

ЯЗЫК C

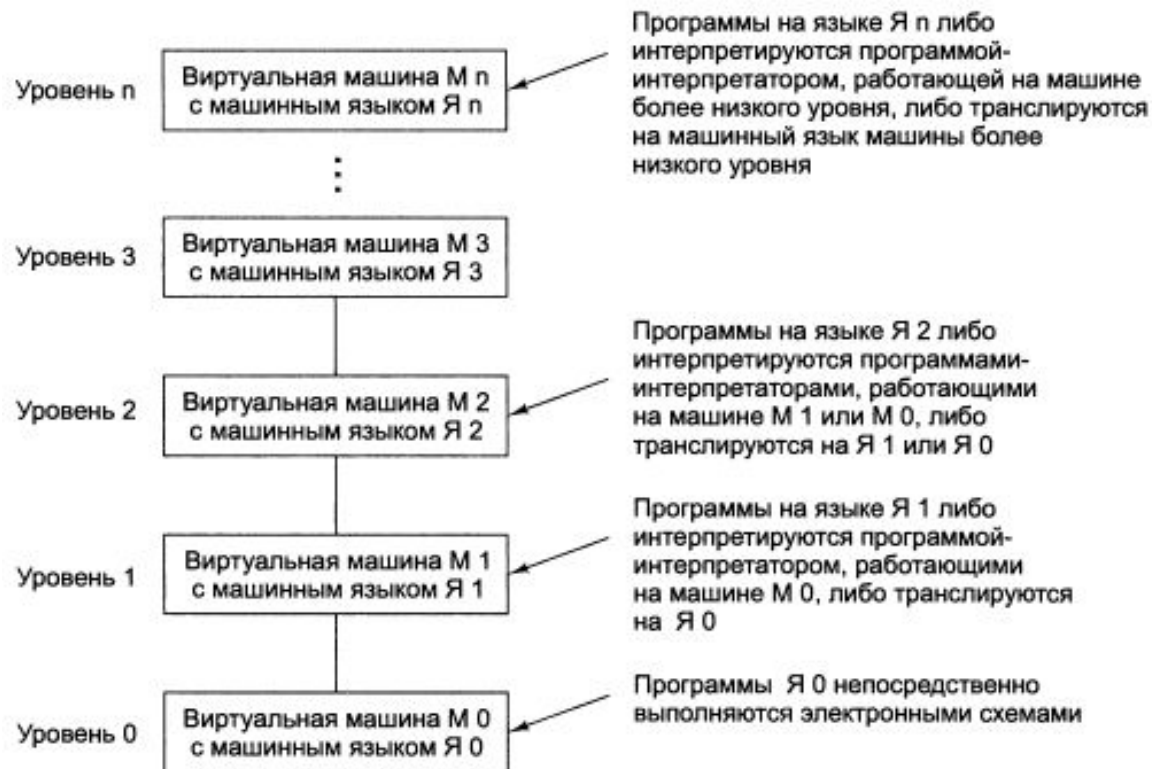
Лекция 1: Многоуровневая компьютерная организация, компиляция, типы данных, основные операторы языка C.

Многоуровневая компьютерная организация

Все программы перед выполнением должны быть превращены в последовательность машинных команд, которые обычно не сложнее, чем, например:

- сложить 2 числа;
- проверить, не является ли число нулем;
- скопировать блок данных из одной части памяти компьютера в другую.

- Большинство машинных языков крайне примитивны, из-за чего писать на них трудно и утомительно.
- Это привело к построению ряда уровней абстракций, каждая из которых надстраивается над абстракцией более низкого уровня. Появилась **многоуровневая компьютерная организация**.



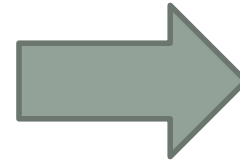
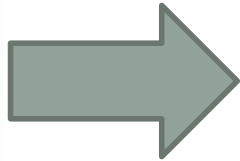


Интерпретация и трансляция

- **Интерпретация** — пооператорный (покомандный, построчный) анализ, обработка и мгновенное выполнение исходной программы или запроса. Выполняется интерпретатором.
- **Трансляция программы** — преобразование программы, представленной на одном из языков программирования, в программу на другом языке и, в определённом смысле, равносильную первой. Выполняется транслятором.
- В отличие от интерпретации, после трансляции мы получаем файл с программой.

Компилятор

- **Компилятор** — программа, выполняющая преобразование файла с исходным кодом программы в исполняемый файл.
- **Компиляция** — преобразование программы, составленной на исходном языке высокого уровня, в эквивалентную программу на низкоуровневом языке (машинном коде).
- Входной информацией для компилятора (исходный код) является описание алгоритма или программы на понятным человеку языке, а на выходе компилятора — эквивалентное описание алгоритма на машинно-ориентированном языке (машинный код).



Создание переменных

- <тип данных> <имя_переменной>;
- **int** a; //целочисленная переменная
- **char** c; //символьная переменная
- **float** f; //переменная с плавающей точкой

Типы данных

Целые типы	Размер в байтах (битах) x86/x64	Интервал изменения (для x86)
char	1 (8)/1(8)	от -128 до 127
unsigned char	1 (8)/1(8)	от 0 до 255
signed char	1 (8) /1(8)	от -128 до 127
int	2 (16) /4(32)	от -32768 до 32767
unsigned int	2 (16)/4(32)	от 0 до 65535
signed int	2 (16)/4(32)	от -32768 до 32767
short int	2 (16)/2(16)	от -32768 до 32767
unsigned short int	2 (16) /2(16)	от 0 до 65535
signed short int	2 (16)/2(16)	от -32768 до 32767
long int	4 (32)/8(64)	от -2147483648 до 2147483647
unsigned long int	4 (32) /8(64)	от 0 до 4294967295
signed long int	4 (32) /8(64)	от -2147483648 до 2147483647

Типы с плавающей запятой	Размер в байтах (битах)	Интервал изменения
float	4 (32)	от 3.4E-38 до 3.4E+38
double	8 (64)	от 1.7E-308 до 1.7E+308
long double	10 (80)	от 3.4E-4932 до 3.4E+4932

1 килобайт = $1024^1 = 2^{10}$ = 1024 байт

1 мегабайт = $1024^2 = 2^{20}$ = 1 048 576 байт

1 гигабайт = $1024^3 = 2^{30}$ = 1 073 741 824 байт

1 терабайт = $1024^4 = 2^{40}$ = 1 099 511 627 776 байт

1 петабайт = $1024^5 = 2^{50}$ = 1 125 899 906 842 624 байт

1 эксабайт = $1024^6 = 2^{60}$ = 1 152 921 504 606 846 976 байт

1 зеттабайт = $1024^7 = 2^{70}$ = 1 180 591 620 717 411 303 424 байт

1 йоттабайт = $1024^8 = 2^{80}$ = 1 208 925 819 614 629 174 706 176 байт

Языки программирования

- Разделяются по уровню абстракции:
- Машинные языки
- Языки низкого уровня (Assembler-ы)
- Языки высокого уровня

Языки высокого уровня

- Высокоуровневые языки программирования разработаны для быстроты и удобства использования программистом. Главная особенность — замена длинных и (порой) сложных для понимания описаний на машинном коде простыми смысловыми конструкциями — абстракцией.
- Также высокоуровневые языки могут брать на себя часть задач стоящих перед программистом: определение размеров выделяемых ячеек памяти, работа с массивами, определение типа данных (динамическая типизация Python, PHP).

IDE

- **Интегри́рованная среда́ разрабо́тки, ИСР** (*Integrated development environment — IDE*) — комплекс программных средств, используемый программистами для разработки программного обеспечения (ПО).
- Среда разработки включает в себя:
 1. Текстовый редактор
 2. Компилятор и/или интерпретатор
 3. Средства автоматизации сборки
 4. Средства отладки

Этапы компиляции программы

- Обработка исходного кода программы (препроцессор).
- Получившийся полный текст программы обрабатывается компилятором, который определяет синтаксические лексемы и наличие ошибок в коде. В случае успешной компиляции создается объектный файл.
- Далее компоновщик, или редактор связей, формирует исполняемый файл программы.

Препроцессор

- Препроцессор — программа(обычно часть компилятора), преобразующая исходный текст программы согласно указанным командам (**директивам**)
- `#include` — вставляет текст из указанного файла
- `#define` — задаёт макроопределение (макрос) или символическую константу
- `#undef` — отменяет предыдущее определение

Парадигма программирования

- **Парадигма программирования** — это комплекс концепций, принципов и абстракций, определяющих стиль программирования.
- Все современные языки программирования в той или иной мере допускают использование различных парадигм

Языки С и С++

- Язык **С** — это императивный, типизированный, высокоуровневый, машинно-ориентированный, не объектно-ориентированный язык программирования.
- Язык **С++** является его объектно-ориентированным расширением.

Обобщённое программирование

- **Обобщённое** программирование — парадигма программирования, заключающаяся в таком описании данных и алгоритмов, которое можно применять к различным типам данных, не меняя само это описание.
- Обобщённое программирование в некотором смысле является логическим продолжением ООП.
- Примеры: Java, C++

Модульное программирование

- **Модульное** программирование — это организация программы как совокупности небольших независимых блоков, называемых модулями, структура и поведение которых подчиняются определённым правилам.
- Примеры: Pascal, Java, Python, C++

Декларативное программирование

- **Декларативная** парадигма программирования определяет процесс вычислений посредством описания логики самого вычисления, а не управляющей логики программы.
- Программа «декларативна», если она описывает, каково нечто, а не как его создать.
- Пример: веб-страницы на HTML

Императивное программирование

- **Императивное** программирование — это парадигма программирования, которая, в отличие от декларативного программирования, описывает процесс вычисления в виде инструкций, изменяющих состояние данных.
- Императивная программа очень похожа на приказы, выражаемые повелительным наклонением в естественных языках, то есть это последовательность команд, которые должен выполнить компьютер.
- Пример: язык `Assembler`'а

Процедурное программирование

- **Процедурное** программирование — программирование на императивном языке, при котором последовательно выполняемые операторы можно собрать в подпрограммы, то есть более крупные целостные единицы кода, с помощью механизмов самого языка.
- Примеры: Pascal, Си

Объектно-ориентированное программирование

- **Объектно-ориентированное** программирование (ООП) — парадигма программирования, в которой основными концепциями являются понятия объектов и классов.
- ООП возникло в результате развития методологии процедурного программирования, где данные и подпрограммы (процедуры, функции) их обработки формально не связаны.
- Примеры: Java, C++

Операторы C

Операция (выражение)	Оператор	Синтаксис
Присваивание	=	a = b
Сложение	+	a + b
Вычитание	-	a - b
Унарный плюс	+	+a
Унарный минус	-	-a
Умножение	*	a * b
Целочисленное деление	/	a / b
Остаток от деления	%	a % b
Префиксный инкремент	++	++a
Постфиксный инкремент	++	a++
Префиксный декремент	--	--a
Постфиксный декремент	--	a--

Логические операторы C

Операция (выражение)	Оператор	Синтаксис
Сравнение	<code>==</code>	<code>a == b</code>
Сравнения не равно	<code>!=</code>	<code>a != b</code>
Сравнение меньше	<code><</code>	<code>a < b</code>
Сравнение больше	<code>></code>	<code>a > b</code>
Сравнение меньше или равно	<code><=</code>	<code>a <= b</code>
Сравнение больше или равно	<code>>=</code>	<code>a >= b</code>
Логическое НЕ	<code>!</code>	<code>!a</code>
Логическое И	<code>&&</code>	<code>a && b</code>
Логическое ИЛИ	<code> </code>	<code>a b</code>

Приведение к типу данных

- Приведение типов данных — изменение типа данных компилятором или программистом с одного на другой. Например из `int` в `float` или из `char` в `int`.
- Приведение вызывается с помощью скобок.
(`<тип данных>`)`<имя_переменной, оператор или функция>`

х	у	Результат деления	Пример
делимое	делитель	частное	х = 15 у = 2
int	int	int	15/2=7
int	float	float	15/2=7.5
float	int	float	15/2=7.5