессором, набором его регистров и доступным командами.

Система программирования (СП) = ide + programming language + compiler. Программа является исполнителем и воспринимает только доступные системой ему команд и может ее расширять ее комбинацией команд. Процессор логически состоит из регистров, а физически из трансляторов.

**Лекция 2**

Денис Ритчи один из создателей языка С. Он чувствителен к регистру. Позволяет реализовать модульное программирование. Все программы в C являются функциями. Подключение библиотек, константы и макросы (функции автоматической подстановки) происходит на этапе препроцессорной обработки.

Структура программы на C глобально:

1. Препроцессорная часть.
2. main function.
3. custom functions.

Структура программы на C подробно:

#include <stdio.h>

// const и макросы

int main() {

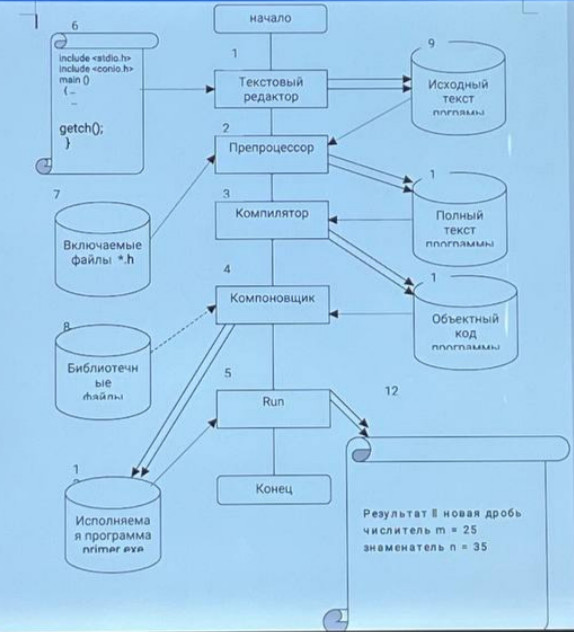
// code

return 0;

}

// описание остальных функций

**Алгоритм обработки программ на C.**



2 этап. Препроцессорная обработка — это преобразования текста программы до ее компиляции. Правила препроцессорной обработки определяет программист директивами (библиотеками) препроцессора.

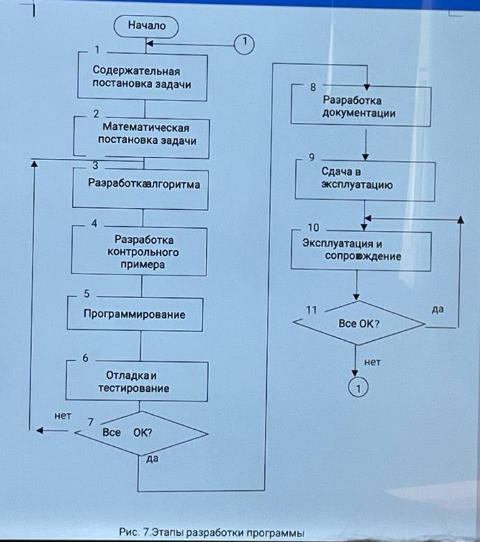
3 этап. Компиляция – это трансляция программы на язык близкий к машинному. На данном этапе показываются синтаксические ошибки.

4 этап. Компоновщик (linker) – это компоновка или редактирование связей.

К программе подключаются библиотечные функции и получается программа, готовая к исполнению.

Каждая программа на С – это последовательность препроцессорных директив (библиотек), описаний и определений глобальных объектов и функций.

Этапы разработки программы:



Недостаток словесной подстановки – это низкая степень формализации.

Математическая подстановка задачи – это высокая степень формализации. Задача формулируется в абстрактных математических формулах и соотношениях. Выбираются переменные, записываются ограничения, связи между переменными, уточняются диапазоны представления входных данных и результатов. Грамотная математическая подстановка позволяет отнести задачу к определенному классу задач и тем самым помогает найти алгоритм решения.

Контрольный пример (КП) – это подготовка набора исходных данных и соответствующих им результатов. С помощью КП программист проводит отладку и тестирование.

Программа – это описание данных и алгоритмов на алгоритмическом языке для исполнителя, которым является компьютер + СП

Отладка – это получение исполняемой программы, выполнение которой компьютером обеспечивает решение поставленной задачи.

Основной прием поиска ошибок является тестирование. Т.е. выполнение программы на наборе исходных данных, для которых заранее известен результат. Отладка может занимать до 70% времени от всех этапов на создание программы. Тестирование часто выполняется на данных КП

Базовое понятие языка программирования – это алфавит, идентификатор, ключевые слова и const. На этапе компиляции выполняется лексический анализ текста и выделяются лексемы. Лексема – это единица текста программы, которая при компиляции воспринимается как единое целое и по смыслу не может быть разделена.

К лексемам относят: id, keywords, const, operation symbols, dividers.  
Одно из основных понятий в С это объект, т.е. именованная область памяти. Частный случай объекта это переменная. Ее отличает то, что имени переменной можно связывать различные значения, совокупность которых определяется типом переменной.

**Лекция 3**

Работа с массивами

Массив – это упорядоченное последовательность однотипных элементов, расположенная по определнному адресу и имеющее имя.

[][][][][] А(N), N = 6

012345 - адреса массива

Весь массив характеризуется:

1. именем;
2. типом данных;
3. размером;
4. размерностью – кол-во индексов, необходимых и достаточных для идентификации конкретного элемента массива;
5. адресом.

Каждый элемент массива характеризуется:

1. значением;
2. порядковый номером (индекс/позиция);
3. адресом в памяти (отсчитывается от 1 элемента массива;
4. типом данных (наследуется от типа всего массива.

Требования к отчетам по лаб. на массивы:

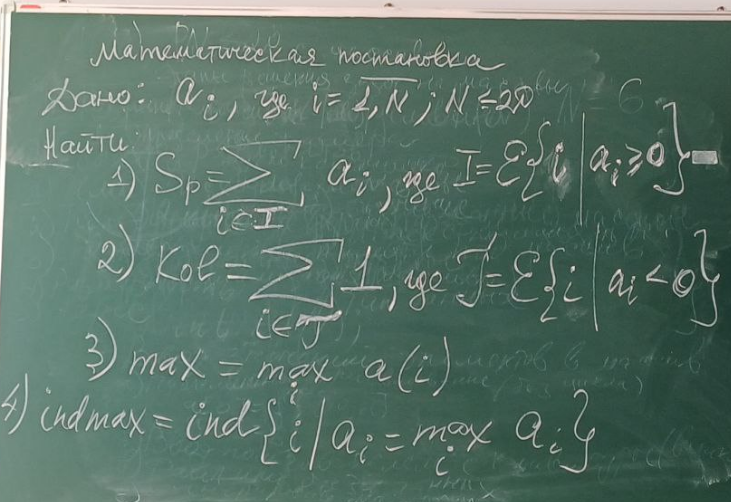
1. словесная подстановки задачи;
2. мат. постановка задач;
3. контрольный пример;
4. блок-схема;
5. листинг программы/код программы;
6. вывод результатов;
7. выводы по работе.

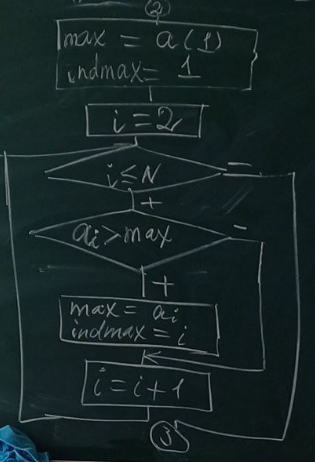
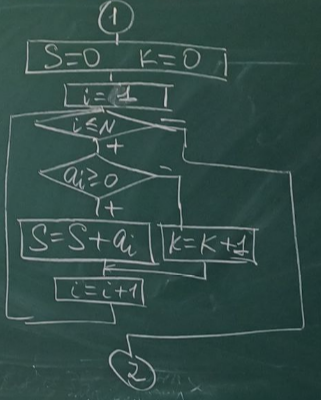
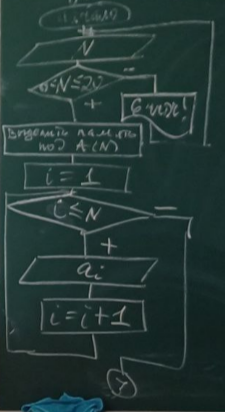
Дан целочисленный массив А(N), N <= 20. Найти сумму положительных элементов и количество отрицательных, а также макс. элемент массива и его номер.

Этапы решения задания на массивы:

1. создание массива(-ов):
   1. определить размер массива:
      1. задает константно. Например, #define N 6
      2. задает сам пользователь/ввод пользователя
   2. выделение памяти под массив (инициализация массива). Например, int A(N);
   3. вся дальнейшая работа с массивом идет на уровне каждого конкретного элемента. Ввод значений элементов в массив
      1. поэлементное присваивание/без цикла. Например, A[0] = 5; A[1] = 8;
      2. ввод пользователя с клавиатуры (с циклом)
      3. через файл/база данных
      4. сгенерировать случайным образом.
2. решение задачи по условию
3. вывод результатов:
   1. исходные массивы. (подписываем, что выводим в программе)
   2. полученные (преобразованные массивы)
   3. единичные результаты

Для каждого этапа открываем свой цикл



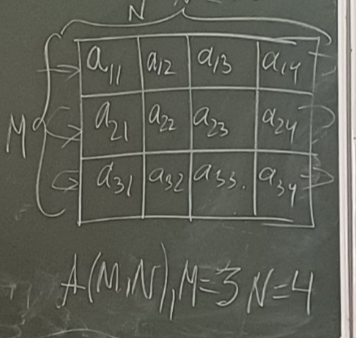
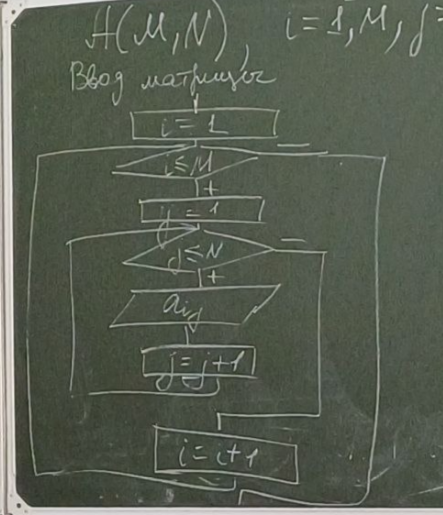


**Лекция 5**

Работа с матрицами

A(M, N), , , где N <= 15 M <= 20

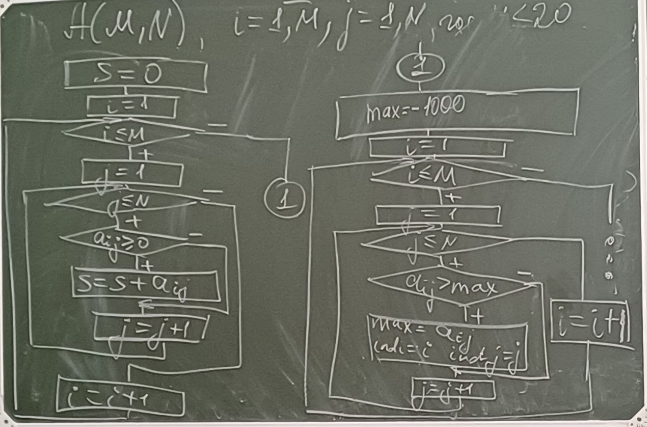
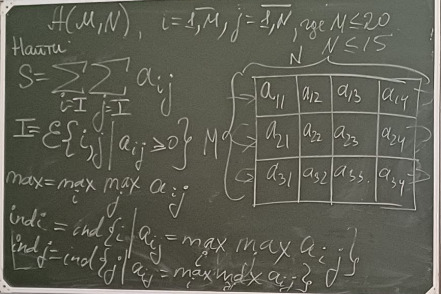
В языке C матрицы хранятся в виде вектора, где элемент строки является указателем на элемент столбца.



**Три основных типа задач на матрицы:**

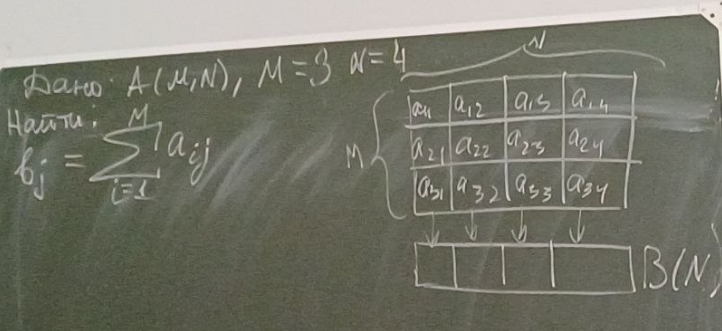
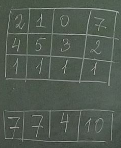
1. Обработка всей матрицы

Найти сумму положительных элементов матрицы, максимальный элемент и его номер.

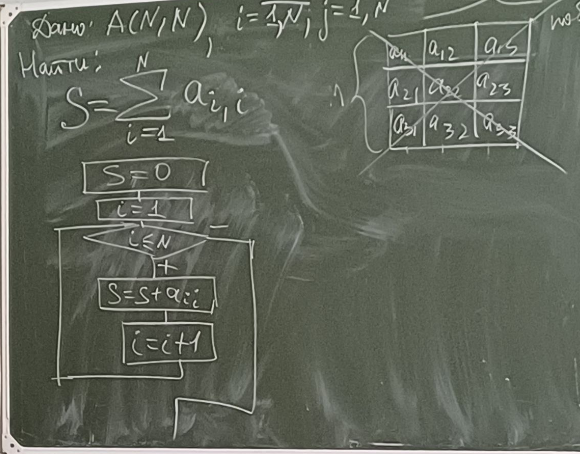


1. Обработка матрицы по строкам или по столбцам

Дана матрица A, M, N построить вектор из сумм элементов по столбцам.

1. Обработка матрицы по диагонали



* 1. На диагонали
  2. Выше диагонали I < J
  3. Ниже диагонали I > J