Lecture_01

> **В лекции**
>
> 1. Введение в программирование на примере языка C.
> 2. Введение (переход) в язык программирования 1C

Введение в программирование на примере языка C

В данной части мы не будет сильно углубляться в специфику языка программирования **C,** а лишь рассмотрим на его примере базовые конструкции. Это делается для того, чтобы вспомнить уже ранее известные практики программирования и попытаться плавно наложить уже известные знания на язык программирования **1C**.

Самая первая программа -- вывода строковой константы "**Hello, world**" в поток вывода.

```c
#include <stdio.h>

main()
{
    printf("hello, world");
}
```

Программа на C независимо от ее размера состоит из функций и переменных. Функция содержит **операторы** -- команды для выполнения определенных вычислительных операций, а в переменных хранятся числа и другие данные, используемые в этих операция.

Мы можем определить сколь угодно функций, но в C-подобных языках программирования функция `main(...)` является особым случаем. Любая программа начинается с функции `main(...)`.

Для определения функции необходимо указать ее **название** (название должно отражать логику функции), **передаваемые аргументы в скобках**, **границы самой функции**. В C-подобных языках граница определяется фигурными скобками.

- Название
- Аргументы
- Границы

Далее из функции `main(...)` мы вызываем функцию `printf(...)`, которая в качестве аргумента может принимать строковую константу и выводить ее в поток вывода.

Суть использования функций заключается в умышленном сокрытии кода (инкапсуляция) за некоторым псевдонимом (названием функции). Если мы знаем, что по выполнению функции случиться, то нам необязательно помнить то, как она работает. Также выделение некоторого кода в функцию облегчает процесс восприятия смысла и структуры кода.

Теперь попробуем написать что-то более сложное.

```c
#include <stdio.h>

// Вывод таблицы температур по Фаренгейту и Цельсию.

main()
{
    int fahr, celsius;
```

```
    int lower = 0, upper = 300, step = 20;

    fahr = lower;
    while(fahr <= upper) {
        celsius = 5 * (fahr-32) / 9;
        printf("%d\t%d\n", fahr, celsius);
        fahr = fahr + step;
    }
}
```

> Данная программа выводит таблицу температуры по Фаренгейту и их соответствий по шкале Цельсия.

В данной программе появляется новые конструкции, такие как **комментарии**, **объявления**, **переменные**, **арифметические выражения**, **циклы** и **форматированный вывод**.

- Комментарии содержат краткое описание того, что делает программа. Любые символы после `//` или `/* */`, игнорируется компилятором.
- Переменные -- некоторые структуры в коде для хранения данных. Обычно переменные объявляется до любых выполняемых операторов (обычно это в начале функции). Также в строго-типизированных языках программирования требуется указывать тип данных перед объявлением переменной. Самые часто используемые типы данных: `int` -- целые числа, `float` -- вещественные числа с плавающей точкой, `double` -- вещественное число с двойной точностью, `char` -- символ. Ключевым моментом в использовании типов данных является их размер, если необходимо записать очень большое число, то оно точно не поместить в тип `int`, диапазон которого от -2 147 483 648 до 2 147 483 647 (при 32-битном разряде), поэтому можно использовать тип большего размера, например, `long`.
- Арифметические выражения представляют собой базовые математические операции. Они позволяют получать новые данные на основе уже полученных. Символы, представляющие математические операции (**сложение**, **вычитание** и т.д.) **называются операторами**.
- Циклы -- управляющая конструкция, которая позволяет выполнять один и тот же фрагмент кода многократно до тех пор, пока не выполниться некоторое условие. Циклы также содержат в себе операторы, в данном примере содержится оператор сравнения **меньше-или-равное**: `<=`.
- Форматированный вывод -- способ задания стиля вывода символов в поток вывода. Для того, чтобы компилятор мог различать символы на вывод и символы, отвечающие за форматирование, вводятся **управляющие последовательности (escape sequence)**:
  - `\n` -- перенос строки;
  - `\t` -- табуляция или четыре пробела;
  - `\b` -- возврат на один символ назад с затиранием;
  - `\"` -- двойная кавычка;
  - `\\` -- обратная косая черта.

Однако любую программу можно переписать короче, в данном случае мы можем упростить код используя цикл for.

Цикл `for` во много является обобщением цикла `while`. Цикл `for` состоит, как правило, из трех выражение: **инициализация**, **условия**, **приращения шага**. Выбор между `while` и `for` определяется соображениями ясности программы. Цикл `for` более удобен в тех случаях, когда инициализация и приращение шага логически связаны друг с другом общей переменной и выражаются единичными инструкциями, что позволяет значительно сократить код.

```
#include <stdio.h>

main()
{
    int fahr;
    for (fahr = 0, fahr <= 300; fahr = fahr + 20) {
        printf("%3d %6.1f\n", fahr, (5.0/9.0)*(fahr-32));
```

```
    }
}
```

Цикл `for(...)` может быть записан как:

1. `for(int a = 0; a < n; ++a) {};`
2. `for (;a < n;) {}` -- как while;
3. `for (;;++a) {};`
4. `for (int a = 0;;) {};`
5. `for(;;)` -- бесконечный цикл.

Цикл foreach

Цикл **foreach** не является частью языка C, но его можно встретить во многих производных языках программирования (C++, C#, 1C). Цикл **foreach** позволяет перебирать коллекции не по индексам, а через итератор, делается это от начала и до конца коллекции.

```
for (int n : array) {}
```

Условные конструкции

Условные конструкции позволяют проверять выражения на истинность или ложность и выполнять соответствующие результату проверки операторы:

```
if (выражение) {
    оператор1;
}
else
    оператор2;
```

> Выполняются всегда только один оператор из двух, ассоциированных с конструкцией `if-else`. Если выражение `Правда`, выполняется `оператор1`, в противном случае выполняется `оператор2`.

Также существует конструкция `if-else if-else`, которая позволяет накладывать дополнительное-альтернативное условие на выражение, если проверка в конструкции `if(...)` определяет выражение как Ложное.

```
if (выражение1) {
    оператор1;
}
else if (выражение2) {
    оператор2;
}
else
    оператор3;
```

> В остальном работа аналогична конструкции `if-else`.

Операторы continue и break

В ряде случаев может возникнуть, по условию, необходимость выйти из цикла до его завершения. В этом случае возможно воспользоваться оператором `break`.

```
#include <stdio.h>

main()
{
   char c;
```

```
    for(;;) {
       printf_s( "\nPress any key, Q to quit: " );

       // Convert to character value
       scanf_s("%c", &c);
       if (c == 'Q')
           break;
    }
} // Loop exits only when 'Q' is pressed
```

> Если пользователь вводит символ `'Q'`, то выполняется прерывание бесконечного цикла и программа завершает свое выполнение.

В отличие от оператора break, оператора `continue` производит переход к следующей итерации.

```
#include <stdio.h>

main()
{
    int c;
    for (int c; (c = getchar()) != EOF;) {
        if (c == '\n' || c == '\t' || c == '\b' ||...) {
            continue;
        }
        putchar(c);
    }
}
```

В данном случае код выполняет копирование входного потока в выходной поток. Но что если в файле встретится символ управляющей последовательности и мы не хотим его копировать в выходной поток. Для добиться этого мы можем проверять каждый символ на **символ управляющей последовательности**, если он таковым является, то оператор **continue** позволяет пропустить последующие в цикле инструкции и сразу перейти на следующую итерацию (чтение нового символа).

Язык программирования 1С. Платформа 1С:Предприятие

Платформа позволяет разрабатывать информационные системы различного назначения и относится к предметно-ориентированным средствам разработки. Решения, построенные на основе **Платформы** принято называть **конфигурациями**. Конфигурации строятся из базовых объектов платформы — объектов метаданных (справочники, документы, регистры, и т. д.).

> Для реализации алгоритмов конфигурации, разрабатываемой в рамках Платформы, применяется встроенный язык 1С.

Сам по себе встроенный язык 1С не является строго-типизированным (динамическая типизация) языком программирования, т.е. не требуется явно указывать тип переменно при ее объявлении или инициализации, это отдается на откуп компилятору.

```
Переменная1 = Переменная2;
Стр = "апываыв";
ПерЧ = 123;
ПерБ = Истина;
ПерД = Дата (2025, 02, 10) // конструктор
Объект = Справочники.Покупатели.НайтиПоНаименованию("Иванов Иван");
```

Примитивные типы данных встроенного языка:

- Число;

- Строка;
- Дата (дата, время, дата и время);
- Булево;
- Неопределённо - NULL в переменных;
- NULL -- null в базах данных.

> Однако при помощи функций `ТипЗнч(...)` `Тип(...)` можно проверять тип переменных. Делается этого для того исключить попадание значения такого типа, который не может быть обработан в конкретной функции.

Функции и процедуры в языке программирования 1С

В отличие от языка программирования С, где нет различий между функцией и процедурой, в языке 1С "функция", которая ничего не возвращает будет называться "Процедура" (т.е. не используется оператор `Возврат выражение`). В противном случае конструкция будет называться традиционно как "Функция".

> В Процедуре может использоваться оператор `Возврат`, но без выражения. Это сделано для экстренного выхода из Процедуры по некоторому условию.

```
Процедура Тест (Пер1, Знач Пер2, Пер3 = Ложь)
    Если Пер <> Истина Тогда // !=
        Возврат // Выход из процедуры
    КонецЕсли;
КонецПроцедуры; // вместо фигурных скобок
```

```
Функция Тест (Пер1, Знач Пер2, Пер3 = Ложь)
    Количество = 5;
    Возврат Количество; // Можно вызвать как: X = Тест("sdf", Y, Истина);
КонецФункции;
```

> Операторы `Знач` позволяет передать копию передаваемого значения в функцию. Изначально значения передаются по ссылке, т.е. мы можем напрямую изменять передаваемое значение без возврата значения функции.

```
Пер1 = 3;
...
Сообщить(Пер1); // 3
...
Процедура Тест(Пер1)
    Пер1 = 5;
КонецПроцедуры;
...
Сообщить(Пер1); // 5
```

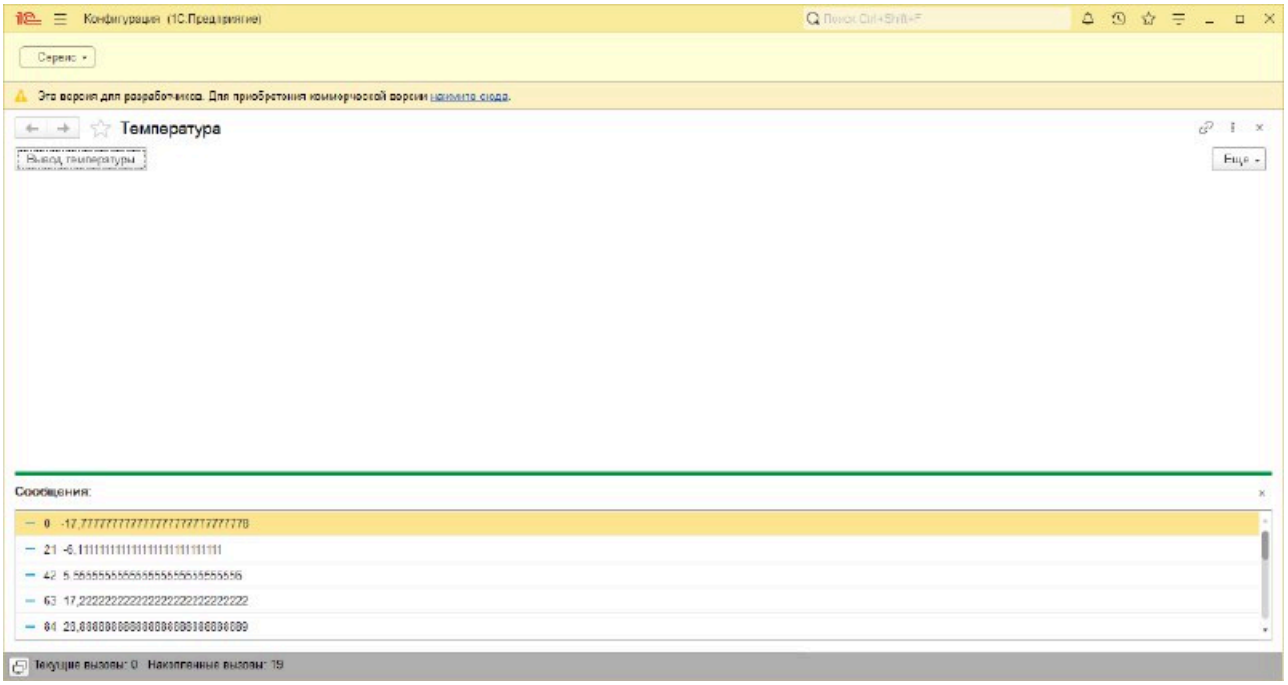
```
Пер1 = 3;
...
Сообщить(Пер1); // 3
...
Функция Тест(Пер1)
    Пер1 = 5;
КонецФункции;
...
Пер1 = Тест(Пер1);
Сообщить(Пер1); // 5
```

> Если рассматривать передачу параметров по ссылке и по значению более внимательно, то при передаче по значения по ссылке, мы работаем в первую очередь с адресом переменной (т.е. где физически

находится значение переменной), что и позволяет ее изменять. При копировании происходит создание новой-временной переменной по новому адресу, но со схожим значением. В некотором роде передача по значению является механизмом защиты данных, но при этом работает медленнее (т.к. выполняется инициализация новой переменной, обращение к адресу значения старой переменной, копирование, и т.д.).

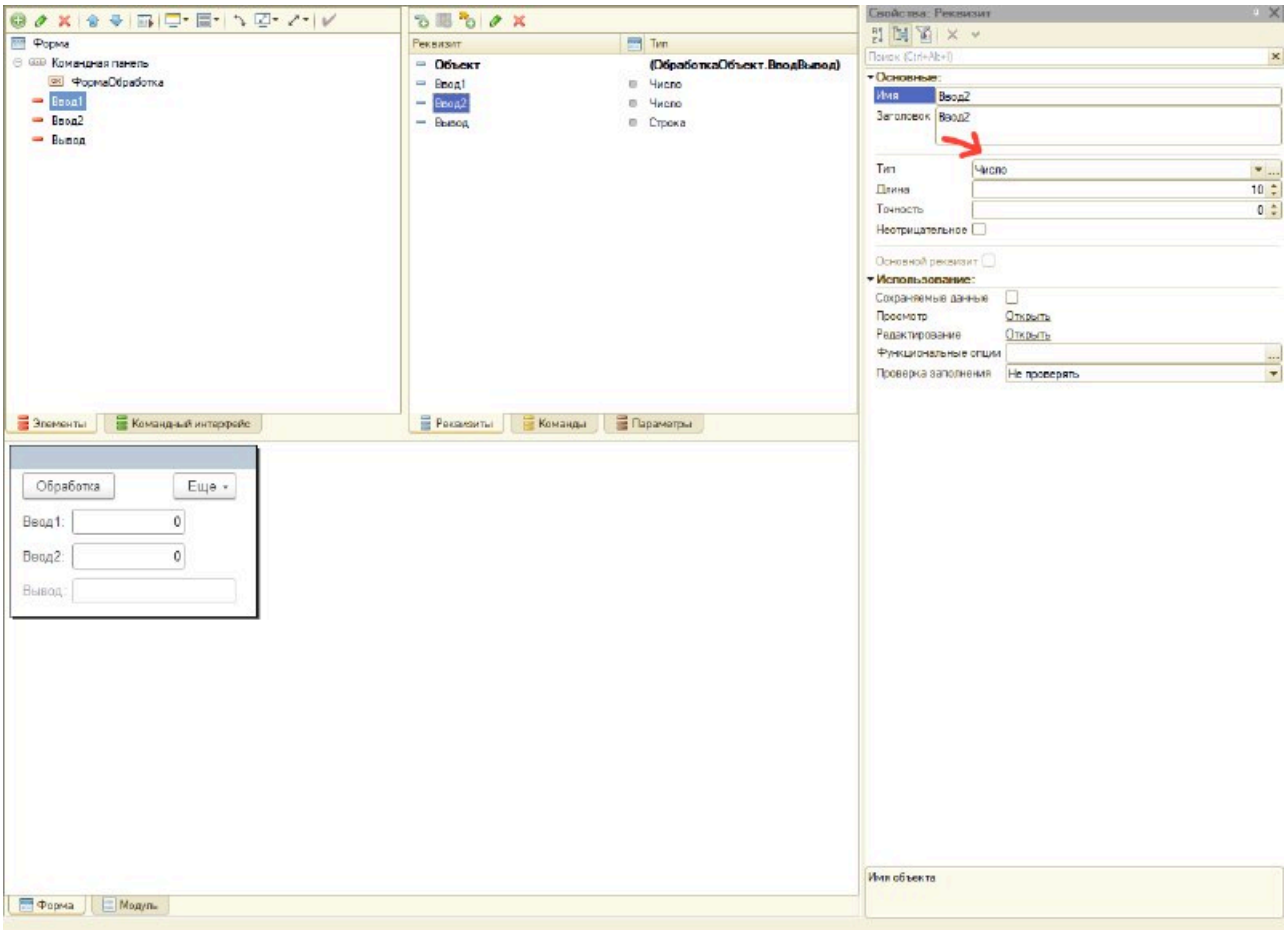
Теперь давайте попробуем переписать раннее написанные программы, но на 1С

```
&НаКлиенте
Процедура ВыводТемпературы(Команда)
    Перем Фар;
    Перем Цельс;

    //Начало = 0;
    //Конец = 300;
    //Шаг = 20;
    //
    //Фар = Начало;
    //
    //Пока фар <= Конец Цикл
    //  Цельс = 5 * (Фар - 32) / 9;           ;
    //  Сообщить(Строка(Фар) + "    " + Строка(Цельс));
    //  Фар = Фар + Шаг;
    //КонецЦикла;

    // Сокращенный вариант программы
    Шаг = 20;
    Для Фар = 0 По 300 Цикл
        Цельс = 5 * (Фар - 32) / 9;           ;
        Сообщить(Строка(Фар) + "    " + Строка(Цельс));
        Фар = Фар + Шаг;
    КонецЦикла
КонецПроцедуры
```



Ввод и вывод данных

В качестве последнего примера в лекции напишем обработку ввода-вывода данных. У нас есть два поля для ввода и одно для вывода. Т.к. мы хотим, чтобы в поля ввода можно было записать только числа, необходимо в свойствах реквизитов указать тип `Число` , чтобы избежать конфликтов или лишнего кода для извлечения цифр. Тип реквизита для поля вывода оставим строкой.

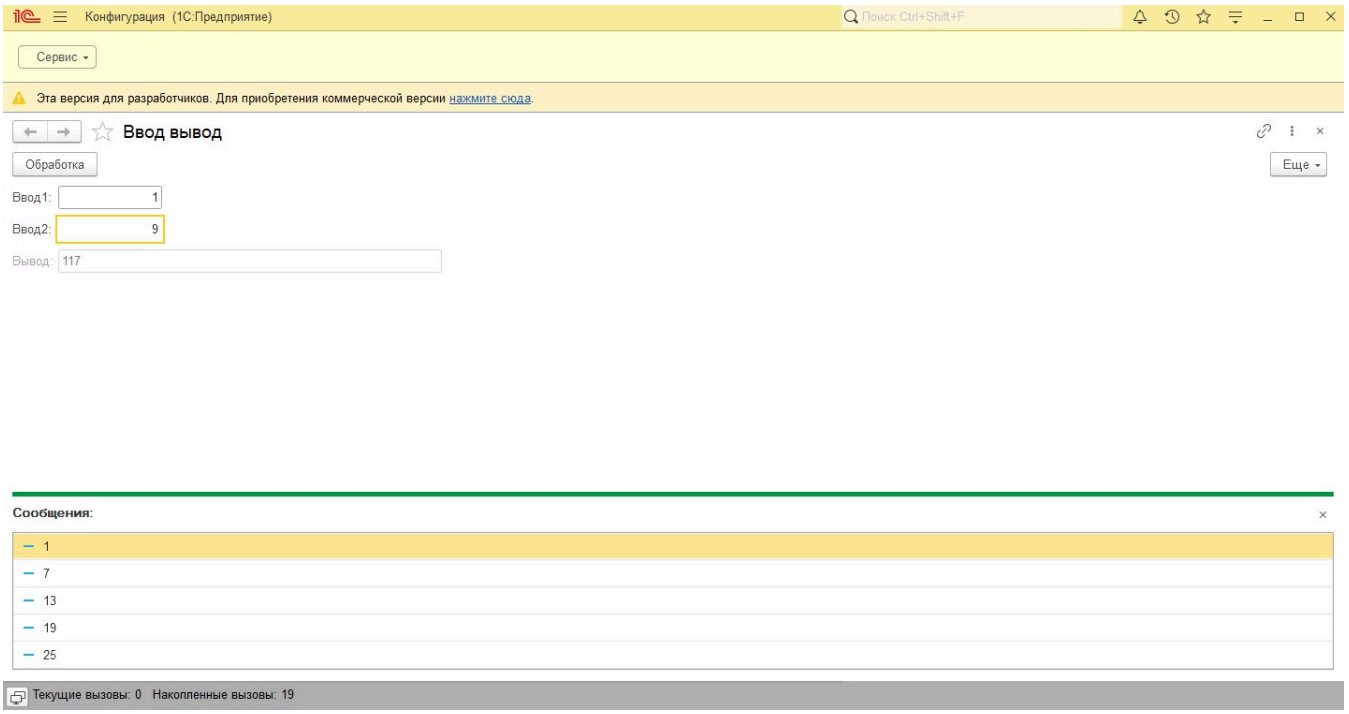


В качестве логики обработчика мы будем высчитывать и выводить арифметическую прогрессию, начиная со значения из поля `Ввод1` (a1) и заканчиваем N членом, который мы задаим в поле `Ввод2`. В поле `Вывод` запишем сумму нашей арифметической прогрессии.

```
&НаКлиенте
Процедура Обработка(Команда)
    Перем an;
    Разность = 3;

    // Ввод2 -- порядковый номер члена арифметической прогрессии
    Для n = 1 по Ввод2 Цикл
        an = Ввод1 + (n - 1) * Разность;
        Если an % 2 = 0 Тогда
            Продолжить; //Прервать;
        КонецЕсли;
        Сообщить(an);
    КонецЦикла;
    Вывод = Строка((Ввод1 + an) * ввод2/2);
КонецПроцедуры
```

> Дополнительно мы накладываем условие для демонстрации работы условной конструкции и операторов `Продолжить` (или `Прервать`). Если an-ое число является четным, то мы его не выводим.

> При вычислении суммы, должно учитываться правильное an-ое число.


Конфигурация (1С:Предприятие)
Поиск Ctrl+Shift+F
Сервис
Эта версия для разработчиков. Для приобретения коммерческой версии нажмите сюда.
Ввод вывод
Обработка
Еще
Ввод1: 1
Ввод2: 9
Вывод: 117

Сообщения:
1
7
13
19
25
Текущие вызовы: 0 Накопленные вызовы: 19