**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() {

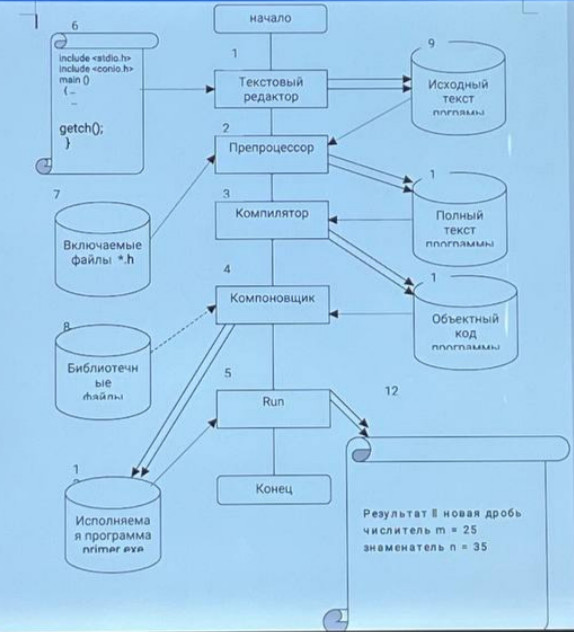
// code

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.

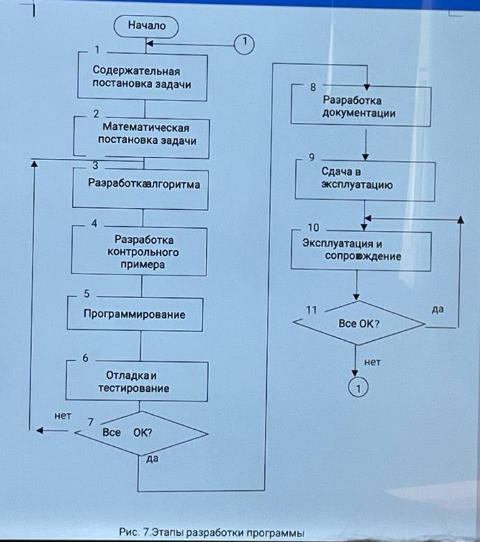
3 этап. Компиляция – это трансляция программы на язык близкий к машинному. На данном этапе показываются синтаксические ошибки.

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:

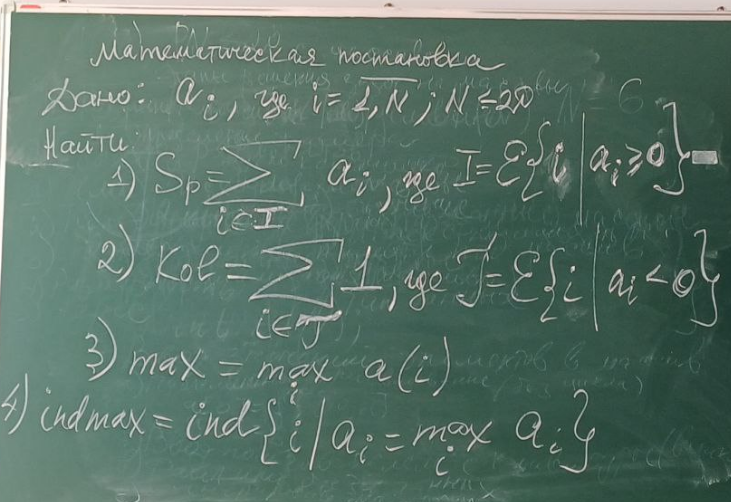
1. словесная подстановки задачи;
2. мат. постановка задач;
3. контрольный пример;
4. блок-схема;
5. листинг программы/код программы;
6. вывод результатов;
7. выводы по работе.

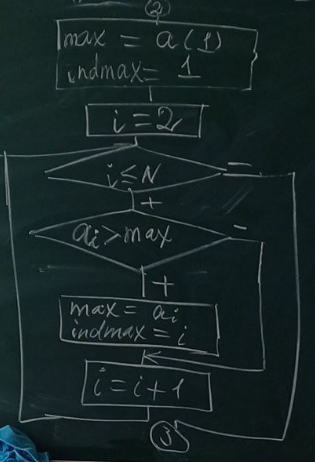
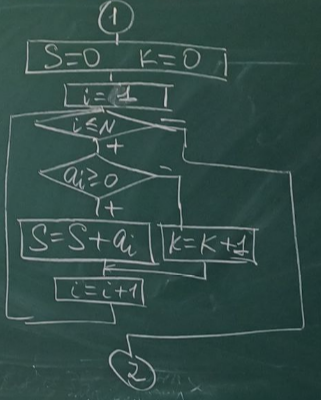
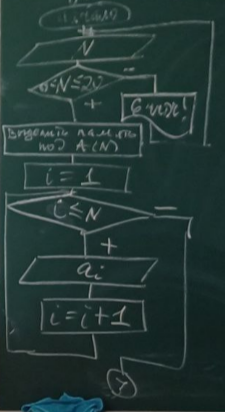
Дан целочисленный массив А(N), N <= 20. Найти сумму положительных элементов и количество отрицательных, а также макс. элемент массива и его номер.

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 4**

Адреса, указатели, массивы

1. Указатели и адреса объектов

Познакомимся с новым для нас объектом - указателем. Указателем называется такой объект программы, который обладает двумя свойствами:

* Значением любого указателя может быть только какой – либо адрес участка оперативной памяти, выделенный под конкретный объект;
* С указателем всегда связан тип объекта, для которого выделен этот участок.

Таким образом, указатель, значение которого определено, указывает адрес объекта в оперативной памяти и его тип, который определяет количество выделенных под этот объект байтов. Другими словами, указатель дает возможность реализации доступа к определенному участку оперативной памяти.

В простейшем случае объявление указателя на некоторый объект имеет вид

type \* имя указателя;

где type – обозначение типа, имя указателя – идентификатор.

Таким образом, признаком указателя при лексическом разборе объявления объекта служит символ \*, помещенный перед именем. Ниже представлены примеры объявления указателей:

int \*p1, \*p2;

float \*p3, \*p4;

В этом примере p1, p2 – это указатели на поле типа int, а p3, p4 – указатели на поле типа float. А какова длина участка поля, на которое могут указывать эти указатели? Для р1, p2 – это 2 байта (поле типа int занимает 2 байта), а для p3, p4

При объявлении указателя возможно выполнить его инициализацию (присвоить ему какое-либо значение). В этом случае возможны две формы объявления:

type \* имя указателя = инициализирующее выражение;

type \* имя указателя = (инициализирующее выражение);

В качестве инициализирующего выражения используется константное выражение, частными случаями которого могут быть:

* явно заданный участок памяти;
* указатель, уже имеющий значение;
* выражение, позволяющее получить адрес объекта с помощью операции & – определения адреса.

int x = 5; // целочисленная переменная

float y = 45.6; // действительная переменная

char c = 'a'; // символьная переменная

char \* p (NULL); //нулевой указатель на объект типа char

char\*p1; //неинициализированный указатель на объект типа char

char \*p2 = &c; // инициализированный указатель на объект типа char

float \*p3 = &y; // инициализированный указатель на объект типа float

int \* p4 = &x; // инициализированный указатель на объект типа int

int \* p5; // неинициализированный указатель на объект типа int

char\*p6; // инициализированный указатель на объект типа char

char int \*\*p7; //неинициализированный указатель на объект типа int \* // p7 - указатель на указатель

float \* pm[5];// массив указателей на тип float из 5 элементов

float (\*pz) [5]:// указатель на массив из 5 элементов типа float

1.2. Типы указателей и операции над ними

Основные типы указателей определяются ключевыми словами char, int, float, long, double, short, unsigned, signed, void, Возможны также типы объектов, описываемые программистом, но с этой возможностью мы будем знакомиться позже. Напомним, что тип void предполагает отсутствие значения, а указатели типа void имеют некоторые особенности, которые должны учитываться при выполнении операций над такими указателями (с ними познакомимся в последующих главах).

Операции над указателями можно представить следующим списком:

* операция разыменования, или доступа по адресу – \* указатель;
* преобразование типов (приведение типов);
* присваивание;
* получение (взятия) адреса – &
* сложение и вычитание (аддитивные операции);
* инкремент, или автоувеличение - (++);
* декремент, или автоуменьшение (--);
* операции отношений (сравнения).

Операция разыменования

Операция разыменования указателя «\*» имеет формат: \*указатель. Операция разыменования обеспечивает доступ к полю, адрес которого содержится в указателе.

#include <stdio.h>

int main() {

int z =20;

int \* p = &z;

printf ("z = %d \*p = %d\n", z, \*p);

\*p = 30;

printf ("z = %d \*p = %d\n", z, \*p);

return 0;

}

Результат: z = 20 \*p = 20, z = 30 \*p = 30

Операция присваивания

Чтобы связать неинициализированный указатель с новым участком памяти, не занятым каким – либо объектом программы, используется оператор new или присваивается указателю явный адрес:

char \*p1, \*p6;

int x,

int \*p5;

p1 = new char;// выделяется участок памяти для переменной

//типа char, значение p1 присваивается

// адресу этого участка

p5 = &x;

// значение р5 присваивается адресу

//участка памяти, выделенного для х

рб = (char \*)

Oxb8000000; // значение рб присваивается адресу

// видеопамяти ПЭВМ для цветного дисплея

// в текстовом режиме

При вычитании из указателя целочисленного значения К, числовое значение

указателя уменьшается на величину

K \* sizeof (type), где – вычитаемое, type – тип объекта, к которому отнесен указатель.

Массивы языков С и С++ обладают очень интересным свойством, отличающим их от массивов в других алгоритмических языках. При объявлении массивов, как и в других алгоритмических языках, им выделяются участки оперативной памяти. Но как только память для массива выделено, имя массива воспринимается как константный (т. е. неизменный) указатель того типа, к которому отнесены элементы массива. Однако для этого указателя две операции выполняются особенным образом:

* операция sizeof, примененная к имени массива, определяет размер выделенного участка памяти для всего массива, а не участка, выделенного для указателя. Например, значение выражения sizeof(x) для последнего примера равно 12
* операция получения адреса, примененная к имени массива, определяет адрес начального (с нулевым индексом) элемента массива.

Во всех остальных случаях значением имени любого массива является адрес первого (нулевого) элемента массива, и это значение невозможно изменить. Таким образом, для любого массива соблюдается соотношение:

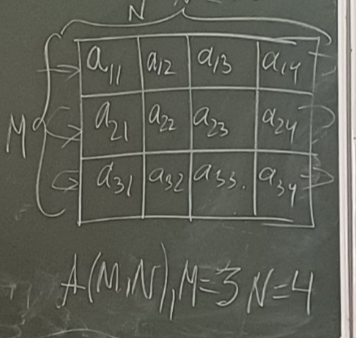
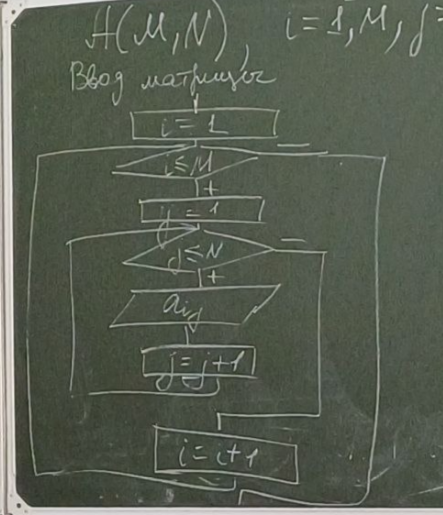
Имя\_массиво == &имя\_массива == имя\_массива [0]

**Лекция 5**

Работа с матрицами

A(M, N), , , где N <= 15 M <= 20

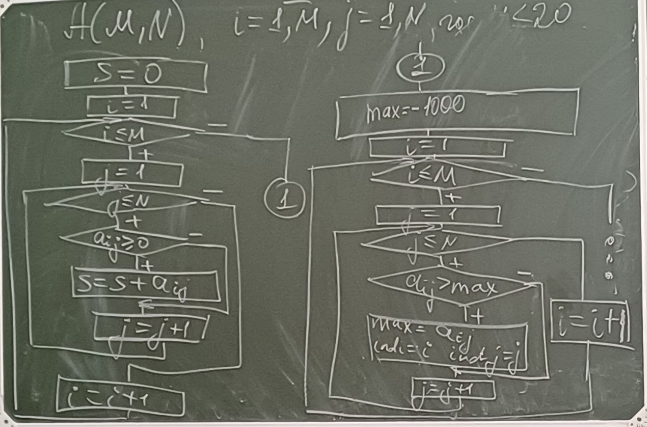
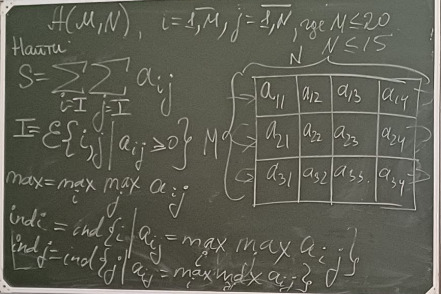
В языке C матрицы хранятся в виде вектора, где элемент строки является указателем на элемент столбца.



**Три основных типа задач на матрицы:**

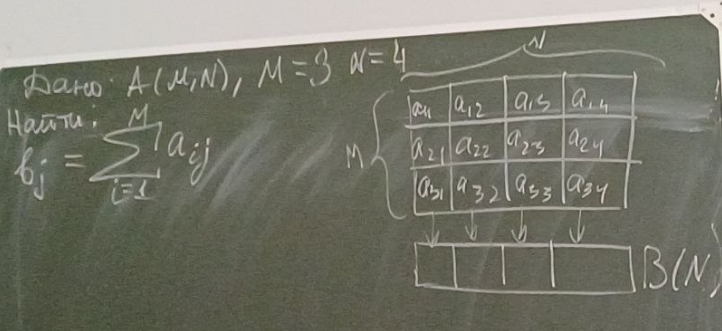
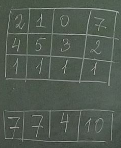
1. Обработка всей матрицы

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

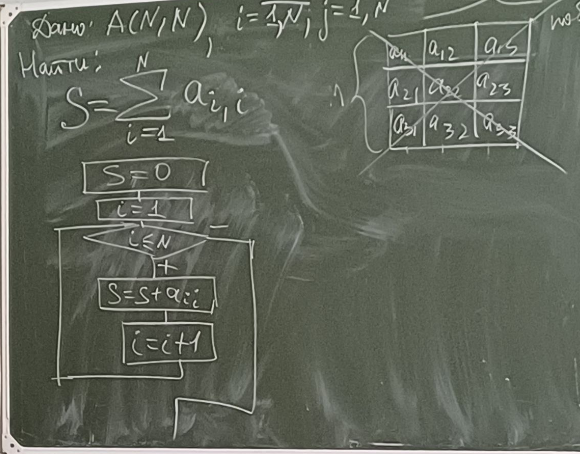


1. Обработка матрицы по строкам или по столбцам

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