Первое занятие
“Изучение языка С”

Для начала вы узнаете:

1. Историю развития языка;
2. Чем язык **С** отличается от других языков программирования;
3. Что такое алгоритм и блок-схема;
4. Создадите простейшую программу;
5. Узнаете, зачем необходимо использовать ESCAPE-последовательности.

История происхождения языка С

1. Кем и когда был создан данный язык ?
2. Для каких целей он был создан ?
3. Почему мы начинаем свое обучения, с изучения именно данного языка ?

Деннис Ритчи из компании **Bell Labs** создал язык программирования **C** в **1972** году во время работы над созданием операционной системы Unix (прообраз современных **UNIX** систем, таких как **Android** и прочих linux подобных систем), его предшественником был язык программирования B созданный **Кеном Томпсоном**.

Изначально язык программирования **C** задумывался, как инструментальное средство для программистов-практиков и его главной целью было создание полезного языка программирования.

Данный язык является родителем для таких знаменитых и востребованных в современном обществе языков программирования как **C++, Objective C, C#, Java**. Его синтаксис послужил основой для данных языков. По этому все выше перечисленные языки называются **C** подобными.

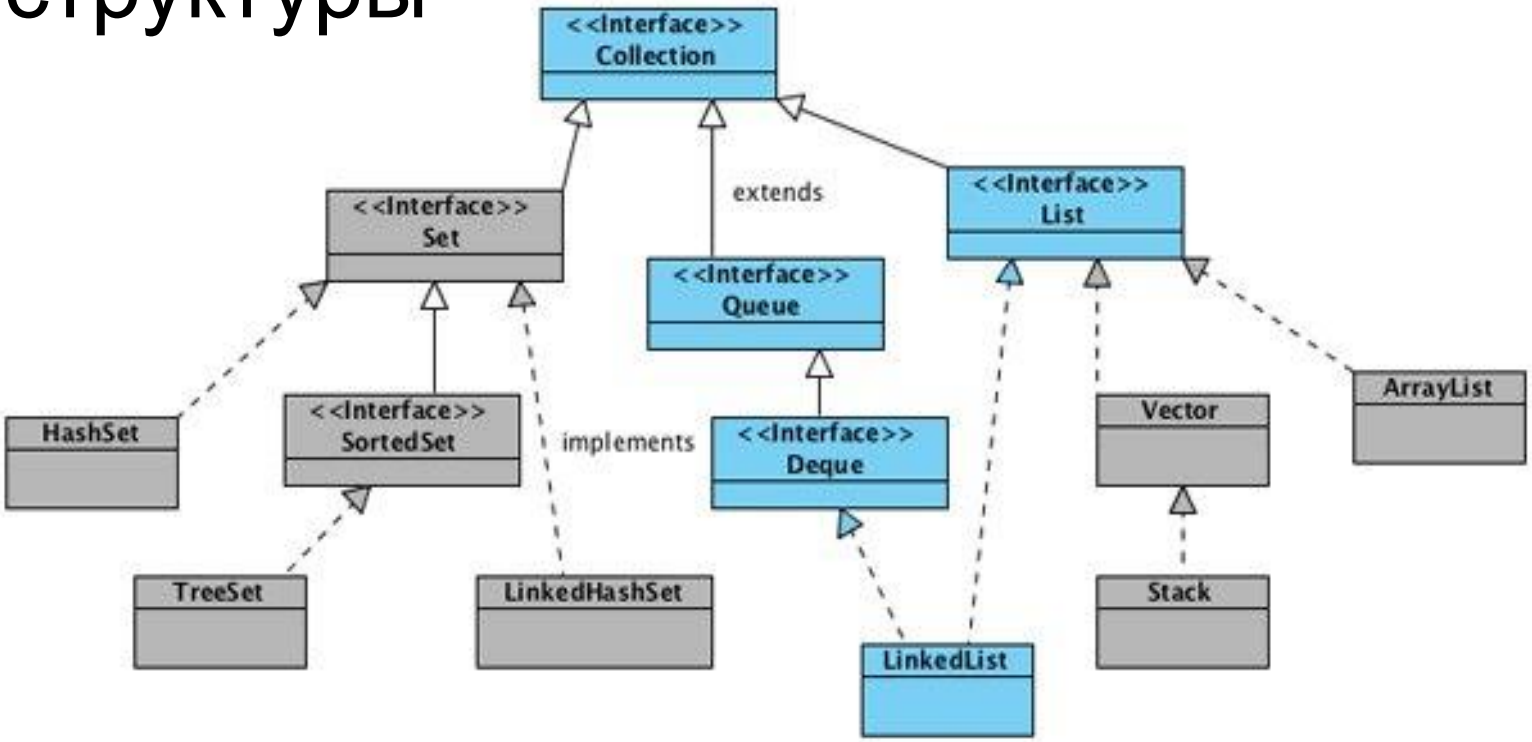
В течение трех последних десятилетий **C** стал одним из основных и наиболее широко распространенных языков программирования. Его популярность росла в связи с тем, что разные люди предпринимали попытки работать с ним, в то время когда он показывал себя с лучшей стороны.

Почему именно C ?

Мы начинаем наше обучение именно с языка **C,** потому что он представляет собой отличную основу, для любого начинающего программиста, после изучения **C**, изучение всех последующих языков уже не вызовет никаких проблем, в независимости от того какой язык программирования вы для себя выберите. Неважно решите ли вы разрабатывать приложения для операционной системы **Android** на **Java**, **WEB** и **WPF** приложения для операционной системы **Windows** на языке **C#**, или мобильные и настольные приложения для **IOS** на **Objective-C** . Все это вам будет гораздо легче освоить, зная язык программирования **C**.

По мере изучения языка , вы убедитесь в том, что он обладает многими достоинствами, с некоторые из которых мы ознакомимся прямо сейчас:

Мощные управляющие структуры



C представляет собой современный язык программирования, включающий управляющие средства. Его конструкция хорошо подходит для планирования сверху вниз, для структурного программирования и для модульного проектирования. Все это позволяет получать надежные и понятные программы.

Быстродействие

C является эффективным языком программирования. Его конструкция продуктивно использует возможности компьютеров, на которых он установлен. Программы на **C** отличаются компактностью и быстротой исполнения.

Компактный программный код

За счет прямой компиляции языка **C** в программный код, программы написанные на нем занимают меньшее количество место, чем программы написанные на аналогичных языках программирования.

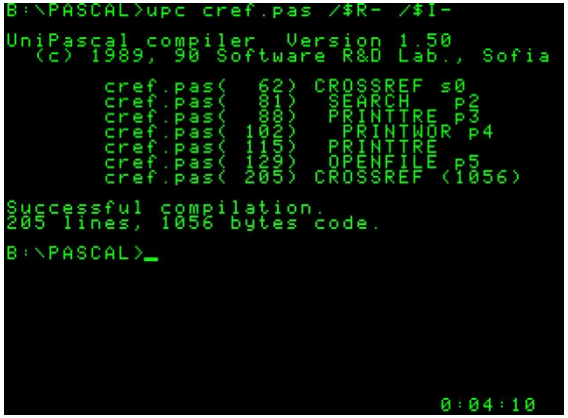
Переносимость на другие компьютеры

Язык **C** является переносимым языком, это означает, что программу, написанную на **C** для одной системы, можно выполнять на другой системе всего лишь с небольшими изменениями, причем иногда удается вообще обходиться без изменений. Компиляторы языка **C** доступны примерно для 40 систем, от 8-разрядных микропроцессоров до суперкомпьютеров Cray.

Недостатки языка C

Язык **C** не лишен недостатков, одним из них является возможность напрямую взаимодействовать с памятью используемой приложением, при помощи указателей (о том что такое указатели мы рассмотрим на последующих занятиях), таким образом у разработчика появляется возможность допустить ошибки, возникновение которых будет очень сложно отследить в будущем. Один из известных людей перефразировал данный комментарий следующим образом: ценой свободы является постоянная бдительность. У языка **C** есть и другие недостатки, но рассматривать их все мы сегодня не будем.

Сферы применения языка С



Создание компиляторов

Разработка операционных

Разработка приложений для PC.

Программирования микропроцессоров.

Что такое алгоритм

Алгоритм - набор инструкций, описывающих порядок действий исполнителя для достижения некоторого результата. В старой трактовке вместо слова «порядок» использовалось слово «последовательность», но по мере развития параллельности в работе компьютеров слово «последовательность» стали заменять более общим словом «порядок».

В качестве некого **алгоритма** можно рассмотреть любое действие, будь то поход в кино, снятие денег со счета, звонок другу и т.д.

В программировании **алгоритм** обозначает последовательность выполнения программы (программный алгоритм), как правило, он представлен в виде цепочки операций.

Пример алгоритма в программировании:

```
int main()
{
    setlocale(LC_ALL, "rus");
    int a = 0;
    int b = 1;
    int summ = Add(10, 12);
    if (a + b<summ)
    {
        printf("Hellow World");
    }
    system("pause");
    return 0;
}
```

Данный пример иллюстрирует алгоритм вывода сообщения на Console;

Блок схемы

Блок схема - распространенный тип схем, описывающих **алгоритмы** или процессы, в которых отдельные шаги изображаются в виде блоков различной формы, соединенных между собой линиями, указывающими направление последовательности. Правила выполнения регламентируются ГОСТ 19.701-90 "Схемы алгоритмов, программ, данных и систем. Условные обозначения и правила выполнения". Стандарт в частности регулирует способы построения схем и внешний вид их элементов.

Основные элементы схем алгоритмов

Процесс:

Начертание

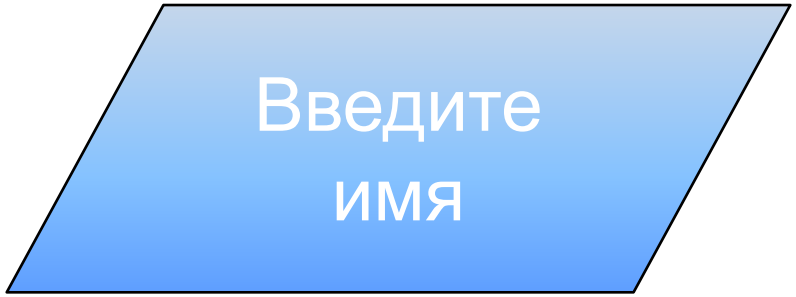
Пример

Символ отображает функцию обработки данных любого вида.

Данные

Начертание

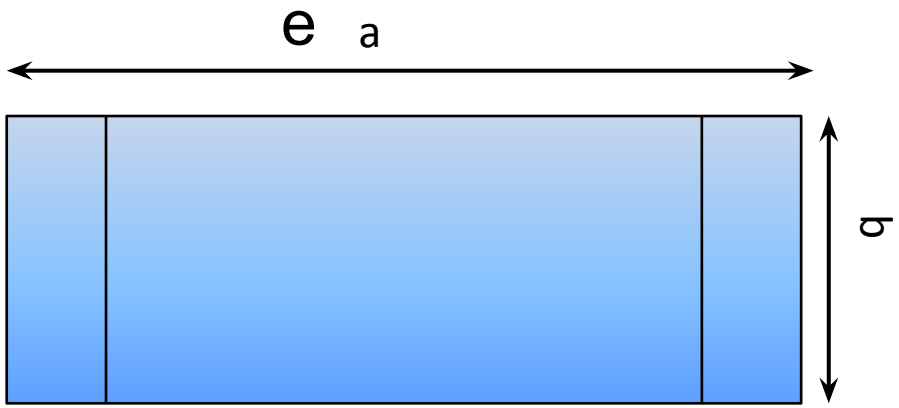
Пример



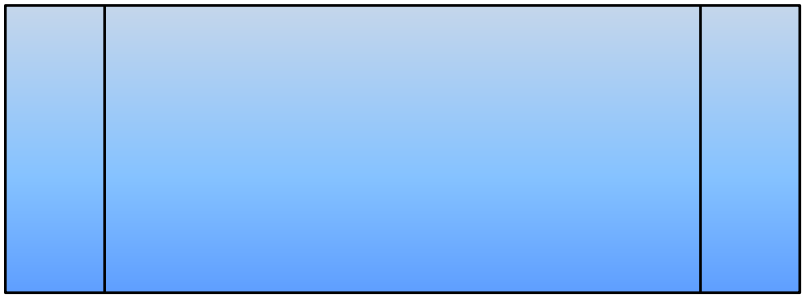
Символ отображает данные, носитель данных не определен.

Переопределенный процесс

Начертание

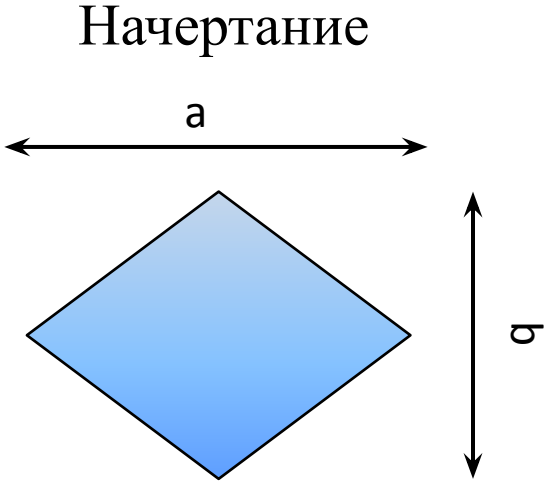


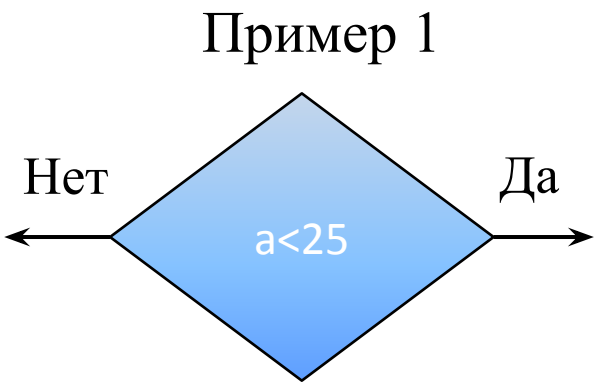
Пример

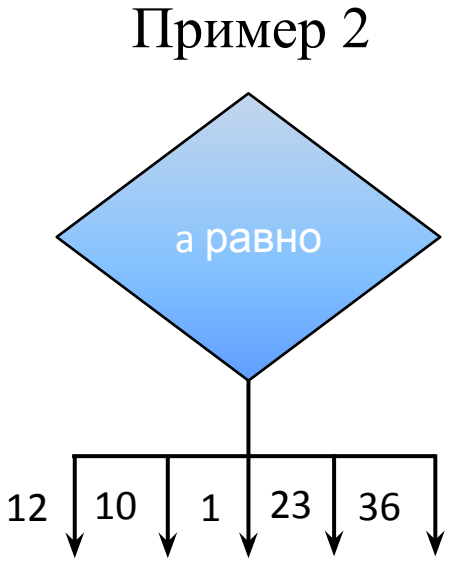


Символ отображает предопределенный процесс, состоящий из одной или нескольких операций или шагов программы, которые определены в другом месте (в подпрограмме, модуле). Например, в программировании – вызов процедуры или функции.

Решение







Отображает решение или функцию переключательного типа с одним входом и двумя или более альтернативными выходами, из которых только один может быть выбран после вычисления условий, определенных внутри этого элемента.

Терминатор

Начертание

Пример

Символ отображает выход во внешнюю среду и вход из внешней среды (начало или конец схемы программы, внешнее использование и источник или пункт назначения данных)