ПРОГРАММИРОВАНИЕ

Преподаватель:
Лекции – Хахаев Иван Анатольевич (iakhakhaev@etu.ru)

СПбГЭТУ «ЛЭТИ»
ПЕРВЫЙ ЭЛЕКТРОТЕХНИЧЕСКИЙ

1-й семестр (осень):
Основы программирования на языке Си (C).

2-й семестр (весна):
Дополнительные вопросы программирования на Си.

Аргументы «за» и «против» выбора Си

- **«За»:**
  - Основной язык для системного программирования
  - Язык «низкого уровня», но не зависит от архитектуры
  - Позволяет создавать (почти) самые эффективные программы
  - В самом языке мало ключевых слов (простой).

- **«Против»:**
  - Используется прямое управление памятью -> вопросы безопасной работы программ ложатся на программиста
  - Нелогичен и методически не последователен -> не для начинающих
  - Базовые средства языка очень скудные, поэтому приходится использовать внешние библиотеки.

Наш путь

Создание программ на Си с использованием структурного подхода в процедурном программировании и особым вниманием к устранению потенциальных уязвимостей.

Комплексная задача

В произвольной строке, состоящей из слов с символами-разделителями, отсортировать слова в прямом лексикографическом порядке*. После перестановки слов последовательности символов-разделителей остаются на прежних позициях**.

* в соответствии с кодировочной таблицей.
** то, что было после N-го слова, остается после N-го слова, хотя само слово меняется.

Пример
Исходная строка:
```
ZZ Top 2000 blues... It is the beatyful music!
```

Результат обработки:
```
2000 It Top ZZ... beatyful blues is music the!
```

Постановка задачи (уточнения и доп. данные)

- Разделителями считается всё, что не буква и не цифра
- Строка не начинается с разделителя
- Строка может заканчиваться как буквой/цифрой, так и разделителем (разделителями)
- Слово — последовательность букв и цифр между разделителями (или с начала строки до первого разделителя, или от последнего разделителя до последней буквы/цифры в строке)
- Из слов и последовательностей разделителей формируются массивы
- Длина строки (общее количество символов в строке) заранее не известна, но не может превышать 256 символов
- Количество слов и последовательностей символов-разделителей в строке заранее не известно
- Строка может состоять из одного слова, но не может состоять только из разделителей (см. выше)
- Для упрощения работы с данными (промежуточного хранения) используются файлы
- Для ввода и вывода используются стандартные устройства (клавиатура и экран)
- Используются только набор символов латиницы и стандартные символы клавиатуры (символы ASCII).

Порядок решения задачи (алгоритм)

- Прочитать исходную строку
- Обработать строку
- Вывести результат обработки (полученную строку).

Порядок решения задачи (алгоритм)

- Прочитать исходную строку
- Обработать строку
  - Разделить строку на массивы слов и символов-разделителей, записать результаты разделения в файлы
  - Сформировать массив строк с разделителями из файла разделителей
  - Сформировать массив слов из файла слов
  - Отсортировать массив слов
  - Составить строку из элементов массива слов и массива строк с разделителями
- Вывести результат обработки (полученную строку).

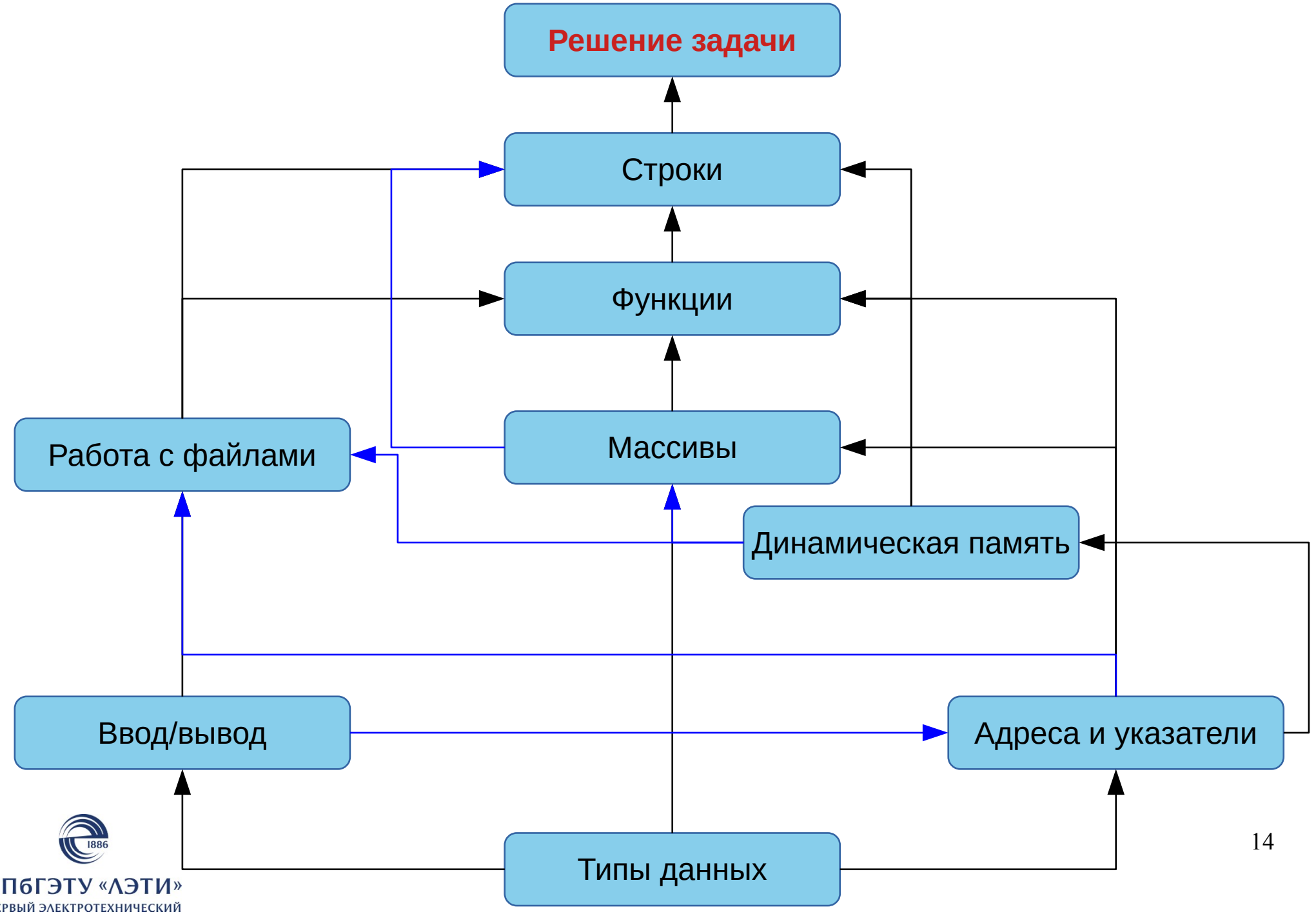
Порядок решения задачи (алгоритм)

- Прочитать исходную строку
- Обработать строку
  - Разделить строку на массивы слов и символов-разделителей
    - Читать символы пока не появится разделитель, сохранить полученное слово в новую строку в файле слов
    - Читать символы, пока не появится буква, сохранить последовательность символов-разделителей в новую строку в файле разделителей
    - Повторять два предыдущих действия, пока не кончится текст
  - Сформировать массив строк с разделителями из файла разделителей
  - Сформировать массив слов из файла слов
  - Отсортировать массив слов
  - Составить строку из элементов массива слов и массива строк с разделителями
- Вывести результат обработки (полученную строку).

Основные вопросы, требующие прояснения при решении комплексной задачи на Си

- Как организован ввод (и вывод)?
- Какие вообще типы данных есть и как с ними работать?
- Как устроены функции в Си?
- Как работать со строками (формировать, определять длину, сравнивать, делить и "склеивать")?
- Как определять буквы и другие символы?
- Как устроены массивы?
- Как работать с массивами, размеры которых неизвестны при запуске программы?
- Как сортировать массивы?
- Как работать с файлами?

Тематическая карта



Программа на Си

Программа на Си состоит из функций. Все функции записываются одна за другой.

Для каждой функции должны быть определены тип (если функция возвращает результат) и набор аргументов.

Если функция не возвращает результата, она должна иметь тип ***void***.

Главная функция называется **`main()`** и она всегда есть в любой программе.

При выполнении программы в первую очередь вызывается функция **`main()`**, все остальные функции должны вызываться из **`main()`** (или из функций, вызываемых из **`main()`**).

Программа:

Простейшая программа — только одна (главная) функция **`main()`**

Модули, единицы трансляции

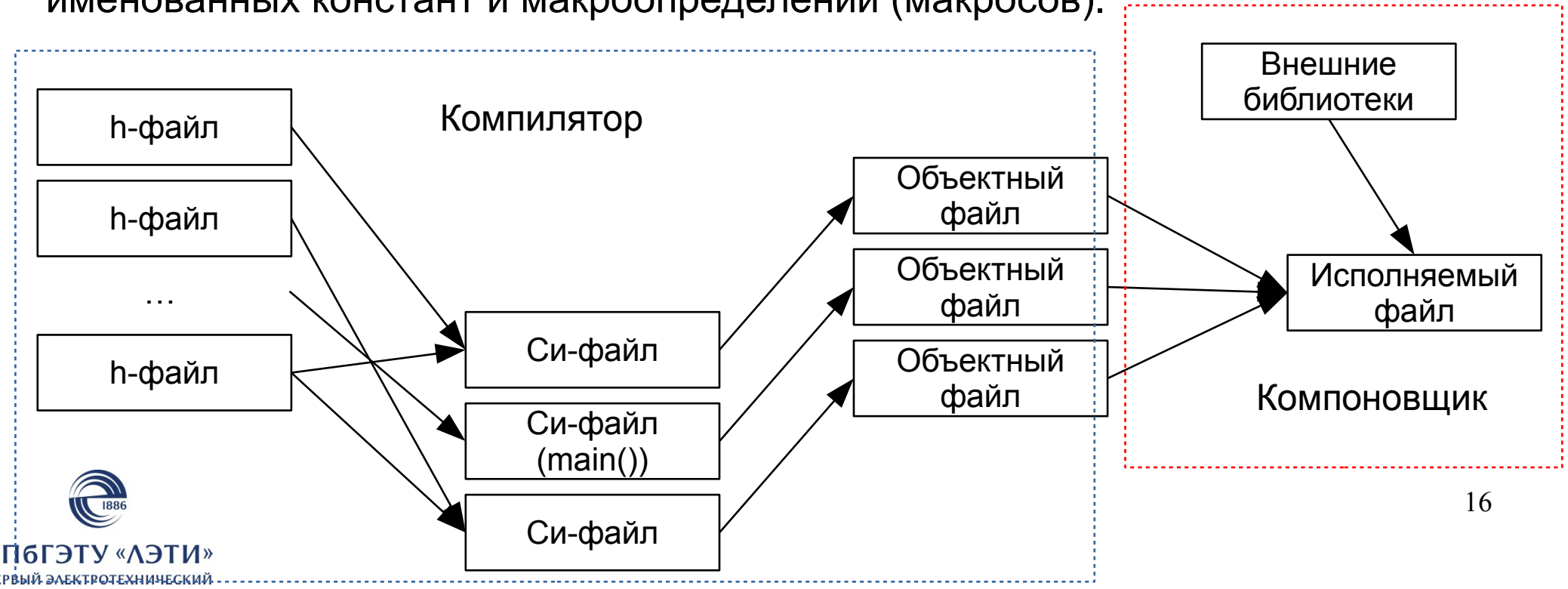
Модуль — отдельная функция в программе.

Единица трансляции — файл с исходным текстом (файл .c).

Если программа состоит из нескольких Си-файлов, то компилятор преобразует каждый Си-файл в объектный файл (файл .o).

Затем на этапе сборки компоновщик (линкер, редактор связей) связывает все объектные файлы в единую программу.

В h-файлах содержатся объявления функций (прототипы) и описания именованных констант и макроопределений (макросов).



Препроцессор Си

Препроцессор — часть компилятора, осуществляющая подготовку программы к компиляции.

Задачи препроцессора:

- включение содержимого одних файлов в другие
- замена имен констант на их значения в тексте исходного кода
- удаление символов конца строки
- …

Действия «по-умолчанию» выполняются всегда.
Дополнительные действия программируются с помощью директив препроцессора в исходном коде.

Директивы препроцессора начинаются со знака # и заканчиваются переходом на новую строку. В конце директивы не надо ставить точку с запятой.

Директивы препроцессора

#include – подключает к программе указанный файл

`#include <stdio.h>` – файл **`stdio.h`** ищется в системных каталогах

`#include "mylib.h"` – файл **`mylib.h`** ищется в текущем каталоге.

#define — определение констант и именованных действий (макросов)

`#define N 100` – N в программе будет иметь целое значение равное 100

`#define OOPS "Error!! Something wrong!"` – вместо сообщения об ошибке можно подставить OOPS

`#define SUM(a,b) a+b` – в программе всегда будет вычисляться сумма двух значений, если написать SUM(x,y).

Существуют также ***директивы условной компиляции***: действия включаются в исполняемую программу в зависимости от значений каких-то параметров (например, в зависимости от версии операционной системы).

Пример простой программы

Задача: Задается целое значение X. Нужно вычислить и вывести значение Y = X+5.

(см. пример lect-01-01.c)

Разбор кода:
- директива препроцессора
- тип, аргументы и возвращаемое значение функции **`main()`**
- описание переменных (выделение памяти)
- вывод сообщений программы
- ввод чисел, спецификатор формата (%i — для любых целых, %d — для целых в десятичной системе)
- вывод результата, «\n»
- окончание операторов
- отступы.

Типы данных в Си

Тип данных — базовое понятие для языков программирования.

Тип переменной определяет

- объем памяти (в байтах), выделяемый для хранения переменной
- способ внутреннего представления переменной (по битам)
- набор возможных операций с переменной данного типа
- точность представления.

Для получения объема памяти, выделяемого для данных какого-то типа, в Си существует функция **sizeof(<имя_типа>)**.

Например, sizeof(int) → 4 (байта).

Байт — минимально адресуемая компилятором область памяти.

Типы данных в Си

Целые числа, для `printf()`

Полное название типа	Краткое название типа	Спецификаторы форматов
signed char	char	%c
unsigned char	unsigned char	%c
signed short	short	%d или %i
unsigned short	unsigned short	%u
signed int	int	%d или %i
unsigned int	unsigned	%u
signed long	long	%ld или %li
unsigned long	unsigned long	%lu
signed long long	long long	%lld или %lli
unsigned long long	unsigned long long	%llu

Типы данных в Си

Вещественные числа, для `printf()`

Название типа	Спецификаторы форматов
float	%a, %A, %e, %E, %f, %F, %g, %G
double	%la, %lA, %le, %lE, %lf, %lF, %lg, %lG
long double	%La, %LA, %Le, %LE, %Lf, %LF, %Lg, %LG

Предельные значения для целых типов определены в файле **`limits.h`**

Предельные значения для вещественных типов вычисляются макросами из **`float.h`**

(см. пример lect-01-02.c)

Элементы языка Си

В любом языке (естественном или искусственном) существует ***алфавит***.

Из символов алфавита строятся ***слова***.

Из слов строятся ***выражения*** (тексты) в соответствии с ***грамматикой*** языка.

Правила построения корректных (правильных) выражений в соответствии с грамматикой кодируются ***формами Бэкуса-Наура*** (БНФ).

Семантикой языка называется смысловая нагрузка [корректных] выражений для этого языка.

Соответственно, для языка Си также существуют

- Алфавит
- Синтаксис
- Семантика

Элементы языка Си

Состав программы:

- Лексемы
  - Идентификаторы (переменные, функции, константы)
  - Ключевые слова
  - Литералы
  - Знаки операций и разделители (символы пунктуации: `[ ] { } ( ) < > , ;`)
- Комментарии

Лексема – единица текста программы, которая при компиляции воспринимается как единое целое и по смыслу не может быть разделена на более мелкие элементы.

Предложения (выражения) строятся из ***лексем.***

Комментарий — произвольная последовательность символов, поясняющая назначение частей программы (игнорируется при компиляции)

Элементы языка Си

Комментарий:
/* Это вариант комментария
который может занимать несколько строк */

// Это вариант комментария (начиная с C99)
// который должен умещаться в одну строку или в часть строки

```
printf("%d", n); /* вывод количества значений */
```

Комментарий нельзя писать внутри лексемы!!

Элементы языка Си

Идентификатор:

Последовательность букв, цифр и символа «_». Буквы — только английские, цифры — арабские. Используется для указания имени переменной, функции или константы (как в алгебре), например, x_1 (или x1).

Идентификатор не должен начинаться с цифры!!

Идентификаторы в Си чувствительны к регистру!

A1 и a1 — разные идентификаторы!

Длина идентификатора определяется стандартом, но может быть увеличена в конкретной реализации.
Для C89/C90 максимальная длина идентификатора — 31 символ, для C99 и C11 — 63 символа.

Константы

```
const int k = 123;
const double g1 = 9.81;
const char ch = 'u';
```

В пределах модуля (функции)

```
#define KMAX 123
#define G1 9.81
#define CH 'u'
```

Доступны для всех модулей (функций)

Элементы языка Си

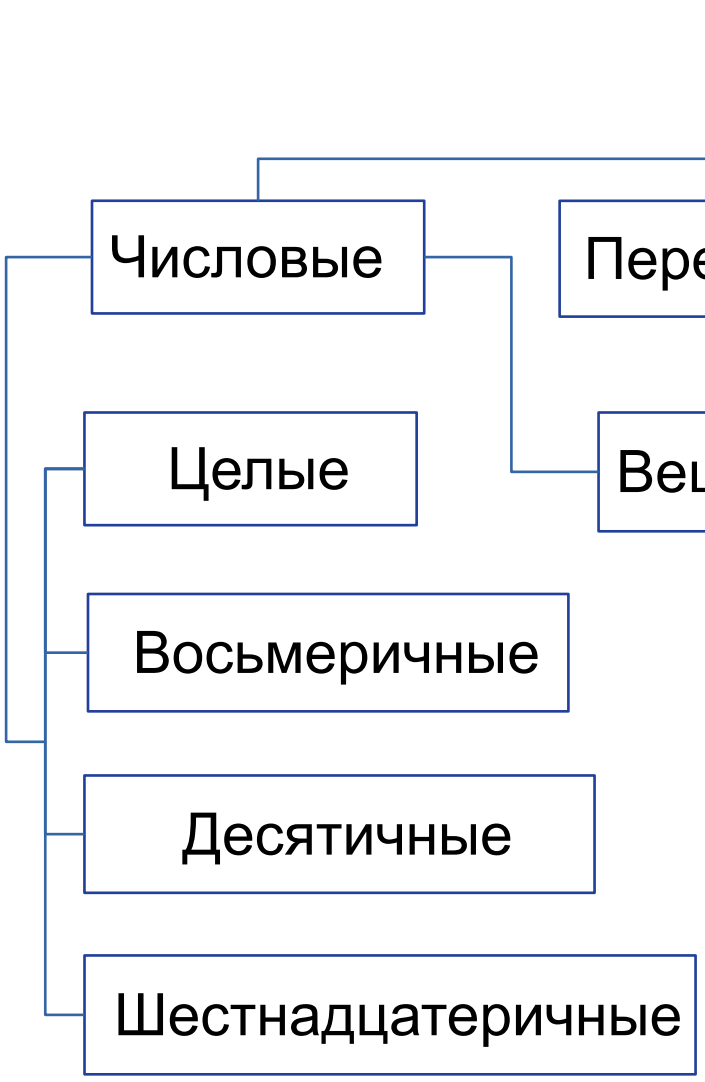
Ключевые слова (C89/C90):

auto	*extern*	sizeof
break	float	*static*
case	for	struct
char	***goto***	switch
const	if	typedef
continue	int	union
default	long	unsigned
do	*register*	void
double	return	*volatile*
else	short	while
enum	signed	

Ключевые слова зарезервированы в языке программирования, они ***не могут использоваться как идентификаторы***.

Ключевые слова всегда пишутся строчными (маленькими) буквами.

Элементы языка Си




```
ek=m*v*v/2.0;

s1=sizeof('D');

printf("Our result: %d\n", y);
```

(пример lect-01-03.c)

Литералы представляют собой константы, непосредственно включаемые в текст программы в отличие от прочих данных — констант и переменных, обращение к которым осуществляется посредством ссылок (по именам).

Литералы не могут быть изменены в процессе работы программы.

Выражения

Выражения описывают операции с переменными, функциями и константами (операндами).

Результат выражения всегда имеет тип (в соответствии с типом операндов). Если типы операндов различные, происходит приведение к наивысшему типу («повышение типа»).

Приоритет типов:

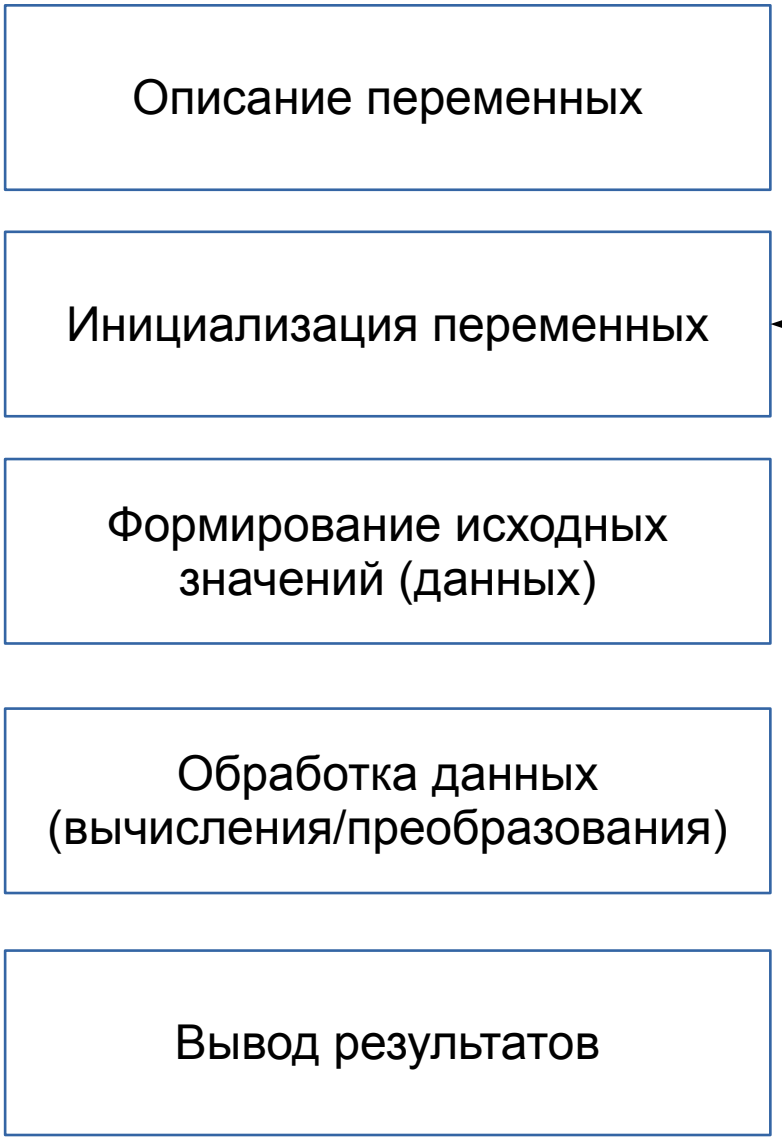
long double > double > float > unsigned long > long long > long > (unsigned int< - unsigned short <- unsigned char) > (int <- short <- char).

Приведение типов бывает *явное* и *неявное* (пример lect-01-04).

Если в умножении участвует вещественное число, результат может быть как «float», так и «double».

Тип «float» – 4 байта, «double» – 8 байтов.

Структура программы



Установка начальных значений параметров.
Параметры используются при обработке данных, но не являются исходными данными.

Простой ввод и вывод в Си

Ввод:

– Отдельные символы

```
c = getchar();
```

– Числа различных типов и символы в одном операторе

```
n=scanf("%d %lf %c", &a, &x, &c);
```

(n — общее число прочитанных значений, соответствующих формату).

Вывод:

– Отдельные символы

```
putchar(c);
```

– Сообщения

```
puts("Message for you");
```

– Результаты вычислений

```
printf("Results: %d %lf %c\n", a, x, c);
```

или

```
n=printf("Results: %d %lf %c\n", a, x, c);
```

(n — общее число выведенных символов).

Пример lect-01-05.c

Порядок разработки программы

I. Постановка задачи: что дано, что требуется получить, какие преобразования входных данных нужно выполнить (проверки, расчеты и т. п.), какие нужны дополнительные сведения и предположения.

II. Разработка алгоритма

III. Разработка тестовых (контрольных) примеров. Тесты должны позволять проверить все варианты, имеющиеся в алгоритме

IV. Тестирование алгоритма по примерам (трассировка) — какие значения принимают переменные на каждом шаге алгоритма

V. Написание программы

VI. Отладка программы – поиск ошибок.

В Си до запуска программы нужно выполнить две операции — ***компиляцию*** в объектный модуль и ***сборку*** исполняемого файла с использованием объектного модуля и редактора связей (линкера, компоновщика).

Библиотеки функций, описанных в заголовочных файлах (#include …), подключаются компоновщиком именно на этапе сборки.

Ошибки в программе

I. Синтаксические ошибки (ошибки в написании операторов) – обнаруживаются компилятором. Пока не устранены эти ошибки, программа не заработает.

II. Логические ошибки (ошибки в алгоритме) — обнаруживаются при тестировании алгоритма и программы с помощью контрольных примеров. При наличии таких ошибок программа работает, но неправильно.

III. Ошибки времени выполнения (сбои и «зависания») — обнаруживаются при некоторых комбинациях данных или при определенных условиях запуска программы. Самый плохой вариант ошибок. Для устранения требуется тщательное изучение кода, тестирование более чем в одной операционной системе (и, возможно, более чем с одним компилятором).