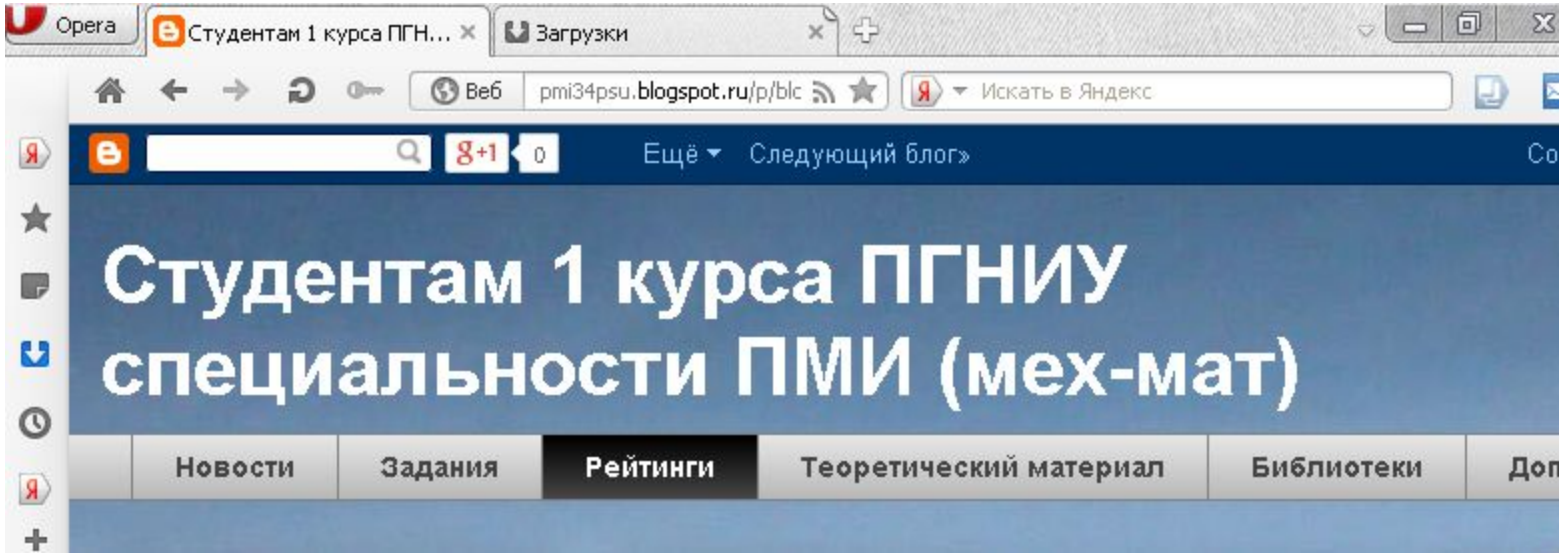
Алгоритмизация и программирование I

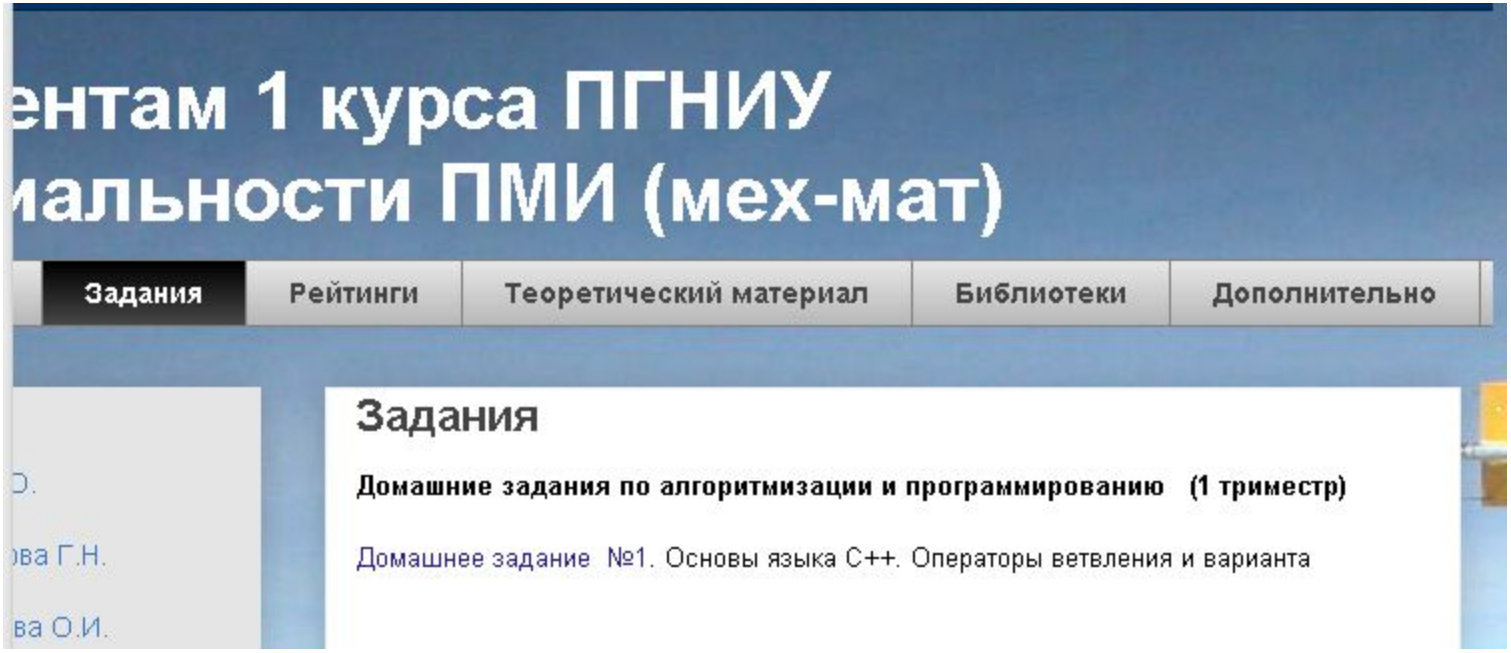
Лекция 1

Алгоритмизация и программирование I_Лекция 2

Алгоритмизация и программирование 1

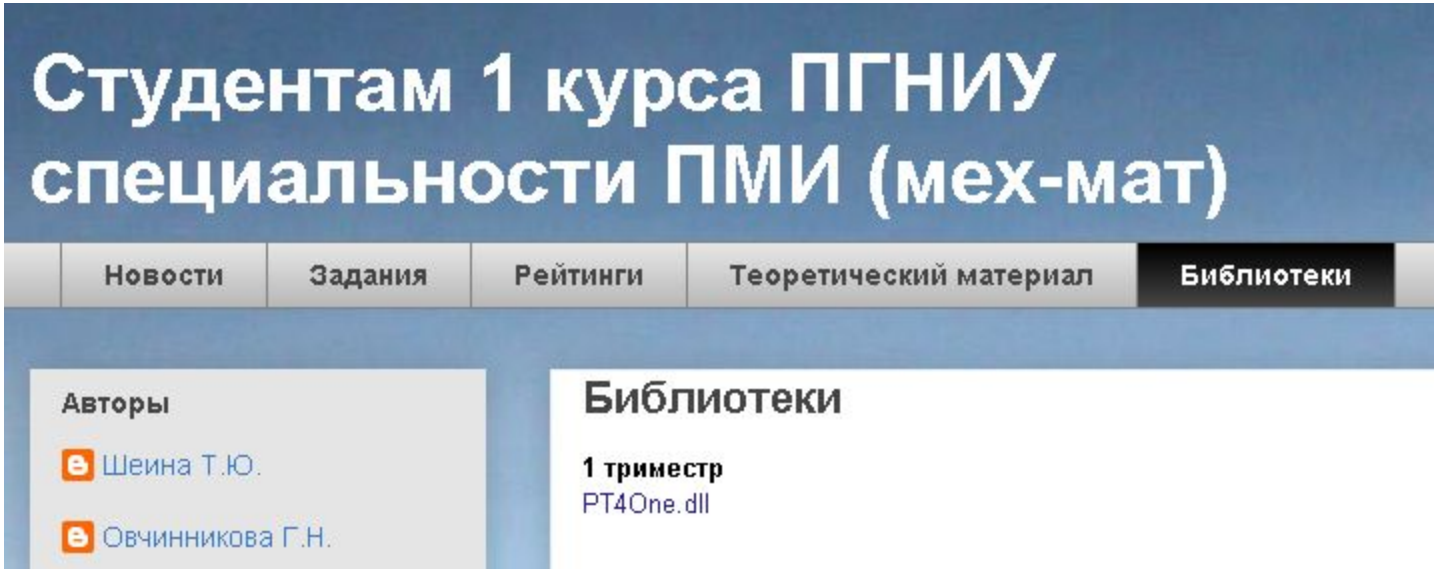
- Лекции - 28 часов
- Практика – 14 часов
- Самостоятельная работа - 64 часов
- Контрольные мероприятия - 2
- Итоговое контрольное мероприятие - экзамен


Opera
Студентам 1 курса ПГН...
Загрузки
Веб
pmi34psu.blogspot.ru/p/blc
Искать в Яндекс
+1
0
Ещё
Следующий блог»
Студентам 1 курса ПГНИУ
специальности ПМИ (мех-мат)
Новости
Задания
Рейтинги
Теоретический материал
Библиотеки


ентам 1 курса ПГНИУ
иальности ПМИ (мех-мат)
Задания
Рейтинги
Теоретический материал
Библиотеки
Дополнительно
Задания
Домашние задания по алгоритмизации и программированию (1 триместр)
Домашнее задание №1. Основы языка С++. Операторы ветвления и варианта

Адрес блога

pmi34psu.blogspot.com



Студентам 1 курса ПГНИУ специальности ПМИ (мех-мат)

Авторы

Шеина Т.Ю.

Овчинникова Г.Н.

Перескокова О.И.

Полезные ссылки

Сайт ПГНИУ

Сайт мех-мата ПГНИУ

Дополнительно

VisualC++ Express2010 (ключ регистрации 6VPJ7-H3CXH-HBTPT-X4T74-3YVY7)
Задачник (устанавливается после установки VisualC++)
Инструкция по работе с VisualC++ и задачником

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"

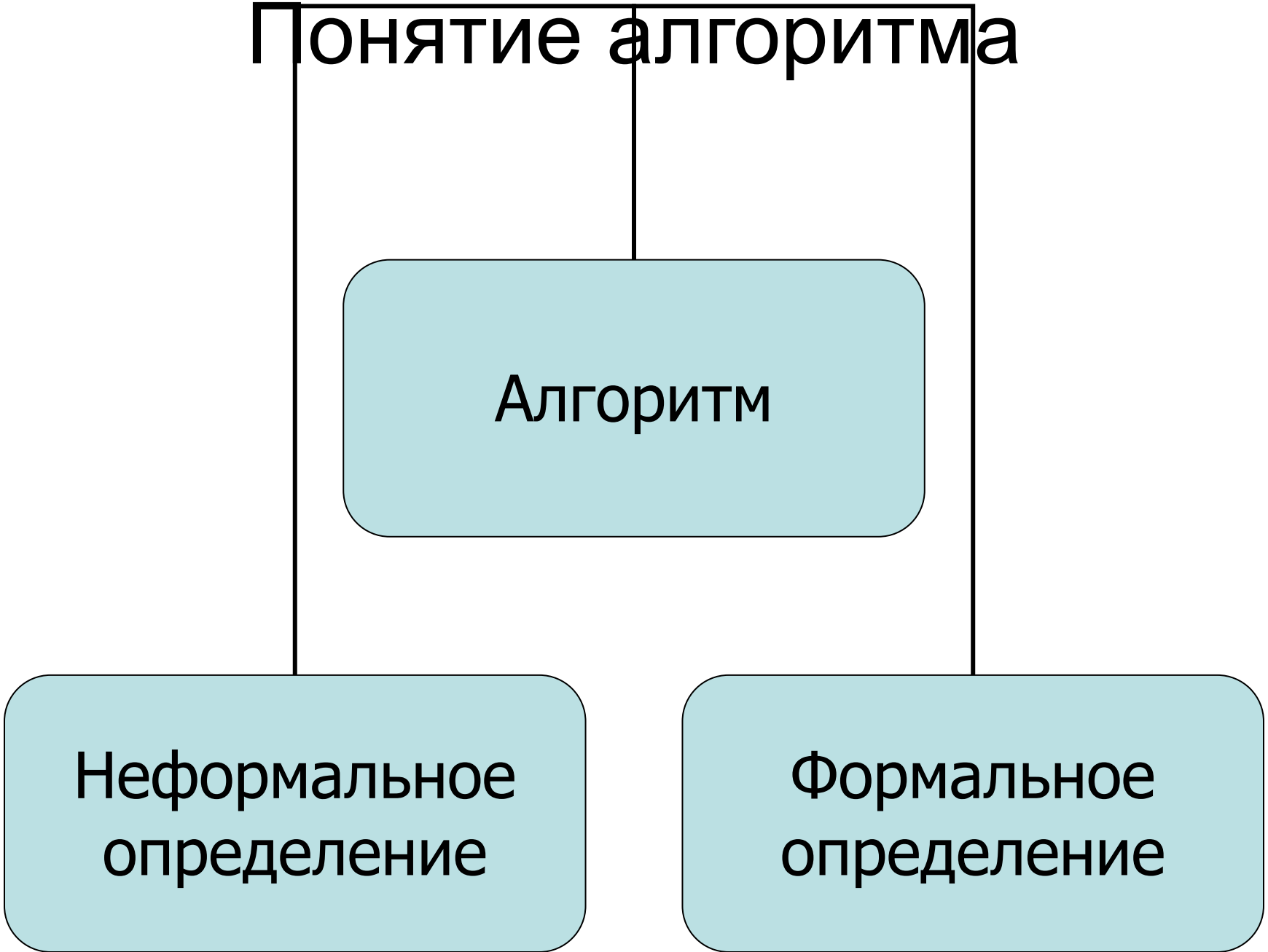
Лекция 1

- Введение в понятие алгоритма. Свойства алгоритма.
- Способы записи алгоритмов
- Язык программирования
- Способы описания языков программирования
- Этапы решения задачи с помощью ЭВМ

Введение в понятие алгоритма. Свойства алгоритма.

- Обработка информации

Понятие алгоритма



Неформальное определение алгоритма

- **Алгоритм** – это понятное и точное предписание исполнителю выполнить конечную последовательность команд, приводящую от исходных данных к искомому результату.

- **Алгоритм** – точное предписание, которое задает вычислительный процесс, начинающийся с произвольного исходного данного (из некоторой совокупности возможных для данного алгоритма исходных данных) и направленный на получение полностью определяемого этим исходным данным результата.

Математическая энциклопедия, 1977

Исполнитель алгоритма

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

Свойства алгоритма

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

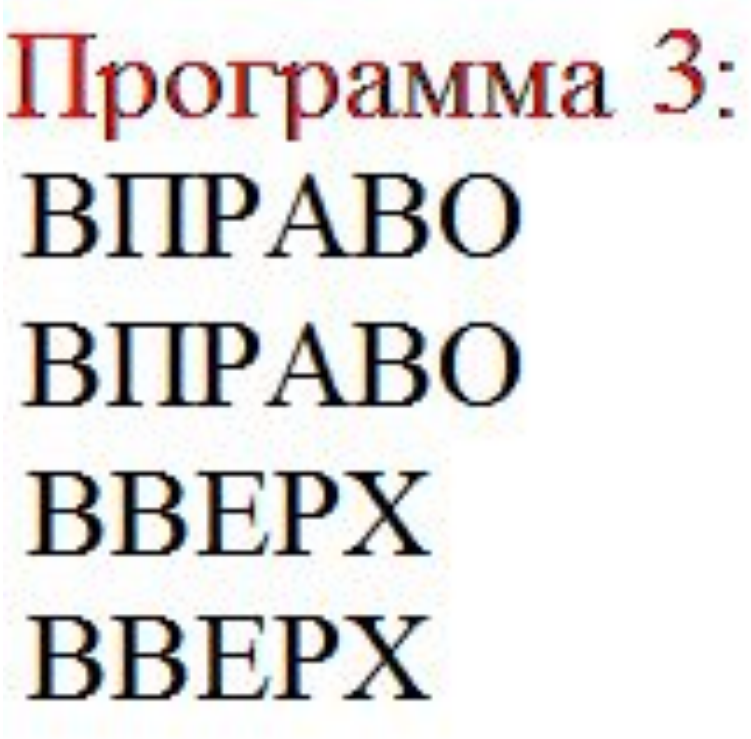
Понятность

Алгоритм должен быть записан на языке, понятном исполнителю.

СКИ:

ВВЕРХ

ВПРАВО



Дискретность

- Алгоритм состоит из конечного числа инструкций и все инструкции выполняются в дискретном времени.

- Инструкции выполняются мгновенно в моменты времени t_0, t_1, t_2, … , и между этими моментами ничего не происходит.

Элементарность шагов

- Объем работы выполняемый на всяком шаге ограничен сверху некоторой константой, не зависящей от объема данных.

Элементарные шаги	НЕ элементарны
- сложение; - вычитание; - умножение; - деление; - сравнение чисел …	- сравнение двух файлов; - проверка жесткого диска на вирусы; - архивирование папки …

Детерминированность (определенность, точность)

- Для каждого шага по набору исходных данных результат выполнения шага определяется однозначно и не зависит ни от каких случайных факторов.
- Тогда и итоговый результат всего алгоритма тоже будет однозначно определен.

Конечность (финитность)

- Выполнение алгоритма должно завершиться за конечное число шагов.
- Число шагов может быть очень большим, но оно не может быть равно ∞.

Массовость (универсальность)

- Алгоритм должен быть применим к разным наборам допустимых исходных данных.

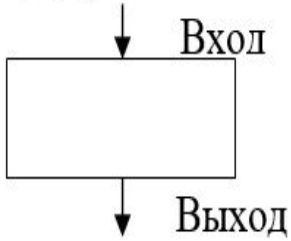
- Алгоритм, выходные данные которого уникальны в силу свойства детерминированности будет давать всегда один и тот же результат. => Построение такого алгоритма теряет смысл.

Способы записи алгоритмов

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

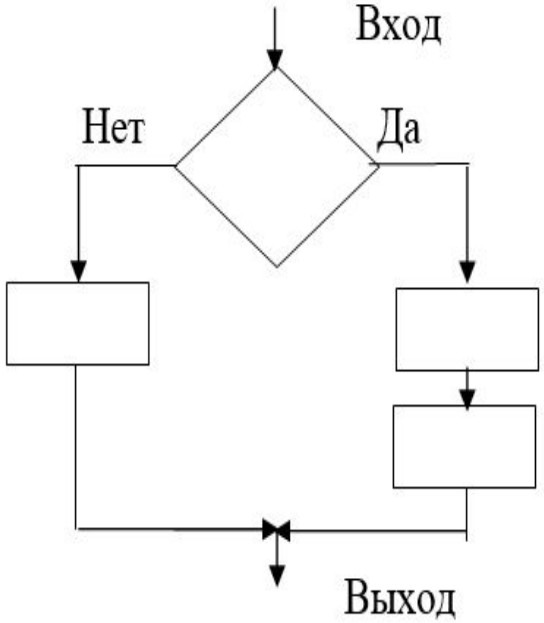
Основные управляющие структуры

1) Простое следование:

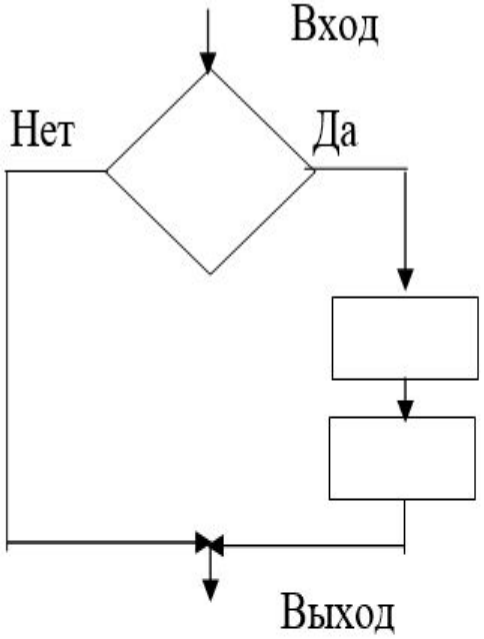


2) Альтернатива:

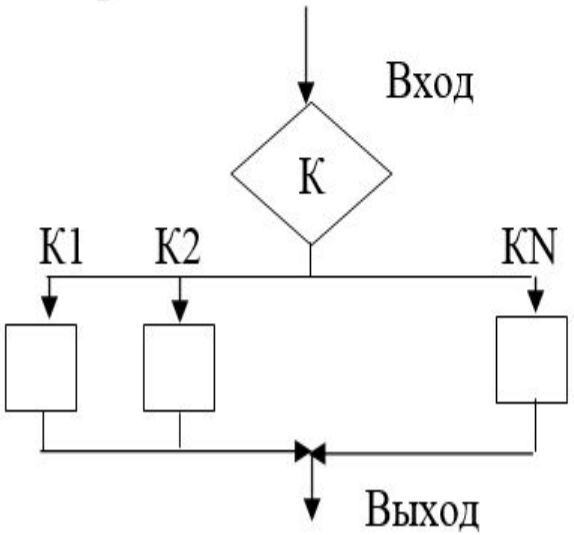
Полный выбор из двух вариантов:

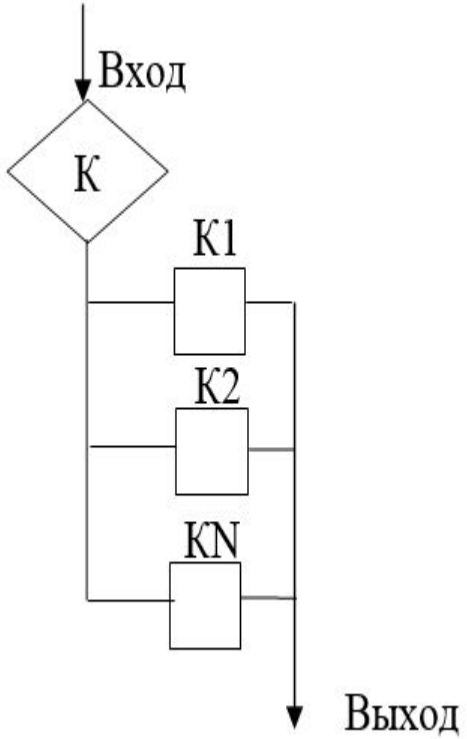


Неполное ветвление:



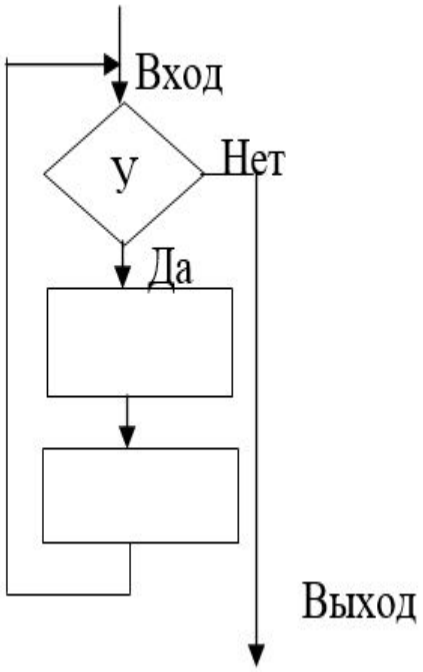
Выбор из множества:

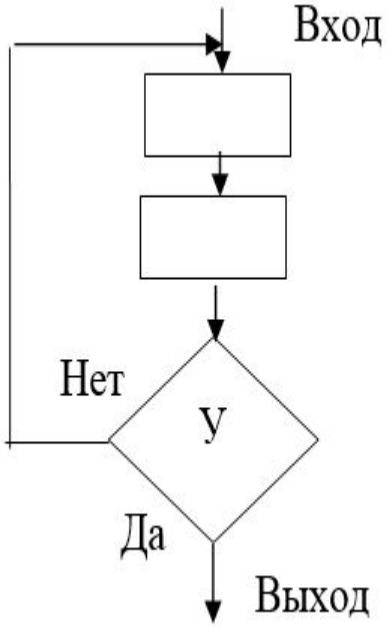


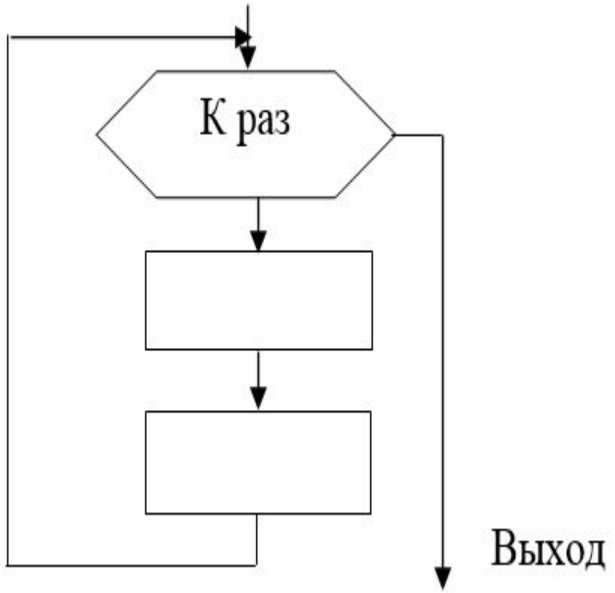


K – конкретное значение

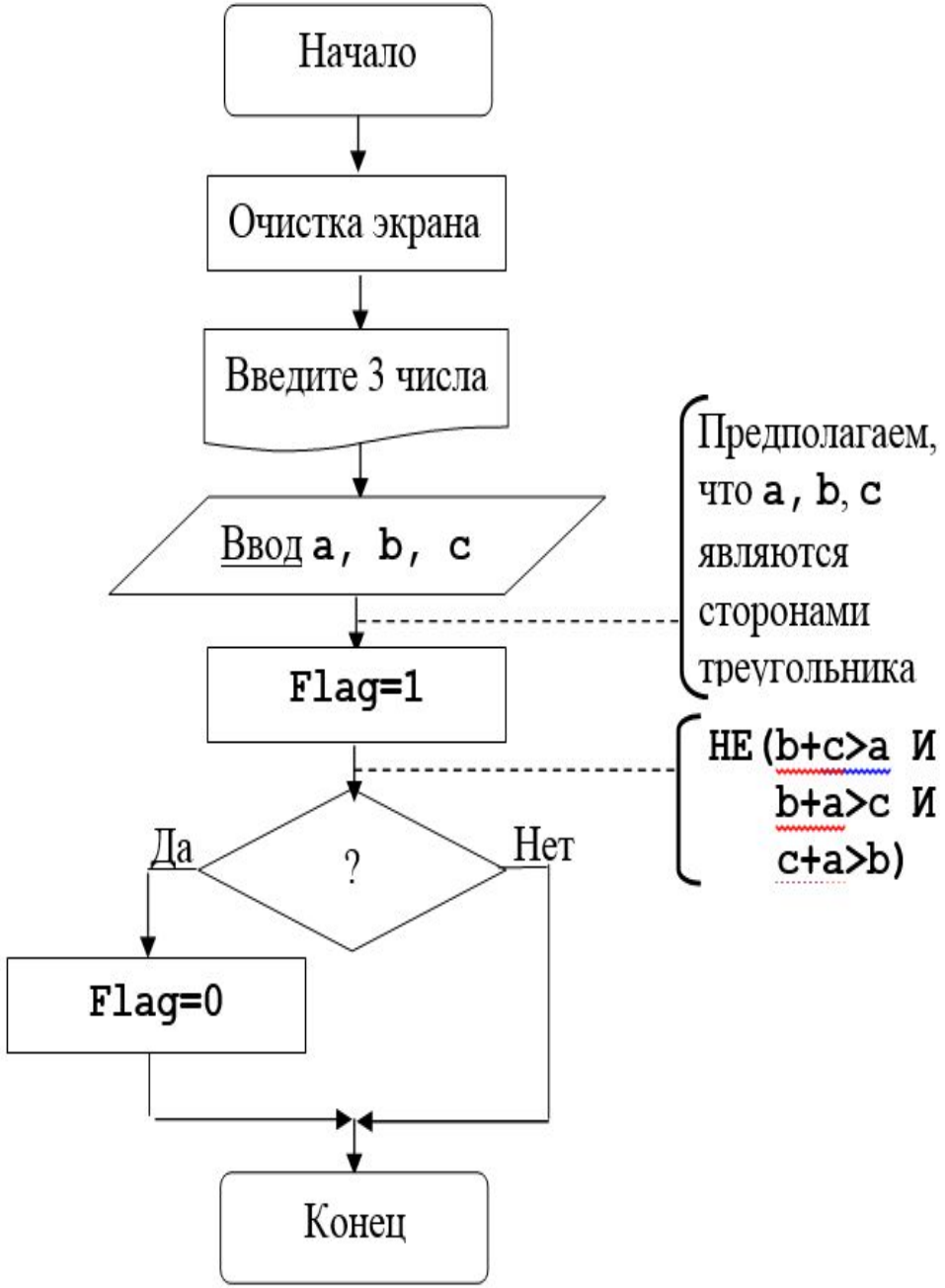
ЦИКЛЫ



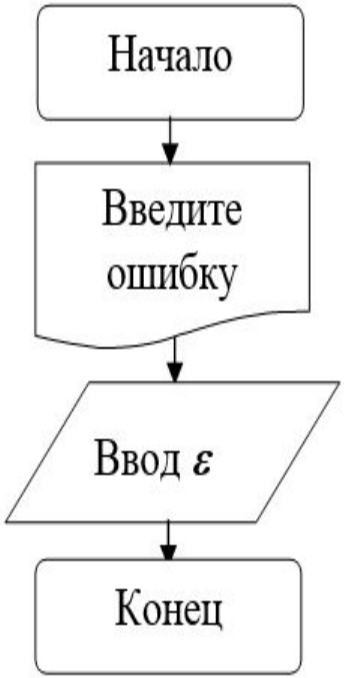




Input:


Начало
Очистка экрана
Введите 3 числа
Ввод a, b, c
Предполагаем, что a, b, c являются сторонами треугольника
Flag=1
НЕ (b+c>a И b+a>c И c+a>b)
Да
?
Нет
Flag=0
Конец

Epsilon:

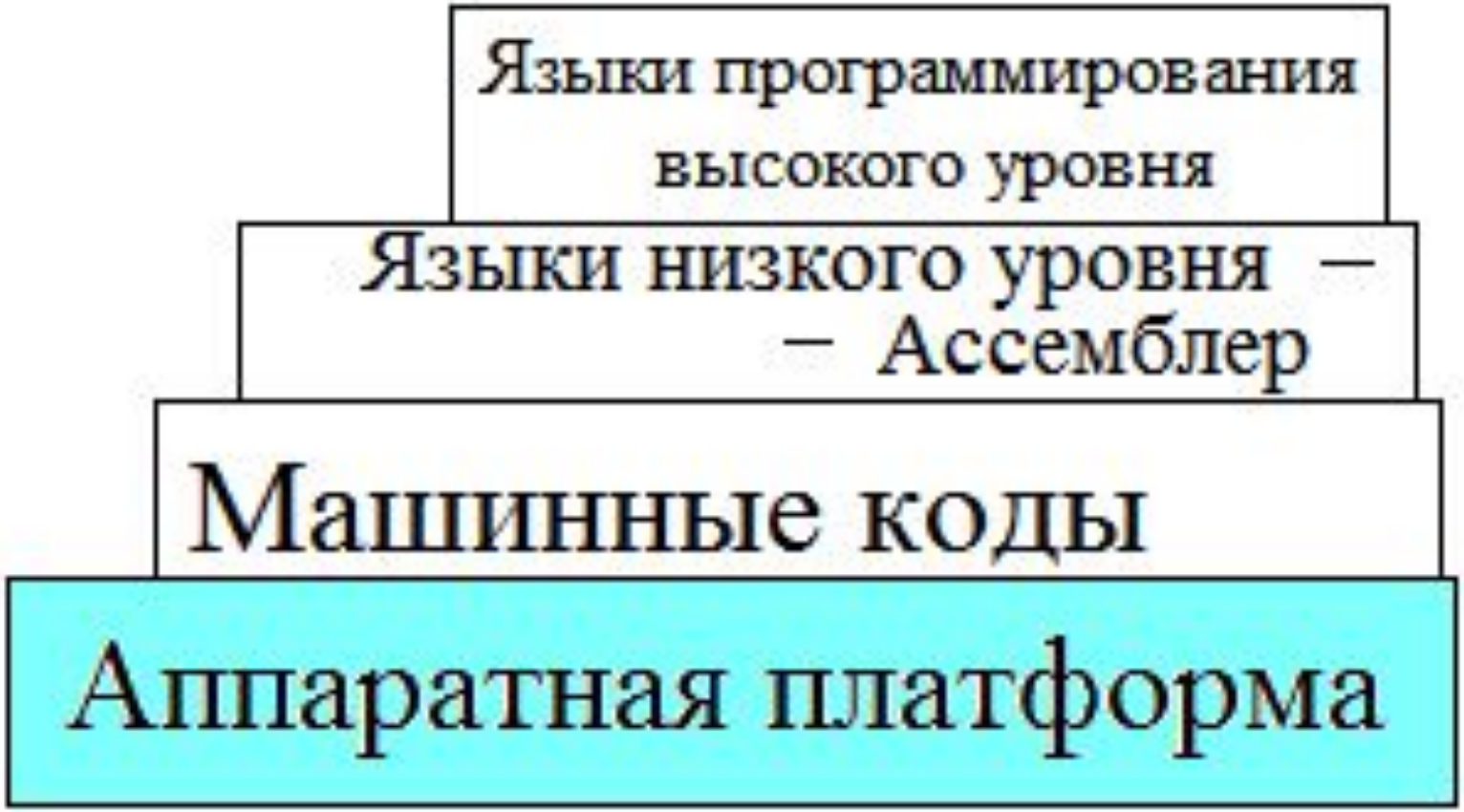


Язык программирования

ПРОГРАММА – это алгоритм, записанный на определенном языке программирования.

ЯЗЫК ПРОГРАММИРОВАНИЯ – это формальная знаковая система, предназначенная для записи компьютерных программ.

Какие бывают ЯП?



ЯПВУ

Языки высокого уровня делятся на:

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

Транслятор

ТРАНСЛЯТОР – это программа, которая переводит программу с языка высокого уровня на язык машинных команд.

По способу трансляции различают:

- компиляторы;
- интерпретаторы.

Компиляторы и интерпретаторы

Компилятор	Интерпретатор
В код переводится весь текст программы целиком	В код переводятся отдельные строки программы и сразу выполняются
Создается исполняемый файл, который впоследствии можно запускать даже при отсутствии компилятора	Исполняемый файл (*.exe) не создается
	В отсутствии интерпретатора программа не запустится

Основные компоненты языка программирования

- **Описание лексики** – задание алфавита языка.
- **Описание синтаксиса** – задание правил построения конструкций ЯП.
- **Описание семантики** – придание смысла конструкциям языка.
- **Описание прагматики** – отвечает на вопрос: «Как писать программы на этом языке?»

Способы описания языков программирования

Используются метаязыки, т.е. посредством которых можно описать другой язык:

- **Нотация Бэкуса-Наура**
- **Синтаксические диаграммы Вирта**

Терминальные символы – это элементы алфавита языка, из них строится текст программы.

Нетерминальные символы – это понятия, которые требуют дальнейшей расшифровки, пока не превратятся в терминальные.

Бэкуса-Наура форма (БНФ)

Нетерминальные символы заключаются в угловые скобки (< >).

Метасимволы БНФ:

::=	“определяется как”	Разделяет левую и правую часть правила.
|	“или”	Разделяет альтернативы.
[ ]	“может быть”	Цепочка, записанная внутри скобок может отсутствовать.
{ }	“может быть ноль или более раз”	Цепочка может повторяться многократно или может отсутствовать.

Примеры БНФ

<двоичная цифра>::= 0 | 1

<двоичный код>::=

<двоичная цифра>{<двоичная цифра>}

<условный оператор>::=

if (<условие>) <оператор>

[else <оператор>]

Диаграммы Вирта

Терминальные символы располагаются внутри кругов или прямоугольников со скругленными углами

Диаграммы Вирта

- Нетерминальные символы заключаются в прямоугольники

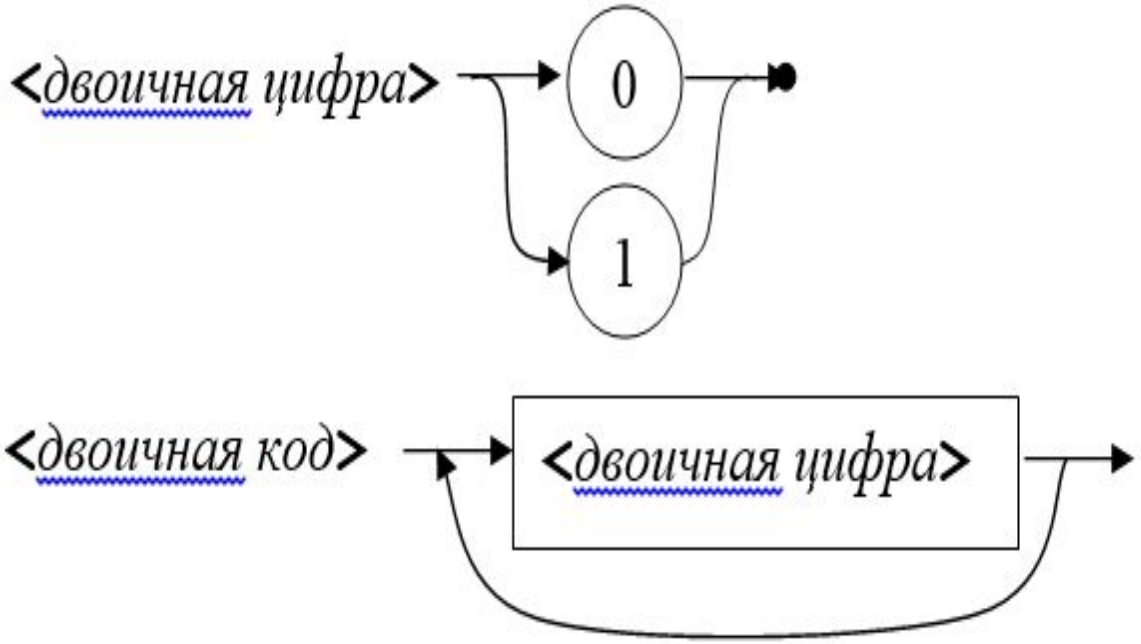
Диаграммы Вирта

- В начале диаграммы указывается расшифровываемое понятие.
- Ветвления и циклы показываются стрелками и изгибами линий.

Примеры

<*двоичная цифра*>::=0|1;

<*двоичный код*>:::=<*двоичная цифра*>|<*двоичный код*>;



Данные

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

Этапы решения задачи с помощью ЭВМ

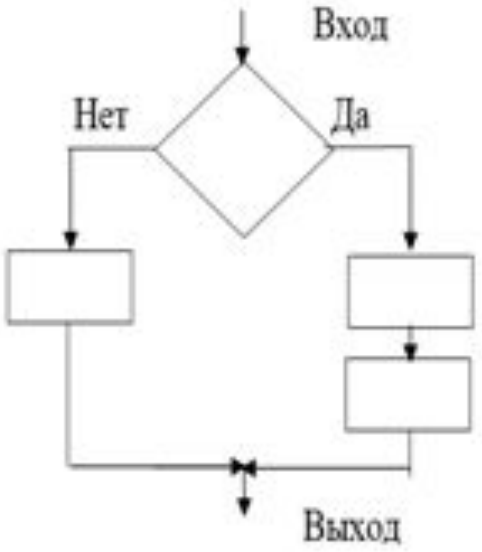
1. Постановка задачи (определение требований к системе)
2. Анализ и проектирование (построение формальных моделей, определение структур данных, выбор методов решения)
3. Разработка (кодирование)
4. Тестирование
5. Развертывание и сопровождение

Алгоритмизация и программирование I

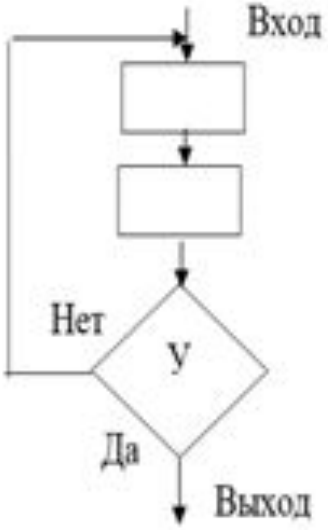
Лекция 2

Как называются эти управляющие структуры?

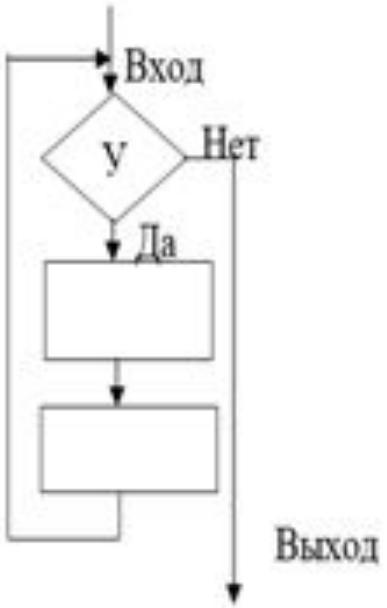
1)



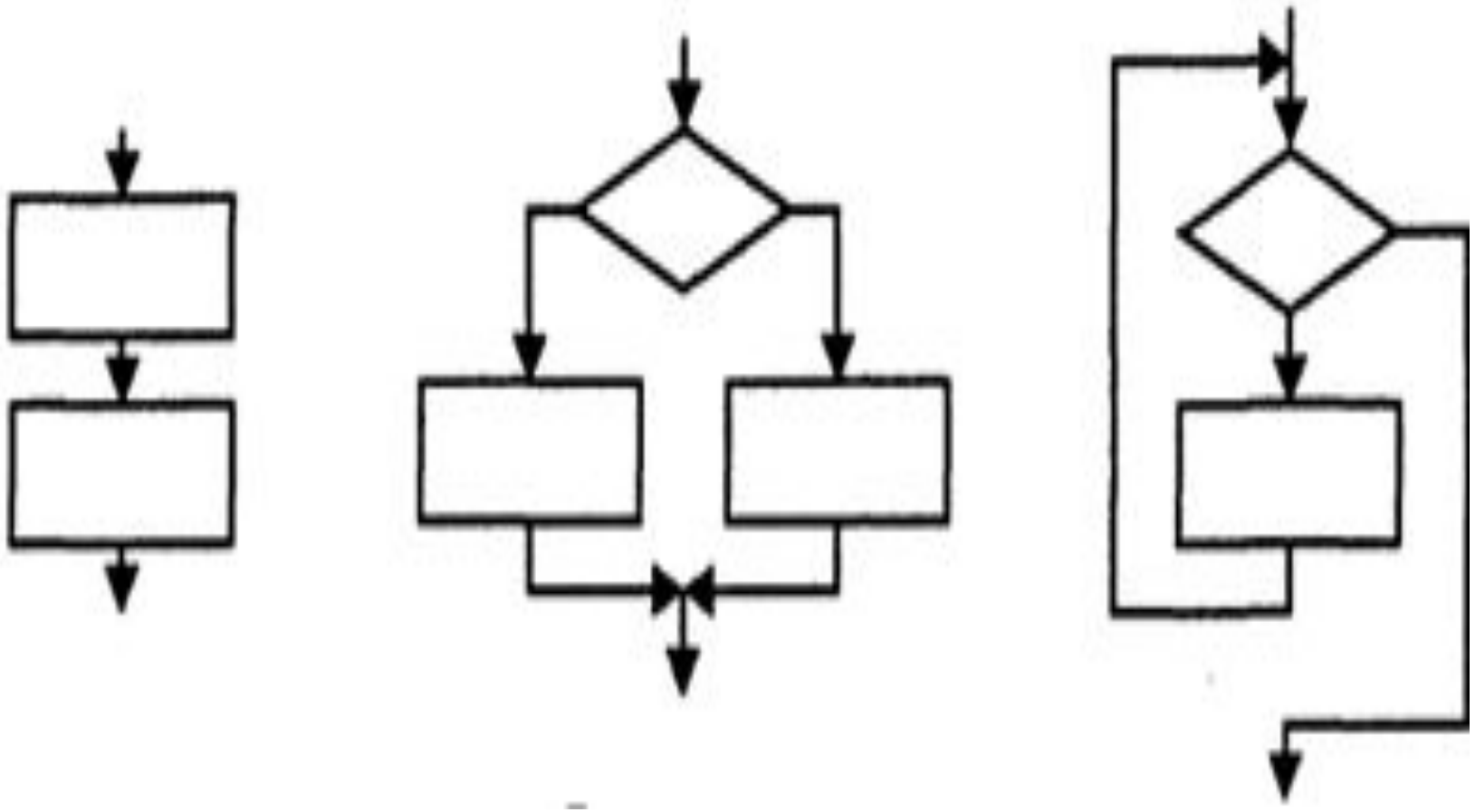
2)



3)



Как называются эти управляющие структуры?



ОТВЕТ

1)Следование

1) Полное ветвление

2) Цикл с предусловием

ОТВЕТ

1)Полное ветвление

2)Цикл с постусловием

3)Цикл с предусловием

- Этап тестирования
- C++
- Типы данных C/C++
- Переменные
- Логическая структура программы
- Ввод и вывод данных в стиле C

Этап тестирования

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

Отладка – определение места ошибки и внесение исправлений в программу.

Принципы тестирования

- Ошибки в программе есть.
- **Тест** – это совокупность исходных данных и ожидаемых результатов.
- Тестовые данные должны быть достаточно просты для проверки.
- Тесты готовятся заранее, до выхода на машину.
- Первые тесты разрабатываются после получения задания на разработку программы до написания программного кода.

Принципы тестирования

- Перед началом тестирования следует сформулировать цели, которые должны быть достигнуты в ходе тестирования.
- В процессе тестирования необходимо фиксировать выполненные тесты и реально полученные результаты.
- Тесты должны быть одинаково тщательны как для правильных, так и для неправильных входных данных.
- Необходимо проверить два момента: программа делает то, что должна делать; программа не делает того, чего делать не должна.

Принципы тестирования

- Результаты теста необходимо изучать досконально и объяснять полностью.
- Недопустимо ради упрощения тестирования изменять программу.
- После исправления программы необходимо повторное тестирование.
- Ошибки «кучкуются».
- Окончательное тестирование программы лучше проводить не ее автору, а другому человеку.

Способы тестирования

- ***Тестирование по принципу «черного ящика»*** описывают тестирование с точки зрения поставленной задачи без учета внутреннего устройства программы.
- ***Тестирование по принципу «белого ящика»*** учитывают структуру программы.

Тестирование по принципу «черного ящика»

- тестирование функций;
- тестирование классов входных данных;
- тестирование классов выходных данных.

Тестирование границ класса

1) нормальные условия

2) граничные (экстремальные) условия

3) исключительные условия (выход за границу класса).

Немного истории

C => C++ => C#

Деннис Ритчи

Бьёрн Страуструп

группа инженеров под руководством Андерса Хейлсберга

Общие правила

- Большие и маленькие буквы различаются (main, Main, MAIN, mAin – разные имена)

- После каждого оператора ставится точка с запятой “;”

- Комментарии бывают многострочные

```
/* Этот комментарий может состоять
     из нескольких строк
 */
```

- и однострочные

```
// вся оставшаяся часть строки - комментарий
```

Алфавит языка

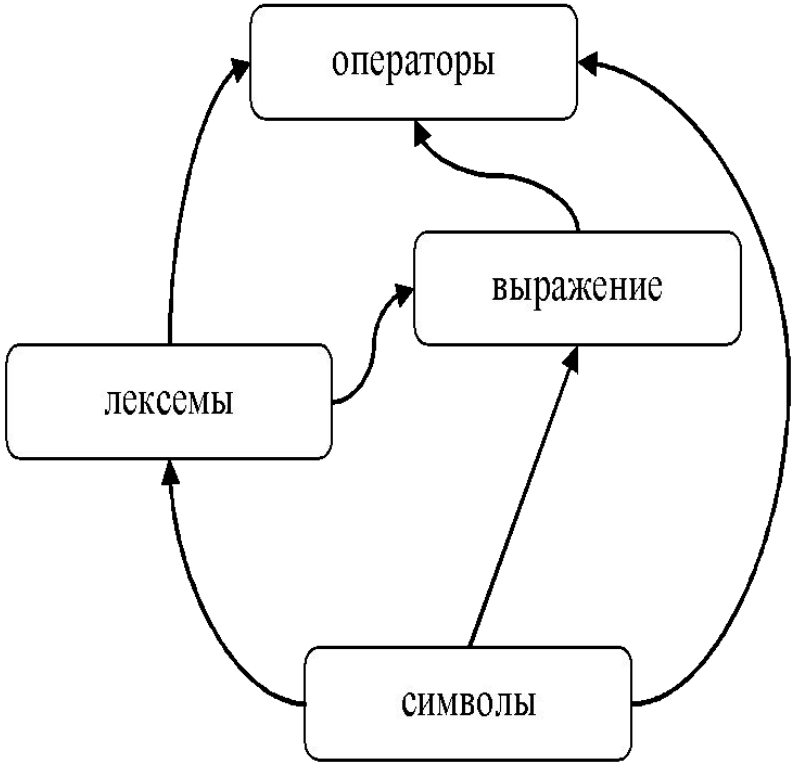
- прописные и строчные латинские буквы и знак подчеркивания;
- арабские цифры от 0 до 9;
- специальные знаки:

  “ { } , | [ ] ( ) + - / % * . \

  ‘ : ? < = > ! & # ~ ; ^
- пробельные символы:

  пробел, символы табуляции,

  символы перехода на новую строку.


операторы
выражение
лексемы
символы

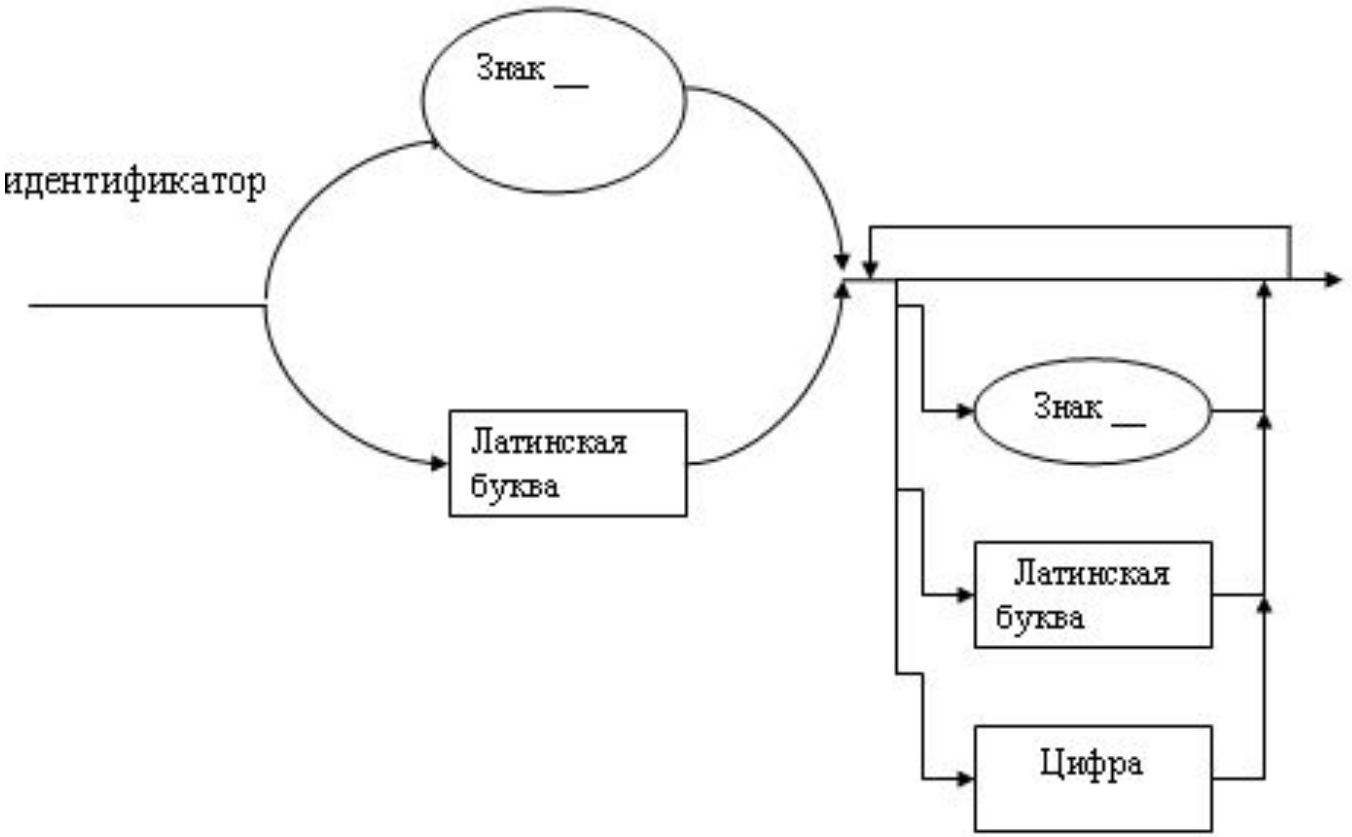
Лексемы языка

- идентификаторы;
- ключевые (зарезервированные) слова;
- знаки операций;
- константы;
- разделители (скобки, точка, запятая, пробельные символы)

Ключевые слова языка

asm	else	new	this
auto	enum	operator	throw
bool	explicit	private	true
break	export	protected	try
case	extern	public	typedef
catch	false	register	typeid
char	float	reinterpret_cast	typename
class	for	return	union
const	friend	short	unsigned
const_cast	goto	signed	using
continue	if	sizeof	virtual
default	inline	static	void
delete	int	static_cast	volatile
do	long	struct	wchar_t
double	mutable	switch	while
dynamic_cast	namespace	template	

<идентификатор>::= _| <латинская буква>{<цифра>|_ |<латинская буква>}



Идентификатор

- Идентификатор – это имя программного объекта.
- При записи идентификатора допустимы:

  латинские буквы, цифры,

  знак подчеркивания (_) `A_B_C_1_`
- Первым символом идентификатора цифра быть не может.
- Идентификатор не может совпадать с зарезервированным словом.

Константа	Формат	Примеры
Целая	Десятичный: последовательность десятичных цифр, начинающаяся не с нуля, если это не число нуль	8, 0, 199226
	Восьмеричный: нуль, за которым следуют восьмеричные цифры (0,1,2,3,4,5,6,7)	01, 020, 07155
	Шестнадцатеричный: 0x или 0X, за которым следуют шестнадцатеричные цифры (0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F)	0xA, 0x1B8, 0X00FF
Вещественная	Десятичный: [цифры].[цифры]	5.7, .001, 35.
	Экспоненциальный: [цифры][.][цифры]{E|e}[+|−][цифры]	0.2E6, .11e−3, 5E10
Символьная	Один или два символа, заключенных в апострофы	'A', 'ю', '*', 'db', '\0', '\n', '\012', '\x07\x07'
Строковая	Последовательность символов, заключенная в кавычки	"Здесь был Vasia", "\tЗначение r=\0xF5\n"

Управляющие последовательности

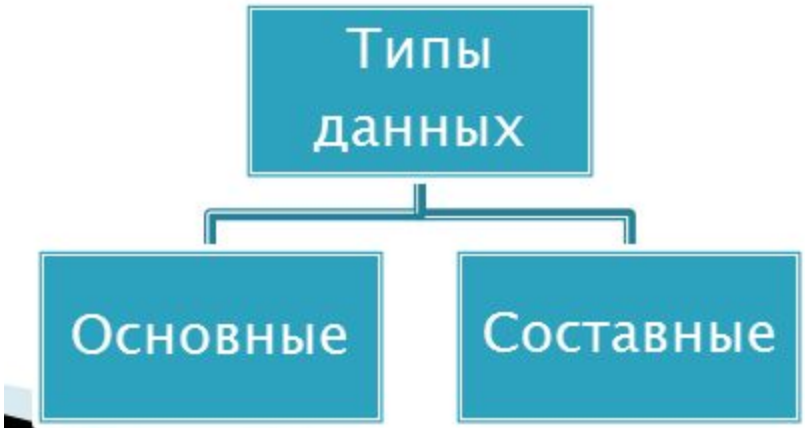
Изображение	Шестнадцатеричный код	Наименование
\a	7	Звуковой сигнал
\b	8	Возврат на шаг
\f	C	Перевод страницы (формата)
\n	A	Перевод строки
\r	D	Возврат каретки
\t	9	Горизонтальная табуляция
\v	B	Вертикальная табуляция
\\	5C	Обратная косая черта
\'	27	Апостроф
\"	22	Кавычка
\?	3F	Вопросительный знак
\0ddd	—	Восьмеричный код символа
\0xddd	ddd	Шестнадцатеричный код символа

```
"Встречаемся у \"баобаба\" в 13:10"
```

Типы данных

Тип данных определяет:

- внутреннее представление данных в памяти компьютера;
- множество значений, которые могут принимать величины этого типа;
- операции и функции, которые можно применять к величинам этого типа.



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 типа

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

Диапазоны значений типов

Тип	Диапазон значений	Размер (байт)
bool	true и false	1
signed char	–128 ... 127	1
unsigned char	0 ... 255	1
signed short int	–32 768 ... 32 767	2
unsigned short int	0 ... 65 535	2
signed long int	–2 147 483 648 ... 2 147 483 647	4
unsigned long int	0 ... 4 294 967 295	4
float	3.4e–38 ... 3.4e+38	4
double	1.7e–308 ... 1.7e+308	8
long double	3.4e–4932 ... 3.4e+4932	10

Соотношение диапазонов

- В стандарте ANSI диапазоны значений для основных типов не задаются, определяются только соотношения между их размерами:

sizeof(float)≤sizeof(double)≤sizeof(long double)

sizeof(char)≤sizeof(short)≤sizeof(int)≤sizeof(long)

Тип int

- Размер типа int не определяется стандартом, а зависит от компьютера и компилятора.
- Для 16-разрядного процессора под величины этого типа отводится 2 байта, а для 32-х разрядного – 4 байта

Использование спецификаторов

- По умолчанию все целочисленные являются знаковыми, т.е. спецификатор **`signed`** можно опускать
- **`short int`** = **`short`**
- **`long int`** = **`long`**
- **`signed int`** = **`signed`**
- **`unsigned int`** = **`unsigned`**
- Для констант:
  - суффиксы **`U,u`** обозначают **`unsigned`**
  - суффиксы **`L,l`** обозначают **`long`**

```
036uL
32L
12LU
0x2FFFu
```

Тип char

- Используется для представления символов из 256-х символьного набора ASCII.

Кроме того, используется для хранения целых чисел, укладывающихся в границы типа.

- `int i1 = 0x01FF;`

```
unsigned char c;
c = i1;
i1 = c;
```

Типы с плавающей точкой

- float
- double
- long double

- Константы с плавающей точкой по умолчанию имею тип double.

- Можно явно указать тип константы с помощью суффиксов:
  - f,F (float),
  - l,L(long double)

```
1.21f
5E-20L
```

Тип bool

- Величины логического типа могут принимать только значения true и false.
- Внутренняя форма представления:

  false – 0 (нуль)

  true – 1 (единица).
- При преобразовании к логическому типу 0 трактуется как false, а любое ненулевое значение как true.

```
if (a*b)  c=10; else c=-10;
```

Тип void

- Множество значений этого типа пусто
- Используется для определения функций, которые не возвращают значение и для указания пустого списка аргументов функции.

Описание переменной

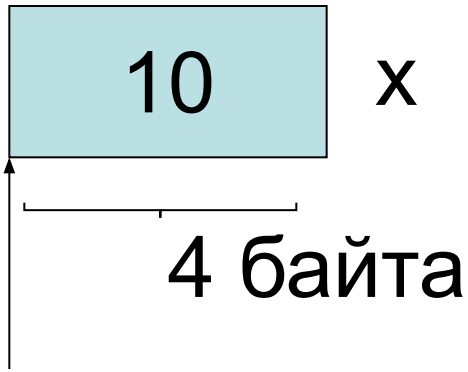
[класс памяти] [const] тип имя [инициализатор]

```
short x,t;
int y=0;   int y(0);
bool flag=true;
const float pi=3.1415926;
double z(2.17),r(.5),p(1E7);
char ch, a='0',s(67);
```

Переменные

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

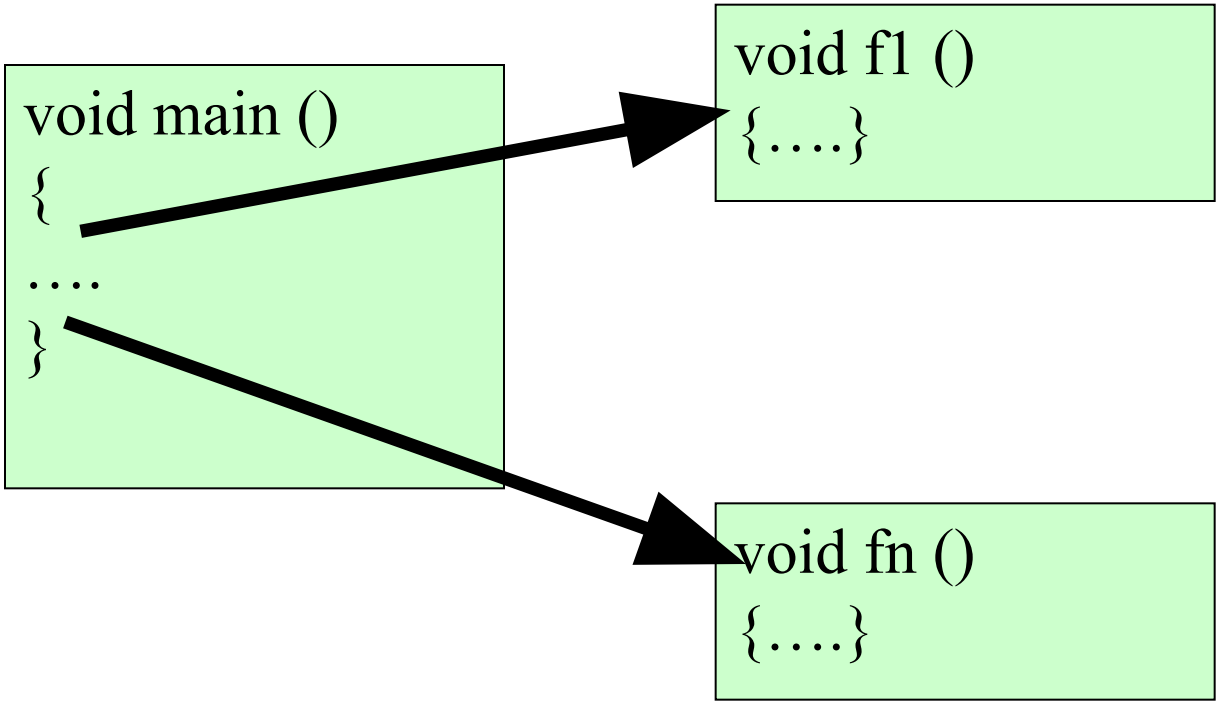
int x=10;



Номер первого байта ячейки в памяти – адрес

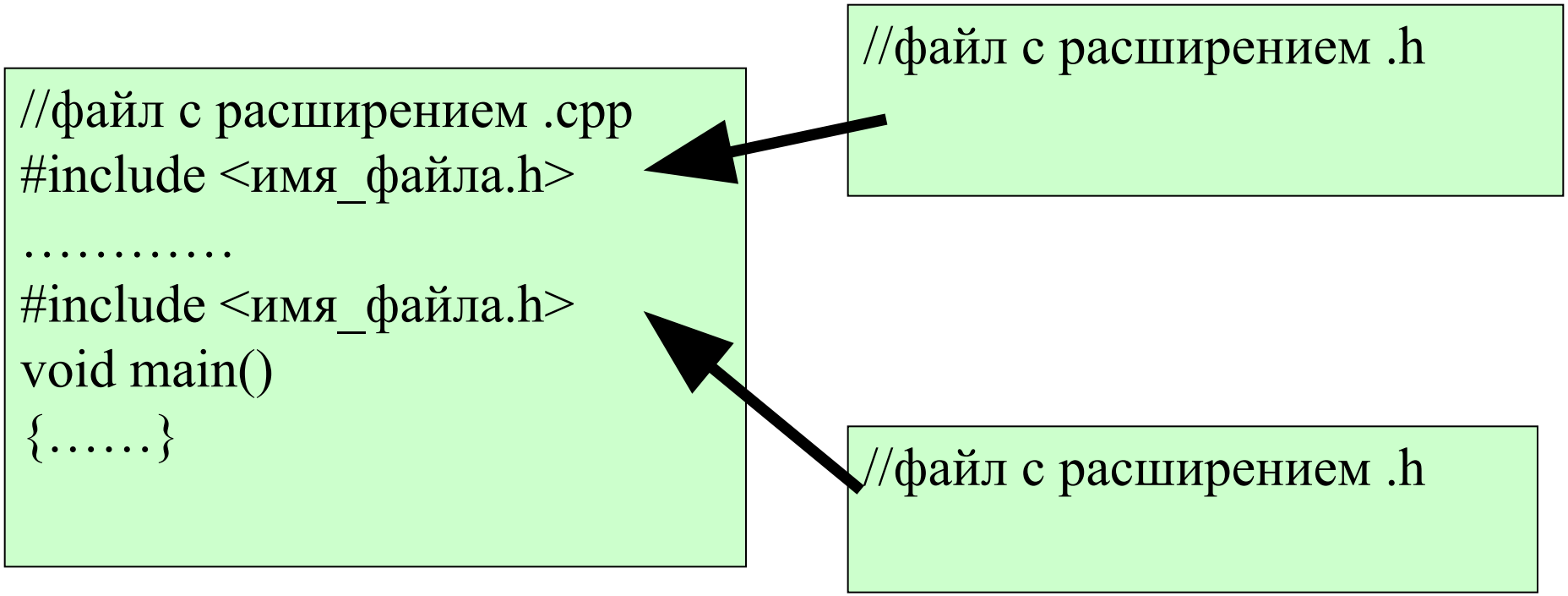
Логическая структура программы

- 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()**.

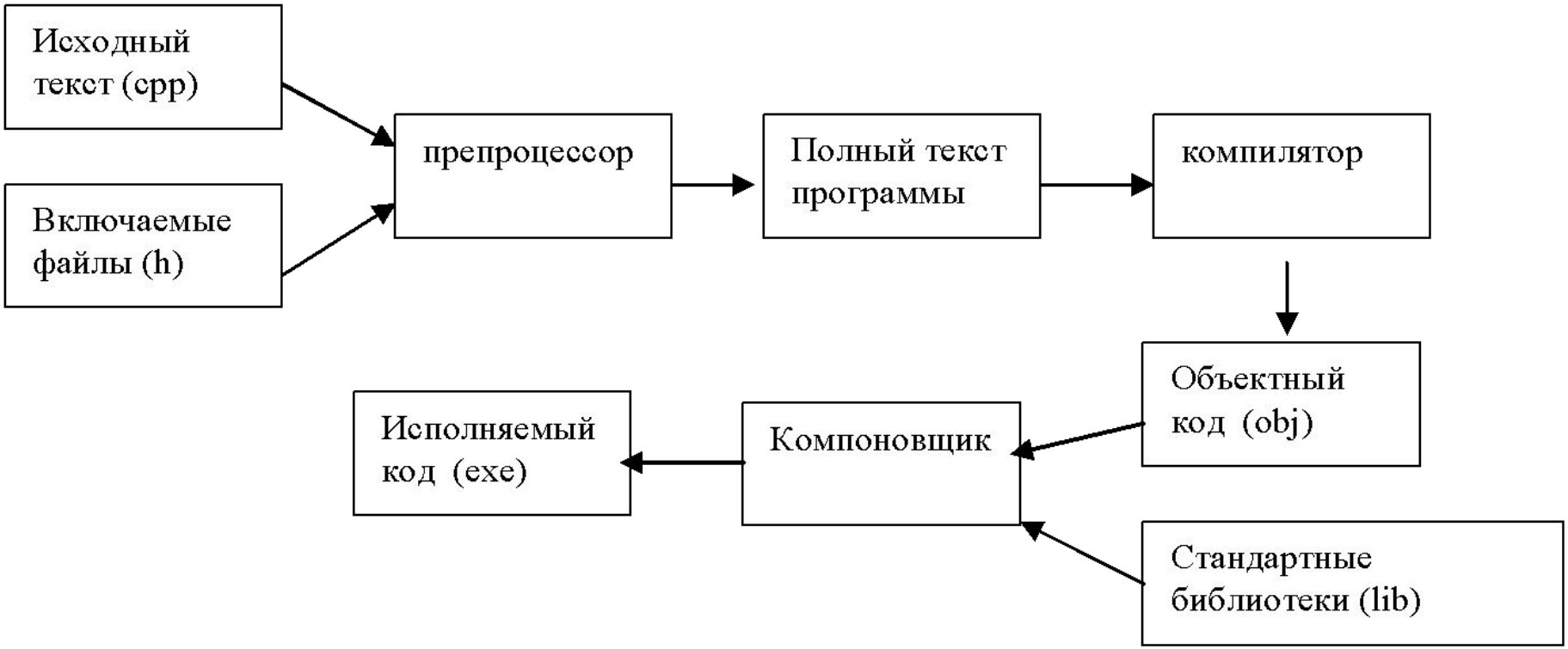


Физическая структура программы

- Физически программа на C++ 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**.



Обработка C++ 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**>
- 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**>.
- В C++ можно использовать также функции из стандартных библиотек языка Си.

Ввод-вывод в C++

	I способ	II способ
	Унаследованный от C	На основе потоковых классов
Подключаемая библиотека	#include <stdio.h>	#include <iostream>
Ввод	scanf(…)	cin >> …
Вывод	printf(…)	cout << …

Ввод и вывод данных в стиле С

- Для ввода/вывода данных в стиле С 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.

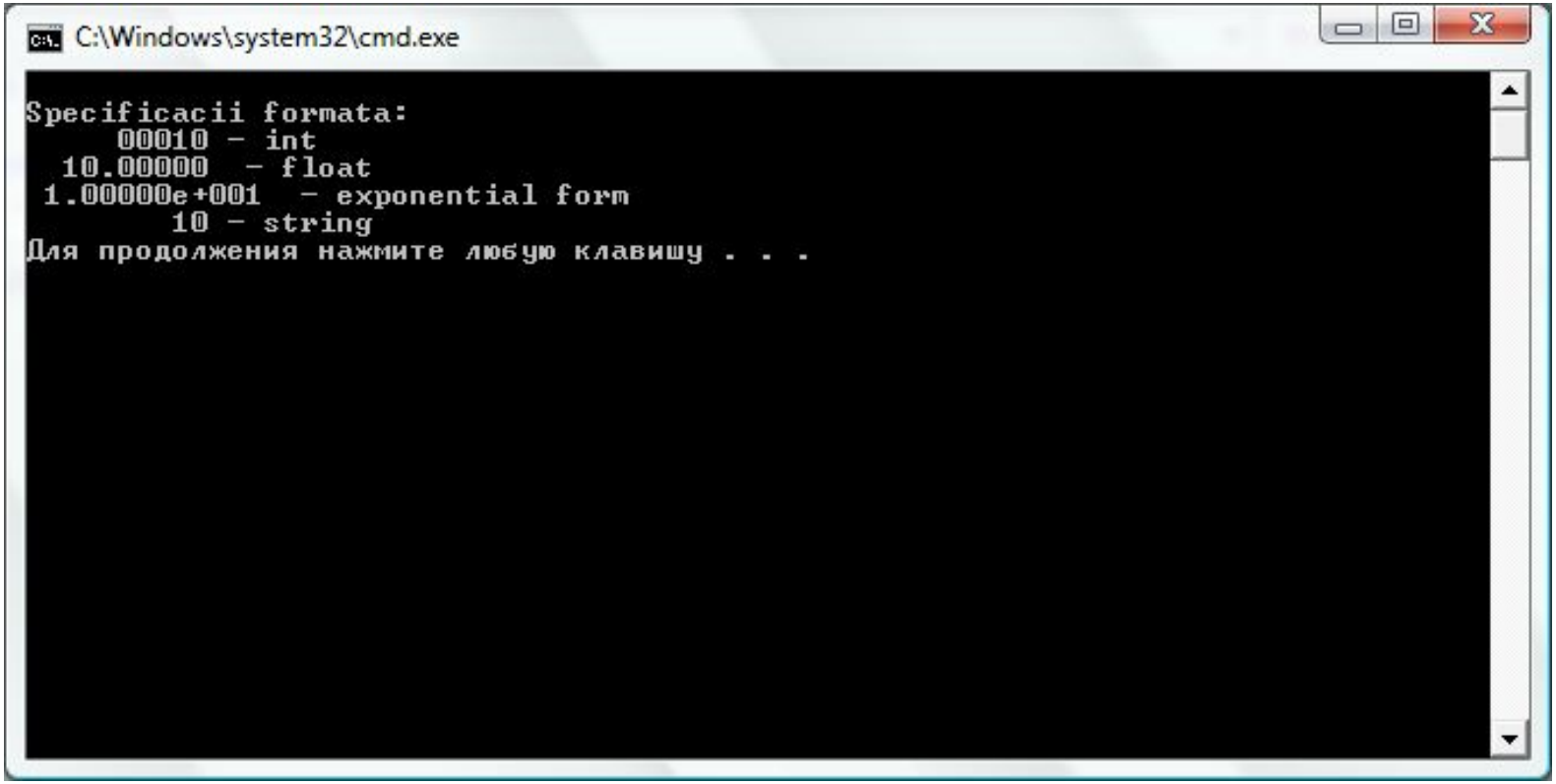
- Модификаторы формата– это числа, которые указывают минимальное количество позиций для вывода значения и количество позиций для вывода дробной части числа:

%[-]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 <cstdio>

using namespace std;
void main()
{
printf("\nSpecificacii formata:\n%10.5d - int\n%10.5f  - float\n %10.5e  - exponential
form\n%10s - string\n", 10, 10.0, 10.0, "10");
}
```



- Ввод:

scanf (форматная строка, список аргументов);

- в качестве аргументов используются адреса переменных.

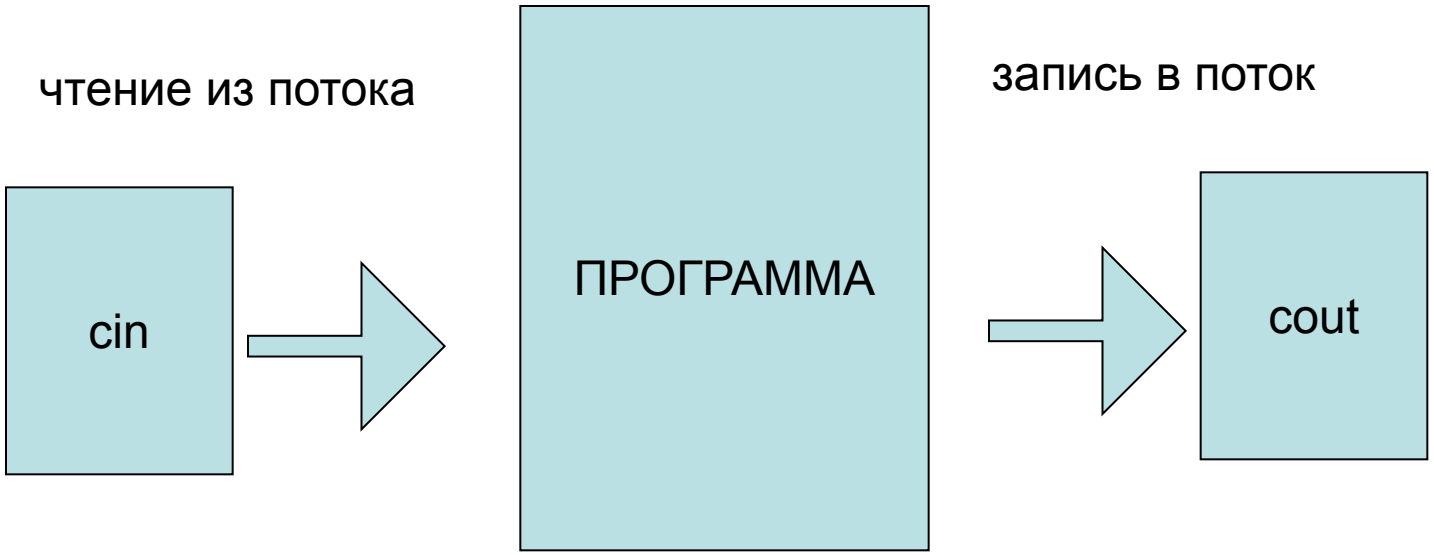
scanf(“ %d%f ”, &x,&y);

Некоторые спецификаторы для scanf() и printf()

- **%d, %i** десятичное целое
- %u беззнаковое десятичное целое
- **%o** беззнаковое восьмеричное целое
- **%x,%X** беззнаковое 16-ричное целое
- %c один символ
- %f вещественное значение
- %e,%E экспоненциальная форма вещественного числа
- %s строковое значение

Ввод-вывод (I способ)

```
#include <stdio.h>
void main()
{  int x,y,z;
   float t;
   printf("x = ");
   scanf(“%d”,&x);
   y = x*x;
   z = x/2;
   t = x/2.;
   printf("x^2  = %d\n",y);
   printf("x div 2 = %d \t x/2 =
%f\n",z,t);
}
```




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

…
cout << “\nВведите количество элементов: ”;
cin >> n;
```

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;

Ввод-вывод (II способ)

```
#include <iostream>
using namespace std;
void main()
{  int x,y,z;
  float t;
  cout << "x = ";
  cin >> x;
  y = x*x;
  z = x/2;
  t = x/2.;
  cout << "x^2  = “ << y << endl;
  cout << "x div 2 = " << z << “\t x/2 = “ << t;
}
```

Основные операции. Бинарные

Мультипликативные	
*	умножение операндов арифметического типа
/	деление операндов арифметического типа (если операнды целочисленные, то выполняется целочисленное деление)
%	получение остатка от деления целочисленных операндов
Аддитивные	
+	бинарный плюс (сложение арифметических операндов)
-	бинарный минус (вычитание арифметических операндов)

Задание 1

- **Найти сумму двух чисел.**

```
#include <iostream>
#include <locale.h>
using namespace std;
void main()
{
    setlocale(LC_ALL, "rus");       /* вывод русских букв */
    int a, b;   // объявление двух переменных a и b целого типа
    cout << "Введите первое число: ";
    cin >> a;       // ввод значения переменной a
    cout << "Введите второе число: ";
    cin >> b;
    int c = a + b;
    cout << "Сумма чисел = " << c << endl;          // вывод ответа.
}
```

Задание 2. Вариант 1

- **Найти сумму цифр двузначного числа. Используйте два варианта ввода-вывода.**

```
#include <iostream>
#include <locale.h>
using namespace std;
void main()
{
    setlocale(LC_ALL, "rus");       /* вывод русских букв */
    int n, S;
    cout << "Введите  число: ";
    cin >> n;
    S = n % 10 + n / 10;
    cout << "Сумма цифр = " << S << "\n";
}
```

Задание 2. Вариант 2

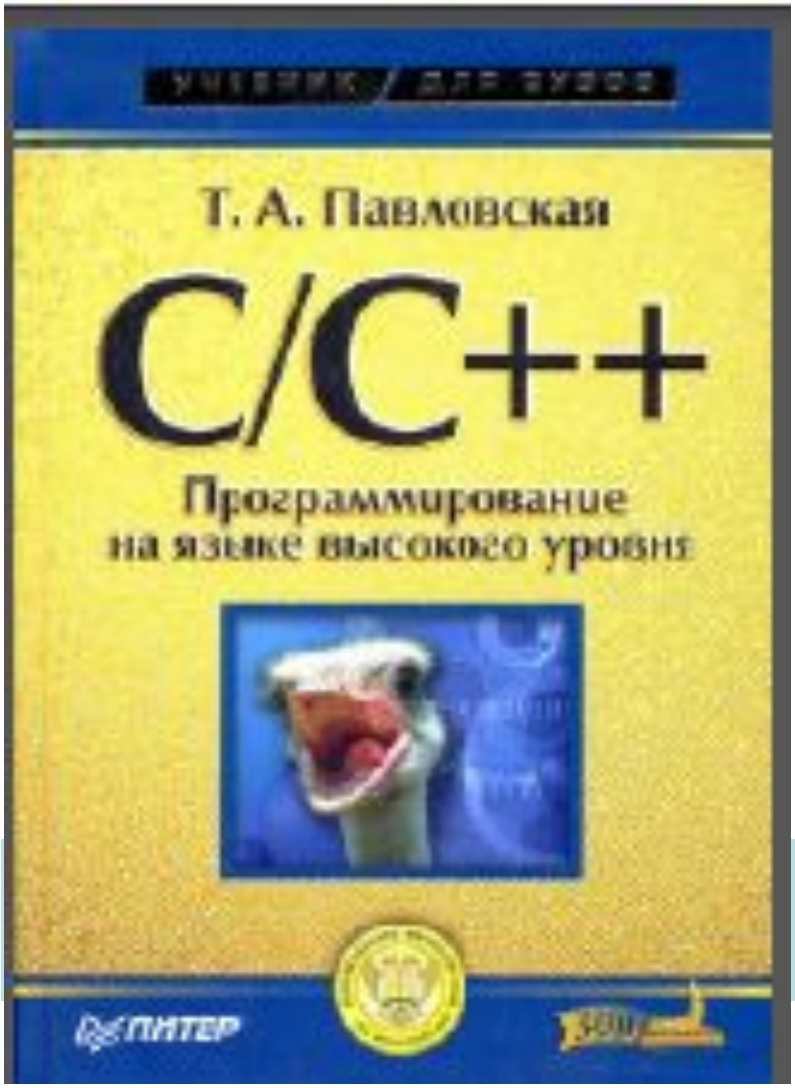
```
#include <stdio.h>
#include <locale.h>
void main()
{
    setlocale(LC_ALL, "rus"); /* вывод русских букв */
    int n, S; // объявление двух переменных a и b целого
    типа
    printf("Введите целое число: ");
    scanf("%d",&n);
    S = n % 10 + n / 10;
    printf("Сумма цифр = %d\n",S);
}
```

Задание 3

- Поменять значения двух переменных.

```
#include <stdio.h>
#include <locale.h>
void main()
{
    setlocale(LC_ALL, "rus"); /* вывод русских букв */
    int a,b,t;
    printf("Введите два целох числа: ");
    scanf("%d%d",&a,&b);
    t=a;
    a=b;
    b=t;
    printf("a=%d b=%d\n",a,b);
}
```

Домашнее задание



стр.11-44