Лекция 2

Введение в C/C++.

Системы счисления

Базовые типы данных

Важен не язык - важно умение мыслить!
Языки программирования по синтаксису и семантике во многом похожи!
Ищите смысл и проникайтесь алгоритмами, а не непонятно как работающими «заготовками»!

```
for (int i = 0; i < n; i++)  s += i; // C/C++
```

Изотерика :–)

Тернарная операция **?:** выполняется над тремя операндами

```
1   int max = a > b ? a : b;
2   ++i + ++i;  //при i=1, выражение =6!
3   (x = 7) + (c = 10 + x);
4   ++(n = (k = 5) % 2);
5   if (-1 > (unsigned int) 1)  *p++;
```

https://neolurk.org/wiki/%2B%2Bi_%2B_%2B%2Bi

Истина!!!

Основные элементы ЯП

Алфавит – набор символов или групп символов, рассматриваемых как единое целое, с помощью которых составляется текст программы.

Оператор (инструкция, *statement*) – основная конструкция ЯП, определяющая конкретное действие.

Синтаксис – правила построения конструкций ЯП с помощью алфавита и операторов языка.

Семантика – смысл и правила использования этих конструкций.

Найдите ошибки:

```
#include <stdio.h>
int main()
{  int a, b;
   scanf("%d %d", a, b);
   printf("a = %d, b = %d\n", a, b);
   return 0;
}
```

→ `scanf("%d %d", &a, &b);`

```
int main()
{  int a, b;
   printf("a = %d, b = %d\n", a, b);
   scanf_s("%d %d", &a, &b);
   return 0;
}
```

поменять местами

```
printf("\n%c, %d, %f, %s", x, y, z);
```

Как создаются исполняемые файлы с расширением .exe

- Создание исходного файла с помощью программы текстового редактора **name.cpp**
- Препроцессорная обработка – программа передается препроцессору, который выполняет директивы, содержащиеся в ее коде (***директивы начинаются со знака #***)
- Компиляция (лексический, синтаксический, семантический анализ) - перевод исходного модуля в объектную программу на машинном языке или языке низкого уровня. **name.obj**
- Компоновка (сборка) программы – редактор связей связывает все объектные модули (файлы) в исполняемую программу (загрузочный модуль) **name.exe**
- Загрузка (запуск) программы на выполнение – загрузочный модуль программы с помощью программы загрузчика помещается в оперативную память
- Выполнение программы

Этапы компиляции и компоновки на C/C++

Можно в литературе более подробно ознакомиться с процессом компиляции программ

Он включает следующие этапы:

- Лексический анализ. Последовательность символов исходного файла преобразуется в последовательность лексем
- Синтаксический анализ. Последовательность лексем преобразуется в дерево разбора
- Семантический анализ
- Оптимизация
- Генерация кода

Основные этапы решения задачи на ЭВМ

1. Постановка задачи (разработка ТЗ)
2. Выбор метода решения
3. Разработка алгоритма решения задачи

Свойства алгоритма:

- Массовость
- Понятность
- Правильность
- Результативность

Не верно

Верно

C	
`x=1;`	`scanf("%d",&x);`
C++	
`x=1;`	`cin >> x;`

4. Разработка алгоритма в заданной среде программирования (IDE)
5. Отладка и тестирование программы
6. Анализ полученных результатов

Вопрос*: *Что такое

тестирование – процесс выявления ошибок;

отладка – поиск места ошибки с целью ее исправления.

Тестирование ПО – его исследование с целью получения информации о качестве продукта.

Набор тестов ⟶ сравнение результатов с эталоном с целью выявления ошибок.

```
//какой тест выявит ошибку в программе?
int main()
{  int a, b;
   scanf_s("%d %d", &a, &b);  //Ввод в b ноль
   printf("\n a/b = %d ", a/b);
   return 0;
}
```

Самая дорогая точка в программировании

Космический аппарат США Маринер-1 22.07.1962 должен был направиться к Венере, но был уничтожен из-за аварии через 293 секунды после старта.

Фрагмент кода на Фортране

DO 17 I = 1, 10 **DO 17 I = 1.10**

Скрытая ошибка!!!

Код интерпретирован как **DO17I = 1.10**,
т.е. переменной DO17I (пробельные символы языком не учитываются) присвоить значение 1.10

Необходимо быть уверенным, что основные компоненты программной системы реализованы правильно и не подведут в критический момент. Для этого при проектировании выполняется верификация и валидация системы.

Верификация – оценка, подтверждение корректности ПО аналитическими методами с целью гарантированности правильности программы, соответствия заданным условиям, ТЗ на ее проектирование. Сопровождает весь жизненный цикл ПО.

Валидация – процесс оценки того, насколько программа соответствует требованиям применения ее по назначению (требованиям заказчика).

Верификация задается вопросом, строим ли мы систему правильно, а **валидация** – строим ли мы правильную систему.

Идентификаторы

В общем виде **идентификатор** (сокращенно **id**) – это некоторое уникальное имя объекта.

В программировании **идентификаторы** – это имена переменных, констант, функций, пользовательских типов, классов и других объектов, определяемых пользователем.

```
number, x, summa, step_1
```

~~`1number, step*`~~

Идентификатор - это последовательность букв, цифр и символов подчёркивания, которая начинается с буквы или символа подчёркивания ("_") и не содержит пробелов.

Язык C/C++ чувствителен к регистру букв

`Summa` и **`summa`** - ***разные идентификаторы***.

Переменные и их типы

Переменная – именованная величина, значение которой может меняться.

- Имя переменной – идентификатор.
- Переменные должны быть объявлены до их использования.
- Переменные должны быть инициализированы до их использования.
- Строчные и прописные символы различаются

  `number ≠ Number`
- При объявлении переменной указывается ее **тип**

```
char ch;   //символьный 1 байт
int num;   //целый 4 байта
float y;   //вещественный 4 байта
```

Тип переменной определяет

- множество значений, которые она может принимать
- размер памяти для хранения значений данного типа
- множество операций, применимых к типу

```
'A'/'B' == 0
'B'/'A' == 1
```

Базовые (встроенные) типы данных

- Целый - `int`
- Символьный - `char`
- Вещественный (числа с плавающей точкой) - `float`
- Логический - `bool`

Кроме ***базовых (встроенных) типов данных***, можно создавать свои типы данных – ***пользовательские типы данных***.

Диапазон чисел типа `int` зависит от компьютера, на котором выполняется программа.

Когда процессоры были 16-битными, `int` было 2 байта.

На 32-разрядных и 64-разрядных системах `int` занимает 4 байта

```
printf("%d", sizeof(int));
```

Тип	байт	Диапазон принимаемых значений
целочисленный (логический) тип данных		
bool	1	true и false
целочисленный (символьный) тип данных		
char	1	0 ÷ 255
целочисленные типы данных		
short int	2	-32 768 ÷ 32 767
unsigned short int	2	0 ÷ 65 535
int	4	-2 147 483 648 ÷ 2 147 483 647
unsigned int	4	0 ÷ 4 294 967 295
long int	4	-2 147 483 648 ÷ 2 147 483 647
unsigned long int	4	0 ÷ 4 294 967 295
типы данных с плавающей точкой		
float	4	-2 147 483 648.0 ÷ 2 147 483 647.0
long float	8	-9 223 372 036 854 775 808 .0 ÷ 9 223 372 036 854 775 807.0
double	8	-9 223 372 036 854 775 808 .0 ÷ 9 223 372 036 854 775 807.0

Рассмотрим пример хранения значений типа short int

Это знаковый двухбайтовый тип данных, с помощью которого можно сохранить **? 2^{16}** значений (-32 768 ÷ 32 767)

$2^{16} = 65536$

Бит знака

-32768 . . 32767

32767

0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

байт (8 бит) **байт**

-32768

1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

-1

1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Чему равно максимальное 2-х байтовое беззнаковое целое?

65535

Отрицательные числа хранятся в обратном коде

Представление чисел в двоичной, восьмеричной и 16-ричной системах счислений

Память компьютера состоит из ячеек – разрядов. В разряд можно записать либо **0**, либо **1**. Разряд может хранить 1 бит информации. Но разряд слишком мал для работы с информацией.

1 байт – 8 бит	2^0
1 КБ – 1024 байт	2^{10}
1 МБ – 1024 КБ	2^{20}
1 ГБ – 1024 МБ	2^{30}
1 ТБ – 1024 ГБ	2^{40}
1 ПБ – 1024 ТБ	2^{50}
1 ЭБ – 1024 ПБ	2^{60}
1 ЗБ – 1024 ПЭ	2^{70}
1 ИБ – 1024 ЗБ	2^{80}

Терабайт петабайт эксабайт зеттабайт йоттабайт

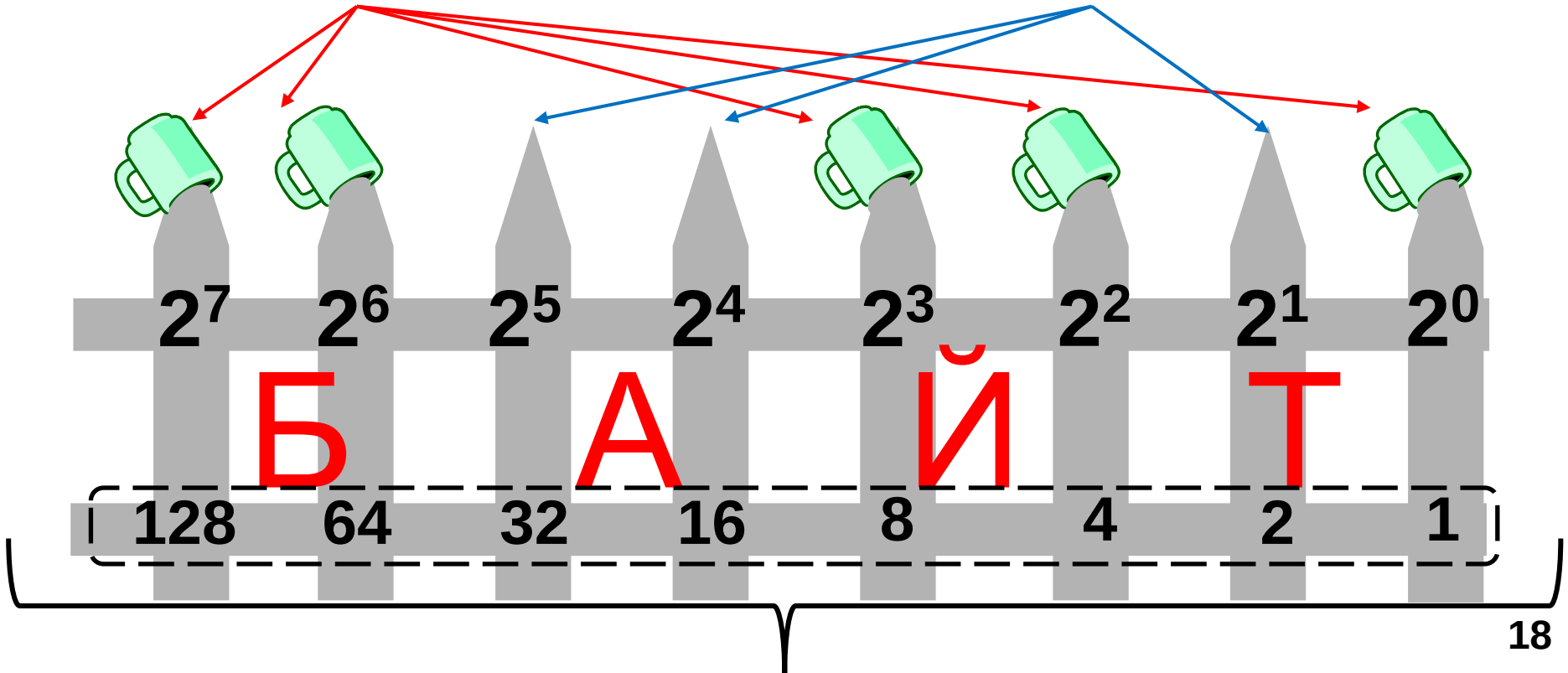
Байт - 8 бит

Нумерация битов в байте начинается с 0 справа налево

7	6	5	4	3	2	1	0
1	1	0	0	1	1	0	1

Разряды, в которых записана 1

Разряды, в которых записан 0



10 с.сч.	2 с.сч.	8 сист.сч.	16 сист.сч.
1	2^0 1	1	1
2	2^1 10	2	2
3	11	3	3
4	2^2 100	4	4
5	101	5	5
6	110	6	6
7	111	7	7
8	2^3 1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C

10 с.сч.	2 с.сч.	8 сист.сч.	16 сист.сч.
13	1101	15	D
14	1110	16	E
15	1111	17	F
16	2^4 10000	20	10

32	2^5	100000
64	2^6	1000000
128	2^7	10000000

! **Нумерация битов в байте начинается с 0 справа налево**

Заполнить таблицу:

10 сист.сч.	8 сист.сч.	16 сист.сч.	2 сист.сч.
15	17	F	0000 1111
100	144	64	0110 0100
303	457	12F	10 0101 0111

Максимальное беззнаковое однобайтное целое - . . . ?

255_{10}

1111 1111

377_8

1111 1111

FF_{16}

0111 1111 - . . . ?

$127_{10}, 177_8, 7F_{16}$

Максимальное беззнаковое двухбайтное целое?

65 535 **0xffff**

Максимальное беззнаковое 4-хбайтное целое?

4 294 967 295 **0xffffffff**

16-ричная система счисления гораздо проще!
Все ячейки памяти имеют адреса, и это именно 16-ричные значения

Д/З

Вывести на экран двоичный код числа.

Решение: например, целым переменным x1, x2, x3 присваиваем десятичное, восьмеричное или 16-ричные значения:

```
int x1 = 123;
int x2 = 0123;
int x3 = 0x123;
```

далее выводим их двоичное представление