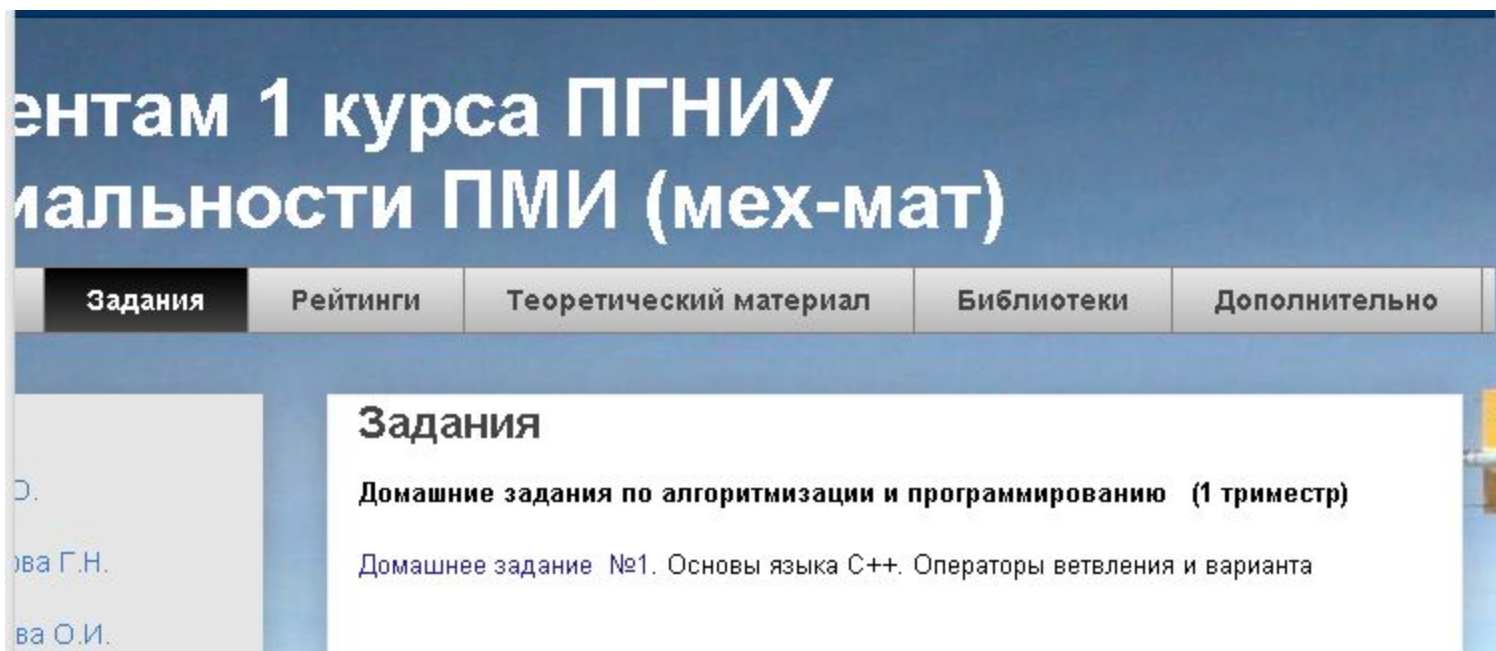
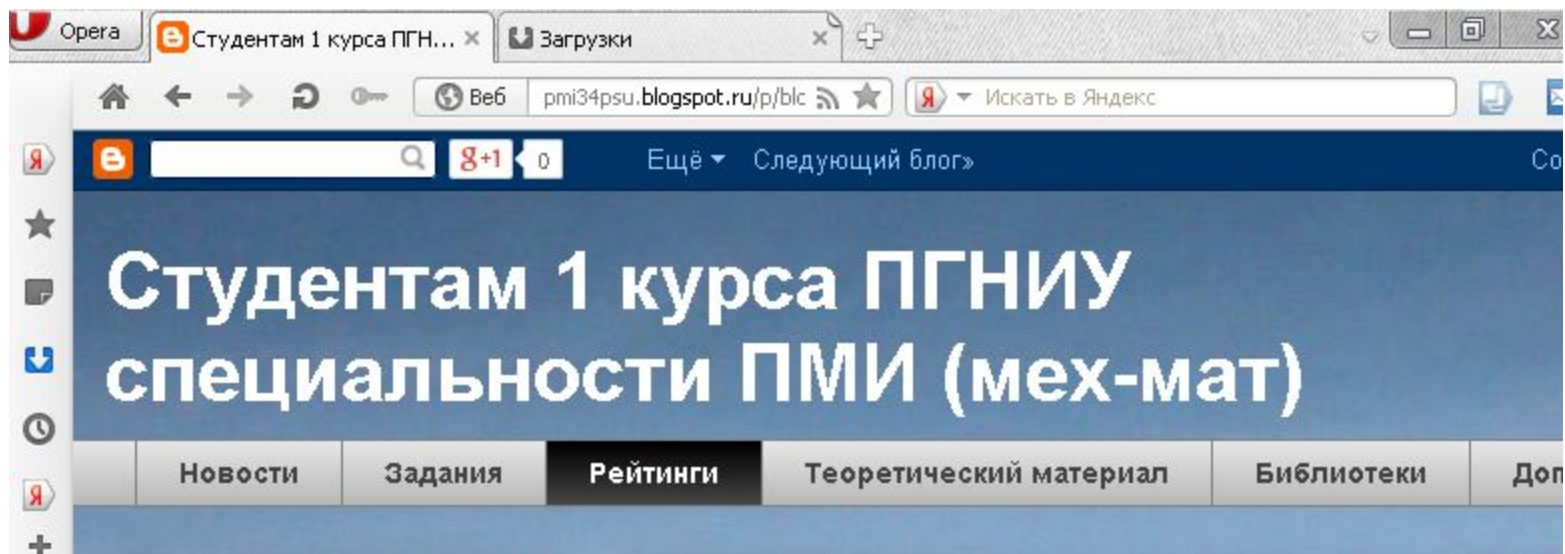


Алгоритмизация и программирование I

- Лекции – 28 часов
- Лабораторные -14 часов
- Контрольные мероприятия -2
- Итоговое контрольное мероприятие - экзамен



Студентам 1 курса ПГНИУ специальности ПМИ (мех-мат)

Новости

Задания

Рейтинги

Теоретический материал

Библиотеки

Авторы

 Шеина Т.Ю.

 Овчинникова Г.Н.

Теоретический материал

Практика 1. Системы счисления

Практика 2. Основы языка C++. Операторы ветвления и варианта

Студентам 1 курса ПГНИУ специальности ПМИ (мех-мат)

Задания

Рейтинги

Теоретический материал

Библиотеки

Дополнительно

Дополнительно

Задачник

Литература

1. Учебник №1 по C++
2. Учебник №2 по C++
3. Учебник №3 по C++
4. Статья по системам счисления и внутреннему машинному представлению из газеты "Информатика"
5. Статья Шестакова А.П. "Представление числовых данных в памяти ЭВМ"

Ссылки

Студентам 1 курса ПГНИУ специальности ПМИ (мех-мат)

Новости

Задания

Рейтинги

Теоретический материал

Библиотеки

Авторы

 [Шейна Т.Ю.](#)

 [Овчинникова Г.Н.](#)

Библиотеки

1 триместр

[PT4One.dll](#)

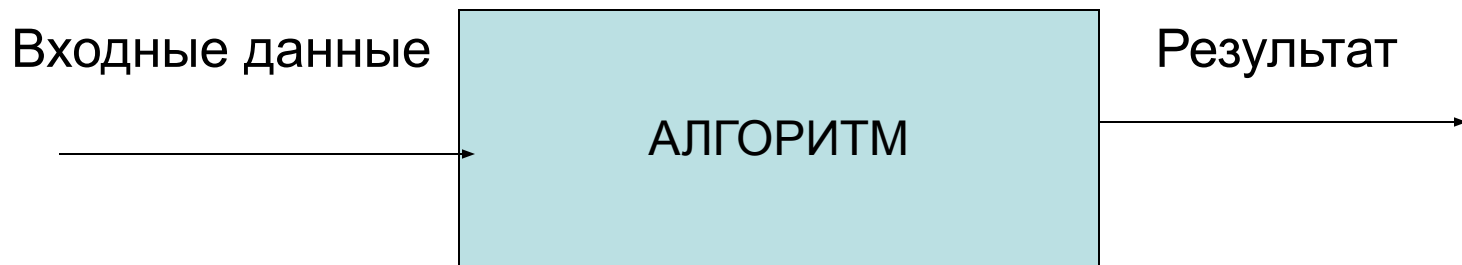
Задания	Рейтинги	Теоретический материал	Библиотеки	Дополнительно
О. ова Г.Н. ва О.И.				среда, 3 сентября 2014 г. <h2>Дистрибутив VisualStudio2005</h2> <p>Уважаемые студенты!</p> <p>Те, кто не смог найти дистрибутивы VS или VC++, может скачать их с компьютера №24 (у дверей в ауд.517) из папки C:/Distr. Необходимо скопировать две папки VS_disk1 и VS_disk2. Серийный номер в первой папке. Сначала необходимо установить VS2005, а затем задачник (в разделе "Дополнительно").</p> <p>Данный дистрибутив устанавливается только с CD-дисков!!! То есть содержимое каждой папки необходимо переписать на отдельный диск и только потом устанавливать. Либо ищите сами другие варианты дистрибутивов в интернет.</p>
СЫЛКИ				

Лекция 1

- Введение в понятие алгоритма.
Свойства алгоритма.
- Этапы решения задачи с
использованием компьютера.
- Языки программирования . Способы
описания языков программирования.
- Структура программы на языке C++.
- Скалярные типы данных.
- Организация ввода-вывода.

Введение в понятие алгоритма. Свойства алгоритма.

- Обработка информации



Понятие алгоритма

- **Алгоритм (по Колмогорову)** – это всякая система вычислений, выполняемых по строго определенным правилам, которая после какого-либо числа шагов заведомо приводит к решению поставленной задачи.
- **Алгоритм (по Маркову)** – это точное предписание, определяющее вычислительный процесс, идущий от варьируемых исходных данных к искомому результату.
- **Алгоритм** – это понятное и точное предписание исполнителю выполнить конечную последовательность команд, приводящую от исходных данных к искомому результату.

Исполнитель алгоритма

- Исполняет алгоритм формально
- Исполняет только команды
- Не задумывается о том какую задачу решает

Свойства алгоритма

1. Дискретность
2. Элементарность шагов
3. Определенность
(детерминированность)
4. Конечность (финитивность)
5. Массовость
6. Понятность

- **Дискретность** означает, что алгоритм состоит из отдельных шагов и эти шаги выполняются в дискретном времени, т.е. любые два последовательных шага разделены при исполнении конечным, ненулевым отрезком времени.
- **Элементарность шагов** означает, что объем работы, выполняемой на отдельном шаге, ограничивается некоторой константой, зависящей от характеристик исполнителя алгоритма и не зависящей от входных данных и промежуточных значений. Для численных алгоритмов такими элементарными шагами могут быть, например, сложение, вычитание, умножение, деление, сравнение двух 32-х разрядных чисел, пересылка одного числа из некоторого места памяти в другое. К элементарным шагам не относится, например, сравнение двух файлов, так как время сравнения зависит от длины файлов, то есть от входных данных.
- **Детерминированность (определенность, точность)** алгоритма означает, что для каждого шага по набору исходных данных могут быть однозначно вычислены результаты выполнения шага, и эти результаты не зависят ни от каких случайных факторов.

- **Конечность (финитивность, результативность)** алгоритма означает, что для получения результата нужно выполнить конечное число шагов, т.е. исполнитель в некоторый момент времени должен остановиться.
- **Массовость** алгоритма означает, что входные данные для алгоритма могут быть выбраны из некоторого множества значений. Если же входные данные уникальны, то алгоритм в силу свойства определенности (детерминированности) всегда будет давать один и тот же результат и само построение алгоритма теряет смысл.
- **Понятность** алгоритма означает, что алгоритм должен быть записан с помощью команд, входящих в ***Систему команд исполнителя (СКИ)***. Чтобы исполнитель мог достичь поставленной цели, пользуясь алгоритмом, он должен уметь выполнять каждое его указание, т. е. понимать каждую из команд, из которых состоит алгоритм.

Данные

- **Данные** – это информация, представленная в виде, пригодном для ее передачи и обработки автоматическими средствами (в том числе компьютером).

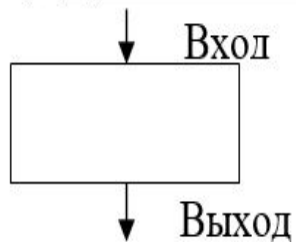
Способы записи алгоритмов

- Естественный язык
- Язык блок-схем
- Язык исполнителя (алгоритмический язык)

Основные уп

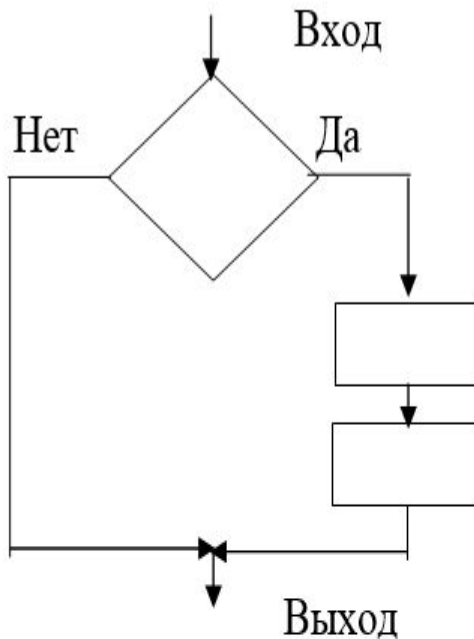
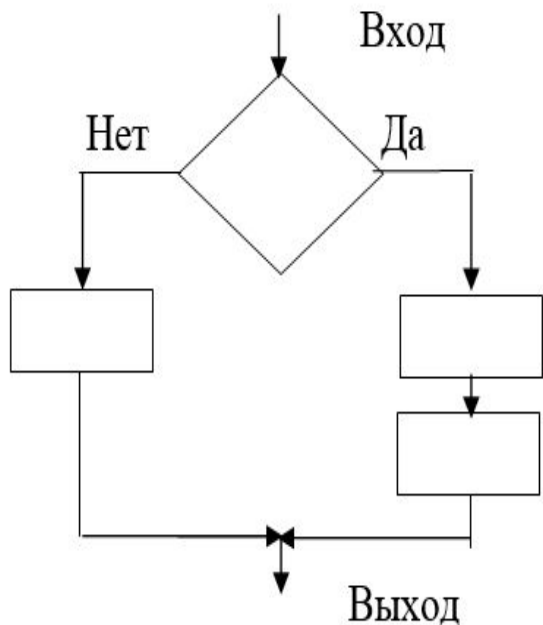
Неполное ветвление:

1) Простое следование:

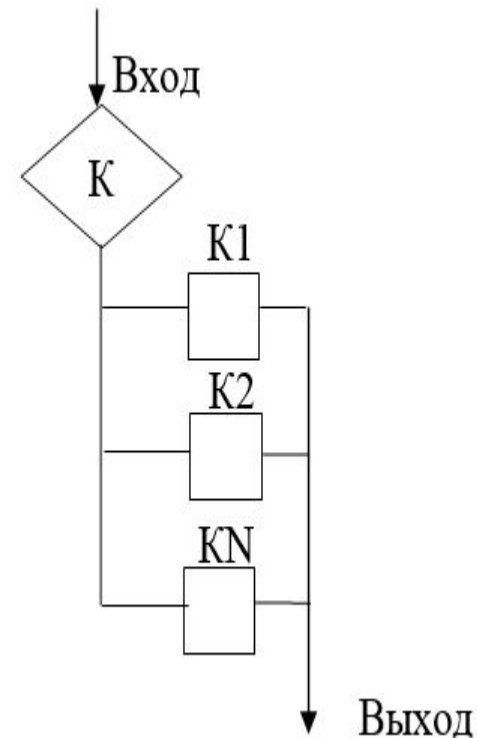
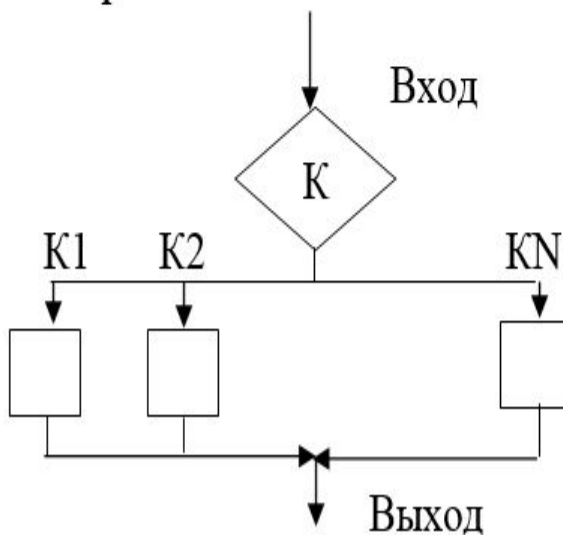


2) Альтернатива:

Полный выбор из двух вариантов:

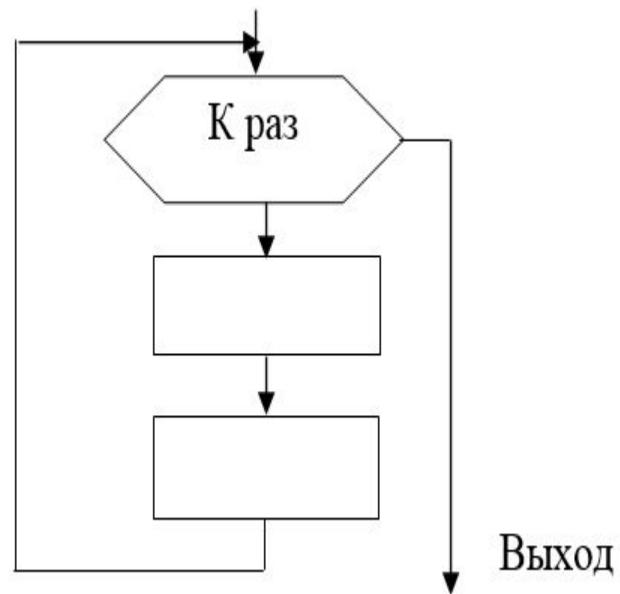
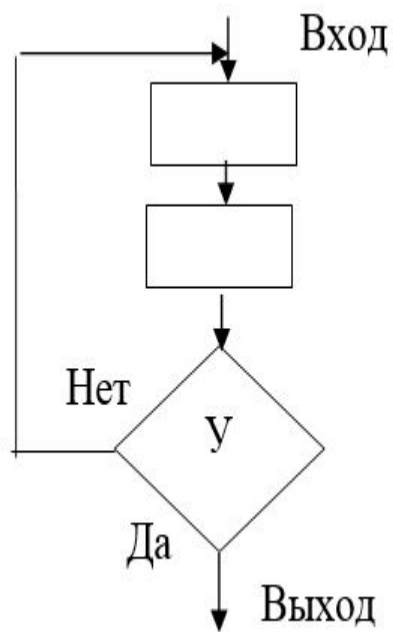
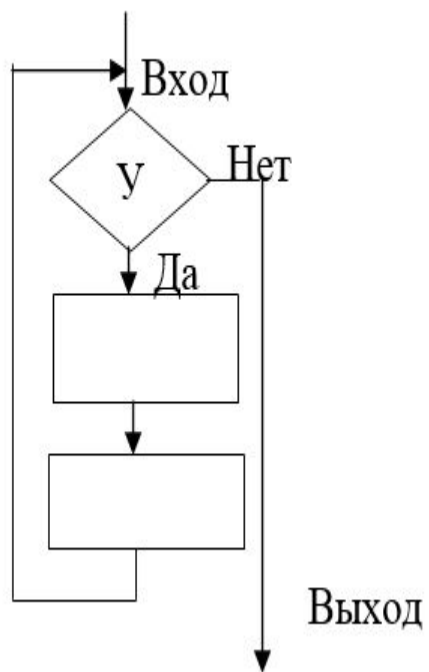


Выбор из множества:

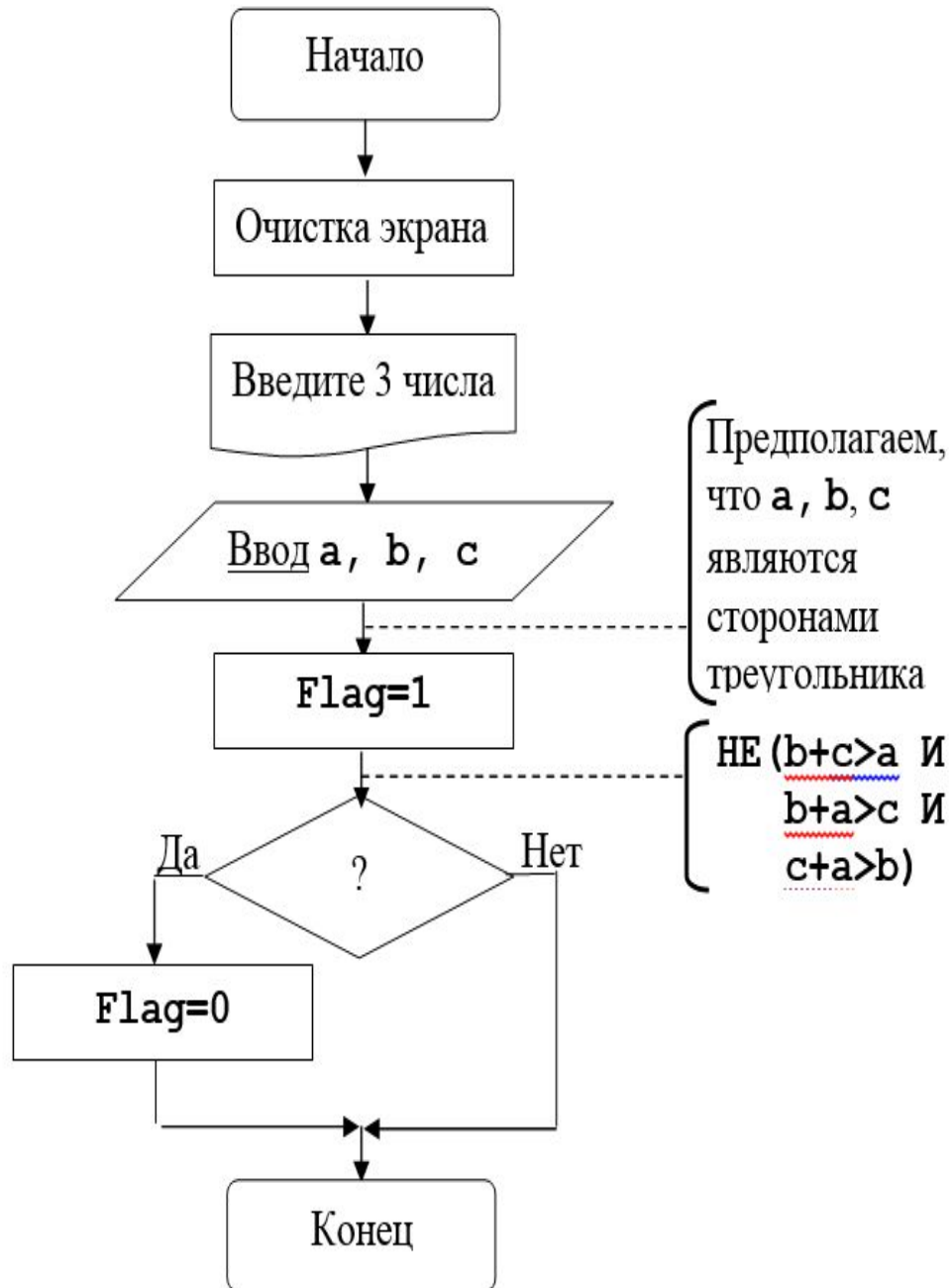


К – конкретное значение

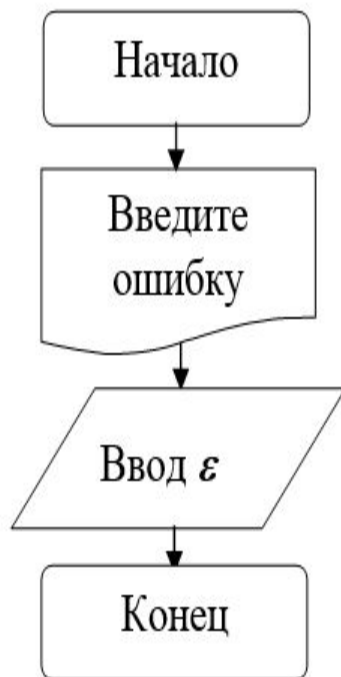
ЦИКЛЫ



Input:



Epsilon:



Компьютер – универсальный исполнитель алгоритмов

Обрабатывает все виды информации:

- Числовую
- Текстовую
- Графическую
- Звуковую

Этапы решения задач на компьютере

- **Постановка задачи** (определение требований к системе):
 1. выясняется конечная цель
 2. вырабатывается общий подход к решению задачи
 3. проверяется, являются ли исходные данные полными, т.е. содержат ли они информацию, необходимую и достаточную для решения задачи
 4. определяются тип и структура данных
 5. определяются ограничения, накладываемые на значения.
 - **Формализация** (математическая постановка).
Строится математическая модель.
- Модель** - это упрощенное представление о реальном объекте, процессе или явлении.

Этапы решения задач на компьютере

- **Выбор метода решения**
- **Разработка алгоритма**
- **Составление программы**
- **Тестирование и Отладка (тестирование)**

Тестирование – это выполнение программы с целью обнаружения факта наличия в программе ошибки.

Отладка – определение места ошибки и внесение исправлений в программу.

- **Эксплуатация программы (вычисление и обработка результатов)**

Описание языков программирования

- **Язык программирования** — набор лексических, синтаксических и семантических правил, задающих внешний вид программы и действия, которые выполнит исполнитель (компьютер) под её управлением.
- машинные;
- машинно-ориентированные (ассемблеры);
- машинно-независимые (языки высокого уровня)

СРС : выписать характеристики каждого уровня ЯП, привести примеры названия ЯП каждого уровня

Языки высокого уровня делятся на:

- процедурные (императивные);
- логические;
- объектно-ориентированные.

Транслятор - это программа, которая переводит входную программу на входном языке в эквивалентную ей выходную программу на результирующем (выходном) языке.

По способу трансляции языки делятся на:

- компиляторы
- интерпретаторы

В компиляторах перевод всего текста программы в код осуществляется сразу, и создаются исполняемый файл (обычно он имеет расширение .exe), который затем можно неоднократно запускать. Исполняемый файл может запускаться на любом компьютере, даже при отсутствии самой программы-компилятора.

В интерпретаторах при запуске программы каждая ее строка последовательно переводится в код и выполняется; затем переводится в код и выполняется другая строка, и так далее. Исполняемого файла при этом не создается. Достоинство интерпретатора — отсутствие промежуточных действий для трансляции. Это упрощает реализацию интерпретатора и делает его удобнее в использовании, в том числе в диалоговом режиме. Недостаток этого метода заключается в том, что интерпретатор должен быть в наличии на целевой машине, где должна исполняться программа.

Язык программирования

- алфавит
- синтаксис
- семантика
- прагматика

Алфавит — это фиксированный для данного языка набор основных символов, т.е. "букв алфавита", из которых должен состоять любой текст на этом языке.

Синтаксис — это правила построения фраз, позволяющие определить, правильно или неправильно написана та или иная фраза.

Семантика определяет смысловое значение предложений языка.

Прагматика определяет смысл конструкций языка.

Метаязык – позволяет описывать любые конструкции любого языка.

- **Терминальные символы** – это символы, из которых строится текст программы. Например, для языка Паскаль терминальными будут символы **begin, end, integer**.
- **Нетерминальные символы** – это символы, которые относятся к описанию языка программирования, а не к самому языку. Например, **идентификатор, двоичная цифра**.

Метасимволы БНФ:

- 1) В БНФ знак « $::=$ » (читается «это есть») – разделяет левую и правую часть правил.
- 2) Знак « $|$ » (читается «или») – разделяет возможные альтернативы в правой части правила.
- 3) Квадратные скобки « $[]$ » (читается «может быть») выделяют подцепочку, которая может присутствовать или отсутствовать в описываемой конструкции.
- 4) Фигурные скобки означают возможный повтор заключенной в них подцепочки нуль или более раз. Повтор 0 – это просто отсутствие подцепочки.

Формулы Бэкуса-Наура используют следующие обозначения:

- любое понятие берется в угловые скобки.

Например, <двоичная цифра>;

- это есть: ::=;
- 0|1; (это ноль или единица).

<двоичная цифра> ::= 0|1

- **<двоичный код> ::= <двоичная цифра> | <двоичный код> <двоичная цифра>**
- **<условный оператор> ::= if <условие> then <оператор> [else <оператор>];**
- **<идентификатор> ::= <буква> { <буква> | <цифра> }**

Синтаксические диаграммы Вирта.

На синтаксических диаграммах используются два вида четырехугольников – с прямыми и скругленными углами (иногда их заменяют кружками или овалами).

В прямоугольники заключаются элементы языка, значение которых должно быть определено (нетерминальные символы).

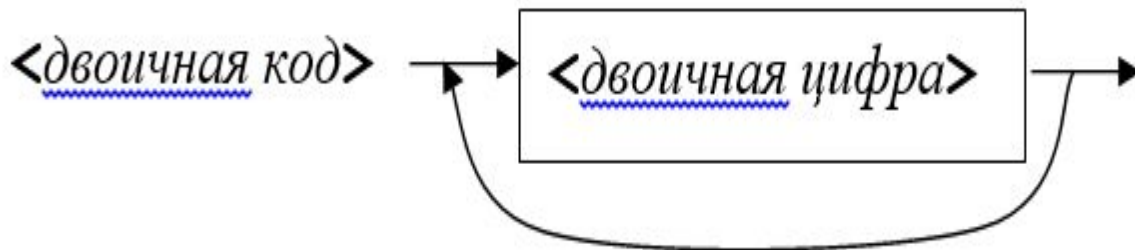
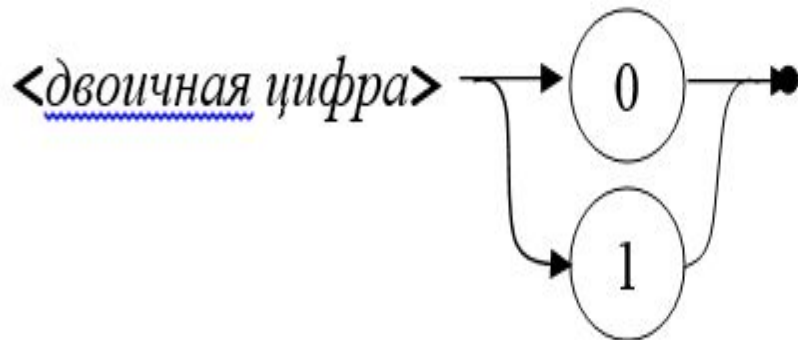
В четырехугольниках со скругленными углами (или кружках, овалах) размещаются терминальные (базовые) символы, или иероглифы языка, значение которых в определении не нуждается.

Направление движения по диаграмме при раскрытии структуры понятия, записанного при входе в диаграмму, указывают стрелки.

Примеры

$\langle \text{двоичная цифра} \rangle ::= 0|1;$

$\langle \text{двоичный код} \rangle ::= \langle \text{двоичная цифра} \rangle | \langle \text{двоичный код} \rangle;$



Принципы тестирования программ

1. Ошибки в программе есть
2. Тест – это совокупность исходных данных и ожидаемых результатов. Тест должен быть разработан заранее.
3. Тестовые данные должны быть достаточно просты для проверки.
4. Первые тесты разрабатываются после получения задания на разработку программы до написания программного кода. Цель ранней разработки тестов – уточнить постановку задачи, выявить «тонкие» места.
5. Перед началом тестирования следует сформулировать цели, которые должны быть достигнуты в ходе тестирования. В частности, набор тестов должен быть полон с точки зрения выбранных критериев полноты тестирования.

Принципы тестирования программ

6. В процессе тестирования необходимо фиксировать выполненные тесты и реально полученные результаты. Это нужно для того, чтобы уметь отделять проверенные участки в программе от непроверенных.
7. Тесты должны быть одинаково тщательны как для правильных, так и для неправильных входных данных.
8. Необходимо проверить два момента: программа делает то, что должна делать; программа не делает того, чего делать не должна.
9. Результаты необходимо изучать досконально и объяснять полностью.
10. Недопустимо ради упрощения тестирования изменять программу.

Принципы тестирования программ

11. После исправления программы необходимо повторное тестирование. «Вероятность внесения новой ошибки при исправлении старой оценивается в 20-25%. Для особо сложных систем эта вероятность может быть значительно выше.
12. Чем больше ошибок в модуле, тем больше вероятность, что там есть еще.
13. 13) Окончательное тестирование программы лучше проводить не ее автору, а другому человеку.

Критерий «черного ящика». Описывают тестирование с точки зрения поставленной задачи без учета внутреннего устройства программы.

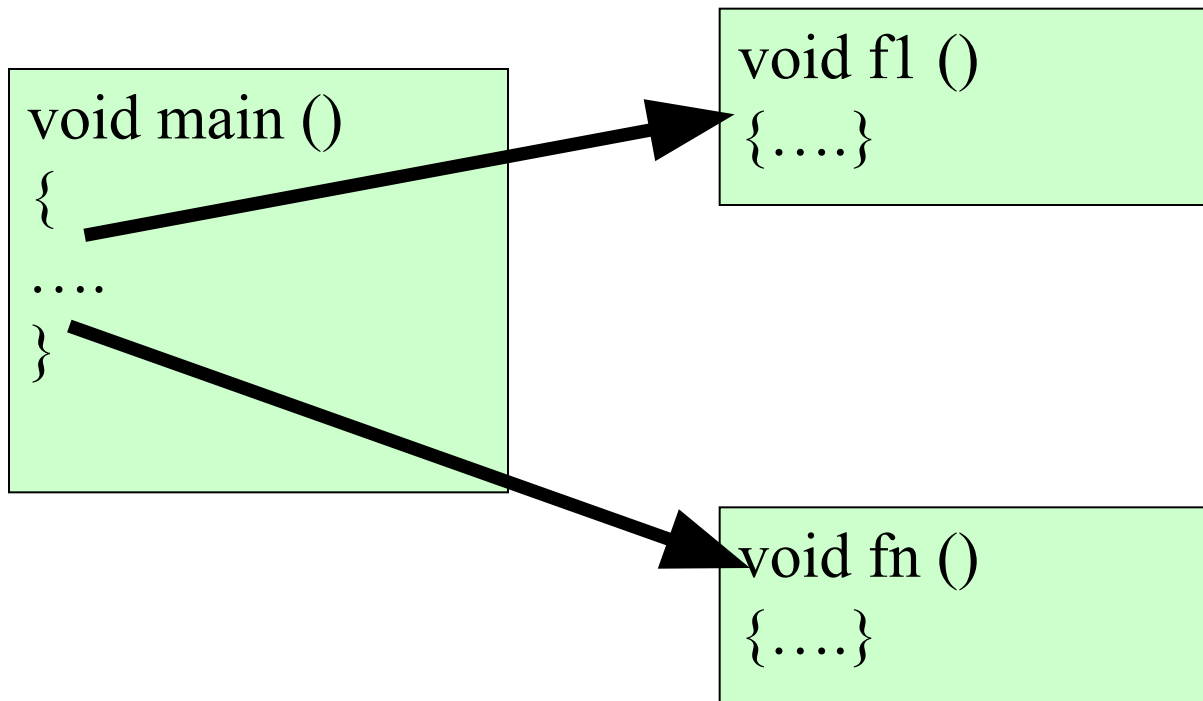
Термин «черный ящик» позаимствован из кибернетики. В кибернетике представление системы в виде черного ящика означает описание ее входов и выходов, а также связей между входными воздействиями и результатами работы системы. При этом структура системы, ее внутреннее устройство никак не затрагиваются и считаются неизвестными. В качестве примера описания систем в виде черных ящиков могут служить инструкции по использованию бытовой техники

Критерии «белого ящика». Учитывают структуру программы.

При проектировании тестов следует начинать с критериев «черного ящика». Собственно, это единственно возможный способ, если вы начинаете придумывать тесты до того, как написали программу. После подготовки текста программы тесты, разработанные исходя из критериев «черного ящика», примеряются на структуру программы. Если их оказывается недостаточно, они дополняются тестами, разработанными исходя из критериев «белого ящика».

Логическая структура программы

- Логически программа на C++ представляет собой набор функций, каждая функция должна реализовывать какое-то логически законченное действие. Функции вызываются либо из других функций, либо из главной функции с именем main().



Физическая структура программы

- Физически программа на С++ представляет собой один или несколько файлов.
- Главная функция `main()` находится в файле с расширением `.cpp` и произвольным именем.
- Другие файлы обычно содержат функции, вызываемые в `main()`, они оформляются в виде специальных заголовочных файлов и имеют расширение `.h`.

```
//файл с расширением .cpp
#include <имя_файла.h>
.....
#include <имя_файла.h>
void main()
{.....}
```

//файл с расширением .h



//файл с расширением .h

Физическая структура программы

- Физически программа на С++ представляет собой один или несколько файлов.
- Главная функция `main()` находится в файле с расширением `.cpp` и произвольными именем.
- Другие файлы обычно содержат функции, вызываемые в `main()`, они оформляются в виде специальных заголовочных файлов и имеют расширение `.h`.

```
//файл с расширением .cpp
#include <имя_файла.h>
.....
#include <имя_файла.h>
void main()
{.....}
```

//файл с расширением .h

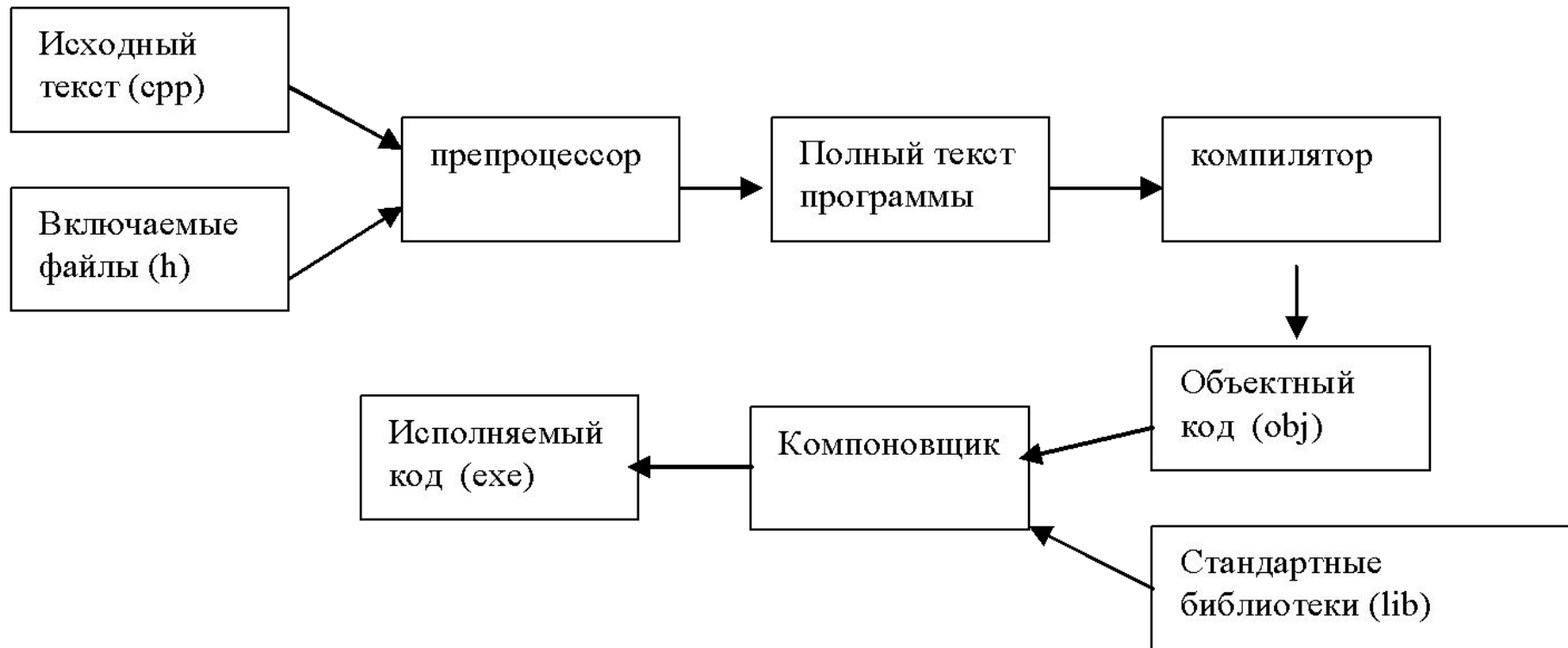


//файл с расширением .h

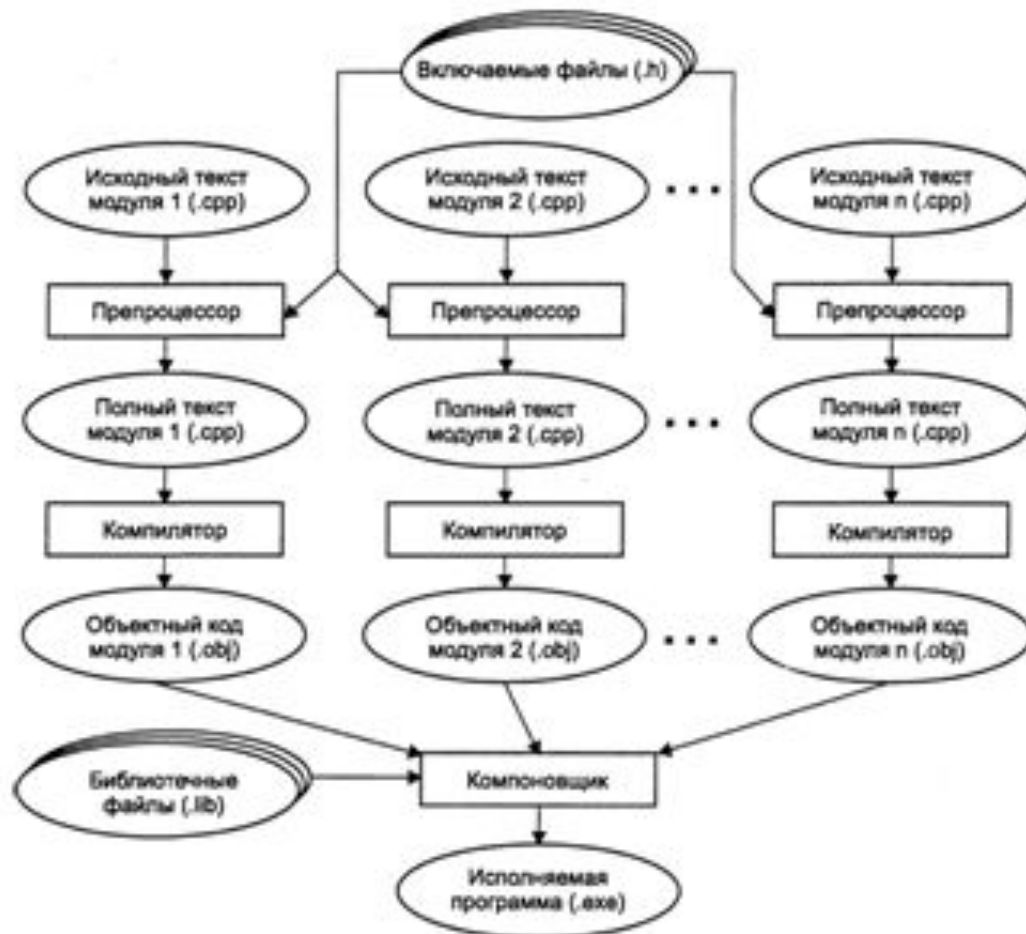
Пример программы

```
#include <iostream>
#include <cmath>
using namespace std;
void main()
{
    int a,n,i,k=0;
    double s=0;
    cout<<"\nEnter n";
    cin>>n;
    for(i=1;i<=n;i++)
        {
            cout<<"\nEnter a";
            cin>>a;
            s+=a;
            k++;
        }
    s=s/k;
    cout<<"\ns="<<s<<"\n";
}
```

Обработка C++ программы



- **Объектный модуль** (объектный файл) — файл с промежуточным представлением отдельного модуля программы, полученный в результате обработки исходного кода компилятором.
- **Компоновщик**, или редактор связей, формирует **исполняемый модуль** программы, подключая к объектному модулю другие объектные модули, в том числе содержащие функции библиотек, обращение к которым содержится в любой программе (например, для осуществления вывода на экран).



Этапы создания исполняемой программы

Директивы препроцессора

- Задача препроцессора – преобразование текста программы до ее компиляции.
- Правила препроцессорной обработки определяет программист с помощью директив препроцессора.
- Директива начинается с #.

`#define` - указывает правила замены в тексте.

```
#define ZERO 0.0
```

`#include <имя заголовочного файла>` – включает в текст программы текст из заголовочного файла, который находится в каталоге заголовочных файлов, поставляемых вместе со стандартными библиотеками.

`#include "имя заголовочного файла"` – включает в текст программы текст из заголовочного файла, который находится в текущем каталоге проекта (он может быть создан разработчиком программы) .

Основной стандартной библиотекой языка Си является библиотека `stdio.h`. Она содержит основные функции для организации ввода-вывода, для работы с файлами, а также ряд некоторых стандартных констант. В языке C++ для организации ввода-вывода используется библиотека `iostream`. Однако в C++ можно использовать также функции из стандартных библиотек языка Си.

Элементы языка C++

- Естественный язык: символы, слова, словосочетания, предложения.
- Язык программирования: символы, лексемы, выражения, операторы.

1. Алфавит языка (символы)

- прописные и строчные латинские буквы и знак подчеркивания;
- арабские цифры от 0 до 9;
- специальные знаки “{ }, |
[] () + - / % * . \ ' : ; & ? < > = ! # ^
- пробельные символы (пробел, символ табуляции, символы перехода на новую строку).

2. Лексемы языка

- *Идентификаторы* – имена объектов C/C++ -программ.
 - могут быть использованы латинские буквы, цифры и знак подчеркивания.
 - Прописные и строчные буквы различаются, например, PROG1, prog1 и Prog1 – три различных идентификатора.
 - Первым символом должна быть буква или знак подчеркивания (но не цифра).
 - Пробелы в идентификаторах не допускаются.

- **Ключевые слова** — это зарезервированные идентификаторы, которые имеют специальное значение для компилятора. Их можно использовать только в том смысле, в котором они определены. Список ключевых слов – стр.19 учебника [1].
- **Комментарии** предназначены для записи пояснений к программе и формирования документации. Компилятор комментарии игнорирует. Внутри комментария можно использовать любые символы. В C++ есть два вида комментариев: однострочные и многострочные.
- Однострочный комментарий начинается с двух символов прямой косой черты (//) и заканчивается символом перехода на новую строку, многострочный заключается между символами-скобками /* и */ и может занимать часть строки, целую строку или несколько строк.

- *Знаки операций* – это один или несколько символов, определяющих действие над операндами. Операции делятся на
 - унарные,
 - бинарные,
 - тернарную.
- *Константы* – это неизменяемые величины.
 - целые,
 - вещественные,
 - перечисляемые,
 - символьные,
 - Строковые.

Компилятор выделяет константу в качестве лексемы (элементарной конструкции) и относит ее к одному из типов по ее внешнему виду.
- *Разделители* – скобки, точка, запятая пробельные символы.

Константы

- Константа – это лексема, представляющая изображение фиксированного числового, строкового или символьного значения. Константы делятся на 5 групп:
 - целые;
 - вещественные (с плавающей точкой);
 - перечислимые;
 - символьные;
 - строковые

Целые константы

- *Целые константы* могут быть десятичными, восьмеричными и шестнадцатеричными.
- Десятичная константа определяется как последовательность десятичных цифр, начинающаяся не с 0, если это число не 0 (8, 0, 192345).
- Восьмеричная константа – это константа, которая всегда начинается с 0 (ноль). За 0 следуют восьмеричные цифры (016, 01).
- Шестнадцатеричные константы – последовательность шестнадцатеричных цифр, которым предшествуют символы 0x или 0X (0xA, 0X00F).

Вещественные константы

- Компилятор распознает вещественные константы по их виду.
- Вещественные константы могут иметь две формы представления:
 - с фиксированной точкой
 - и с плавающей точкой.
- Вид константы с фиксированной точкой:
[цифры].[цифры] (5.7, .0001, 41.).
- Вид константы с плавающей точкой:
[цифры][.][цифры]E|e[+|-][цифры] (0.5e5, .11e-5, 5E3).
- В записи вещественных констант может опускаться либо целая, либо дробная части, либо десятичная точка, либо признак экспоненты с показателем степени.

Перечислимые константы

- Перечислимые константы - это обычные целые константы, которым приспаны уникальные и удобные для использования обозначения.

```
enum {one=1, two=2, three=3, four=4};
```

```
enum {zero,one,two,three};
```

```
enum {ten=10, three=3, four, five, six};
```

```
enum {Sunday, Monday, Tuesday,  
Wednesday, Thursday, Friday, Saturday};
```

Символьные константы

- *Символьные константы* – это один или два символа, заключенные в апострофы.
- Символьные константы, состоящие из одного символа, имеют тип `char` и занимают в памяти один байт, символные константы, состоящие из двух символов, имеют целый тип и занимают два байта.

- Последовательности, начинающиеся со знака \, называются управляющими, они используются:
 - для представления символов, не имеющих графического отображения, например:
 - \a – звуковой сигнал,
 - \b – возврат на один шаг,
 - \n – перевод строки,
 - \t – горизонтальная табуляция;
 - для представления символов: \, ', ? , " (\\, \', \? , \”);
 - для представления символов с помощью шестнадцатеричных или восьмеричных кодов (\073, \0xF5);

Строковая константа

- *Строковая константа* – это последовательность символов, заключенная в кавычки.
- Внутри строк могут использоваться управляющие символы.
 - “\nНовая строка”,
 - “\n”Алгоритмические языки программирования””.

Переменные

Переменная в C++ – именованная область памяти, в которой хранятся данные определенного типа.

`int x=10`



Номер первого байта ячейки в памяти – адрес

Типы данных C/C++

- целочисленные
 - int (целый)
 - char (символьный)
 - wchar_t (расширенный символьный) (C++)
 - bool (логический) (C++)
- с плавающей точкой
 - float (вещественный)
 - double (вещественный с двойной точностью)

Спецификаторы

- short (короткий)
- long (длинный)
- signed (знаковый)
- unsigned (беззнаковый)

Тип int

- int - 4 байта (32-разрядный МП) / 2 байта(16-разрядный МП)
- short int – 2 байта, диапазон $-32768 \dots +32767$;
- long int – 4 байта, диапазон $-2\,147\,483\,648 \dots +2\,147\,483\,647$;
- unsigned short int – 2 байта, диапазон $0 \dots 65536$;
- unsigned long int – занимает 4 байта, диапазон $0 \dots +4\,294\,967\,295$.

Тип char

- char – 1 байт
- signed char, диапазон от –128 до 127.
- unsigned char, диапазон от 0 до 255.
- Для кодировки используется код ASCII (American Standard Code for International Interchange).
- Величины типа char также применяются для хранения чисел из указанных диапазонов.

Тип `wchar_t`

- Размер этого типа, как правило, соответствует типу `short`.
- Строковые константы такого типа записываются с префиксом `L`:
`L"String #1"`.

Тип bool

- true (истина)
- false (ложь)

- Внутренняя форма представления false – 0, любое другое значение интерпретируется как true.

Типы с плавающей точкой

ЧИСЛО=мантисса·10^{порядок}

$$2100=2.1 \times 10^3$$

- float - 4 байта, из которых один разряд отводится под знак мантиссы, 7 разрядов под порядок и 24 – под мантиссу.
- double - 8 байтов, под порядок и мантиссу отводятся 12 (1+11) и 52 разряда соответственно.
- long double - 10 байтов.
- Длина мантиссы определяет точность числа, а длина порядка его диапазон.

Тип void

- Множество значений этого типа – пусто.

Ввод и вывод данных в стиле C

- Для ввода/вывода данных в стиле C используются функции, которые описываются в библиотечном файле `stdio.h.(cstdio)`
- Вывод:

`printf (форматная строка, список аргументов);`

- форматная строка – строка символов, заключенных в кавычки, которая показывает, как должны быть напечатаны аргументы.

`printf ("Значение числа Пи равно %f\n", pi);`

- Форматная строка может содержать:
 - печатаемые символы;
 - спецификации преобразования;
 - управляющие символы.

- Каждому аргументу соответствует своя спецификация преобразования:
 - %d, %i – десятичное целое число;
 - %f – число с плавающей точкой;
 - %e, %E – число с плавающей точкой в экспоненциальной форме;
 - %u – десятичное число в беззнаковой форме;
 - %c – символ;
 - %s – строка.
- В форматную строку также могут входить управляющие символы:
 - \n – управляющий символ новая строка;
 - \t – табуляция;
 - \a – звуковой сигнал и др.

- Модификаторы формата– это числа, которые указывают минимальное количество позиций для вывода значения и количество позиций для вывода дробной части числа:

`%[-]m[.p]C`, где

- `-` – задает выравнивание по левому краю,
- `m` – минимальная ширина поля,
- `p` – количество цифр после запятой для чисел с плавающей точкой и минимальное количество выводимых цифр для целых чисел (если цифр в числе меньше, чем значение `p`, то выводятся начальные нули),
- `C` – спецификация формата вывода.

```
#include <stdio>
```

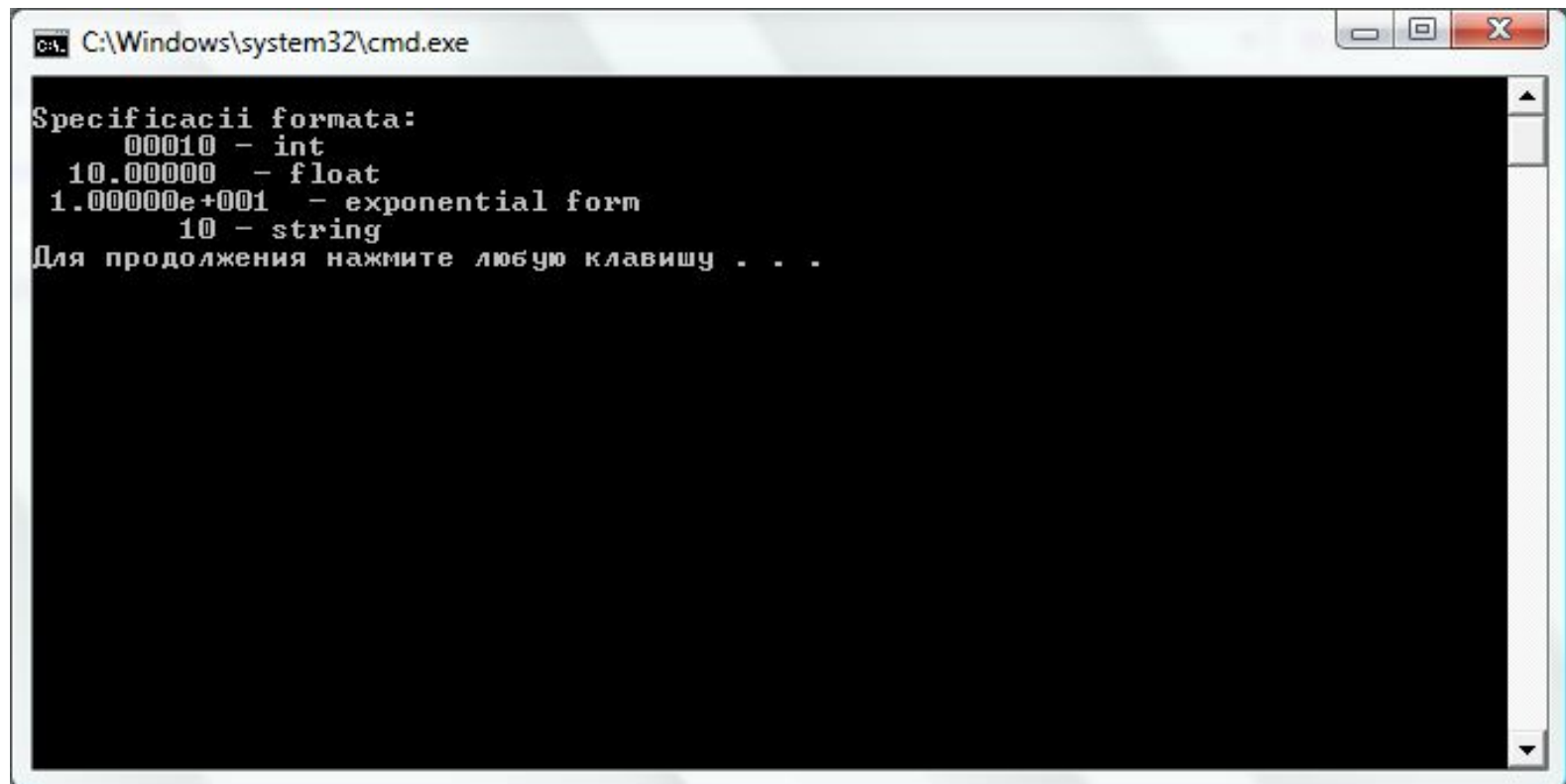
```
using namespace std;
```

```
void main()
```

```
{
```

```
printf("\nSpecificatii formata:\n%10.5d - int\n%10.5f - float\n %10.5e - exponential  
form\n%10s - string\n", 10, 10.0, 10.0, "10");
```

```
}
```



```
C:\Windows\system32\cmd.exe

Specificatii formata:
 00010 - int
 10.00000 - float
1.00000e+001 - exponential form
 10 - string
Для продолжения нажмите любую клавишу . . .
```

- Ввод:

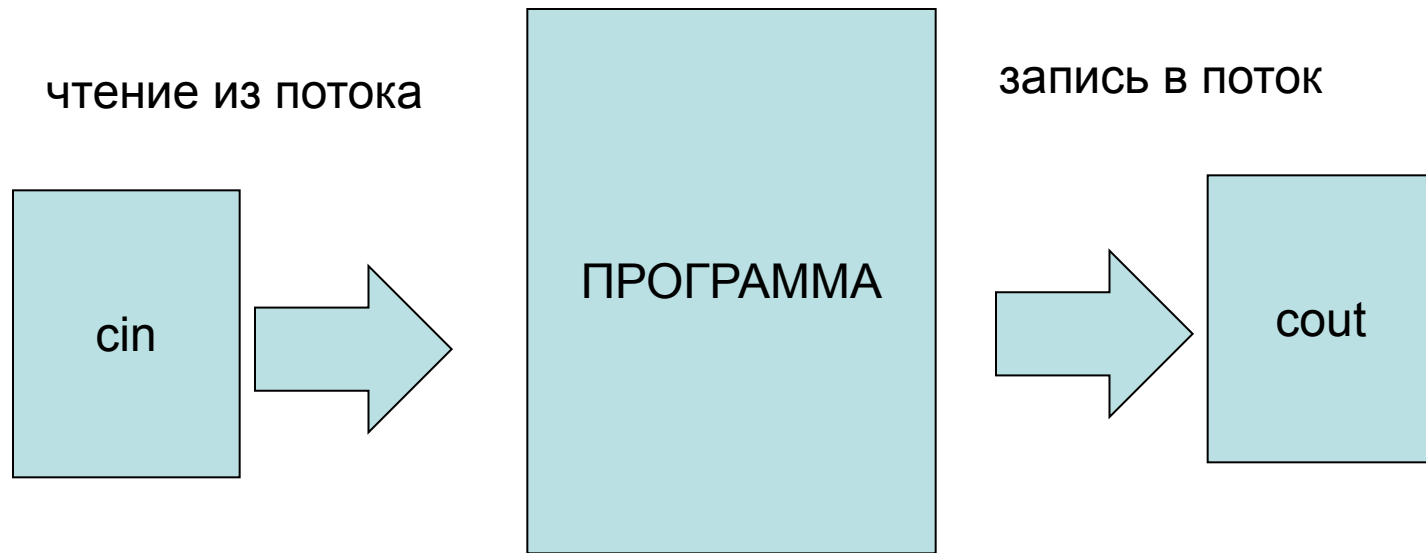
`scanf` (форматная строка, список аргументов);

- в качестве аргументов используются адреса переменных.

```
scanf(" %d%f ", &x,&y);
```

Ввод и вывод данных в стиле C++

- При использовании библиотеки классов C++, используется библиотечный файл `iostream`, в котором определены стандартные потоки ввода данных от клавиатуры `cin` и вывода данных на экран `cout`, а также соответствующие операции:
- `<<` – операция записи данных в поток;
- `>>` – операция чтения данных из потока.



```
#include <iostream>
```

```
...
```

```
cout << "\nВведите количество элементов: ";
```

```
cin >> n;
```

Библиотека `iostream.h` определяет стандартных потока:

- **`cin`** стандартный входной поток
- **`cout`** стандартный выходной поток

Для их использования в Microsoft Visual Studio обязательно необходимо прописать строку:

```
using namespace std;
```

Примеры

- Ввод значения переменной:

cin >> идентификатор;

- Возможно многократное назначение потоков:

**cin >> переменная1 >> переменная2 >>...>>
переменная n;**

- Вывод информации:

cout << значение;

- Возможно многократное назначение потоков:

**cout <<значение1 <<значение2 << ... <<
значение n;**

Программа ввода-вывода значения переменной в C++

```
#include <iostream>
using namespace std;
```

```
void main()
{
    setlocale
(LC_ALL, "rus");
```

```
int i;
cout << "Введите целое
число\n";
cin >> i;
cout << "Вы ввели
число " << i << ",
спасибо!";
}
```

- `using namespace имя_пространства_имен;`

Пространство имен - это группа имен, в которой имена не совпадают.

Эта инструкция заставляет компилятор признавать все члены одного пространства имен без дополнительного определения имени этого пространства.

- Функции `SetConsoleCP(1251)` и `SetConsoleOutputCP(1251)` позволяют использовать русский шрифт при выводе сообщений. Для их использования необходимо подключить библиотеку `windows.h`

Для использования манипуляторов с параметрами в программу необходимо включить заголовочный файл **iomanip**.

Основные стандартные манипуляторы:

- **endl** Вывод символа новой строки (переход в новую строку)
- **setprecision(int p)** Устанавливает число цифр после запятой при выводе
- **setw(int w)** Устанавливает ширину поля равной *w*
- **fixed** Вывод числа в форме с фиксированной точкой
- **scientific** Вывод числа в экспоненциальной форме

Подробнее про манипуляторы и флаги ввода-вывода можно прочитать здесь - <http://kvodo.ru/urok-10-formatirovannyiy-vvod-vyivod-v-c.html>

Операторы C++

1. Оператор «выражение»

```
i++;
```

```
a+=2;
```

```
x=a+b;
```

```
;
```

2. Составные операторы

2.1. Составные операторы

```
{  
n++;           //это составной оператор  
summa+=n;  
}
```

2.2. Блоки

```
{  
int n=0;  
n++;           //это блок  
summa+=n;  
}
```