

# Алгоритмизация и программирование

**Автор –  
Комлева Нина  
Викторовна,**  
доцент Базовой  
кафедры цифровой  
экономики  
института развития  
информационного  
общества



# Цель и задачи дисциплины

Дисциплина «**Алгоритмизация и программирование**» является базовой для широкого спектра дисциплин связанных с программированием

## **Основные цели и задачи изучения дисциплины:**

- знакомство с классификацией программного обеспечения, информационных систем и информационных технологий
- получение теоретических знаний и навыков создания алгоритмов
- понимание принципов функционирования программного обеспечения
- изучение языка программирования C/C++
- приобретение опыта работы в современной среде программирования
- приобретение навыков создания программ для ЭВМ
- приобретение теоретических знаний и опыта работы с динамическими структурами данных.

# Виды учебной работы по дисциплине

- Лекции
  - Лабораторные занятия
  - Практические занятия
  - Самостоятельная работа
  - **Курсовая работа**
- 

**Промежуточная аттестация -экзамен**

# Лекция 1. Программное обеспечение, информационные технологии, алгоритмизация.

## Язык программирования C++

### План лекции:

- Программное обеспечение и информационные технологии
- Алгоритмизация вычислительных процессов и процессов обработки данных
- Развитие языков программирования. Инструментальные средства разработки программ (на примере MS Visual Studio). Общая характеристика языка программирования C++. Онлайн компиляторы C++

# **ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ И ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ**

# Основные определения

- **Архитектура** – это наиболее общие принципы построения ЭВМ, реализующие программное управление работой и взаимодействие основных её функциональных узлов.

# Поколения ЭВМ

- ***1 поколение (1944–1958).***

Ламповые машины с быстродействием порядка 10–20 тыс. операций в секунду, программы писались на машинном языке

- ***2 поколение (1959 – 1963).***

Полупроводниковые машины на транзисторах. Быстродействие 100 тыс. операций в секунду. Имеются программы перевода с алгоритмических языков на машинный язык. Есть набор стандартных программ.

# Поколения ЭВМ

- **3 поколение (1964 – 1970).** Миникомпьютеры на интегральных схемах. Отличаются большей надежностью и малыми размерами. Быстродействие 10 млн. оп/с. Образуют системы программно-совместимых устройств
- **4 поколение (1971 – наши дни).** Вычислительные системы на больших интегральных схемах (БИС). Имеют большой объем памяти, позволяют подключать большое количество устройств ввода и вывода информации. Микропроцессор, разработанный в 1971 году, позволил создать центральный процессор на одном чипе
- **5 поколение (1982- наши дни)** - это правительственная программа Японии по развитию вычислительной техники и искусственного интеллекта



# Программное обеспечение и информационные технологии

Под **программным обеспечением** понимается совокупность программ и документации на них, предназначенных для реализации целей и задач. Существует несколько определений программного обеспечения.

- ***Программное обеспечение*** — это совокупность программ системы обработки информации и программных документов, необходимых для эксплуатации этих программ
- ***Программное обеспечение*** — это совокупность программ, процедур и правил, а также документации, относящихся к функционированию системы обработки данных

# Программное обеспечение и информационные технологии

Программное обеспечение (ПО) в соответствии с выполняемыми функциями (по назначению) делится на **системное, прикладное и инструментальное** программное обеспечение

# Программное обеспечение и информационные технологии

- ***Системное ПО*** – это программное обеспечение, включающее в себя операционные системы, сетевое ПО, сервисные программы, а также средства разработки программ (трансляторы, редакторы связей, отладчики и пр.).

# Системное ПО

- **Операционная система** - это совокупность программных средств, которые обеспечивают управление аппаратными ресурсами вычислительной системы и взаимодействие программных процессов с аппаратурой, другими процессами и пользователем. Это комплекс программ, входящих в состав программного обеспечения компьютера, обеспечивающих управление работой аппаратных средств компьютера, обменом данными между различными аппаратными узлами ПК, а также организующих диалог компьютера с пользователем.

# Системное ПО

- **Сетевое ПО** предназначено для управления общими ресурсами в распределенных вычислительных системах
- **Средства разработки программ** используются для разработки нового программного обеспечения, как системного, так и прикладного

# Прикладное ПО

- ***Прикладное ПО*** – это программное обеспечение, предназначенное для решения определенной целевой задачи из проблемной области. Часто такие программы называют приложениями.

# Программное обеспечение и информационные технологии

- ***Инструментальное ПО*** - программные средства, которые могут оказать помощь на всех стадиях разработки ПО. К ним могут относиться, например, средства разработки программного обеспечения, среда разработки, система управления базами данных.

# Программное обеспечение и информационные технологии

- ***Информационная система*** — это взаимосвязанная совокупность средств, методов и персонала, используемых для хранения, обработки и выдачи информации для достижения цели управления. В современных условиях основным техническим средством обработки информации является персональный компьютер. Большинство современных информационных систем преобразуют не информацию, а данные. Поэтому часто их называют системами обработки данных.



# Программное обеспечение и информационные технологии

- **Принцип интеграции**, заключающийся в том, что обрабатываемые данные, однажды введенные в систему, многократно используются для решения большого числа задач
- **Принцип системности**, заключающийся в обработке данных в различных аспектах, чтобы получить информацию, необходимую для принятия решений на всех уровнях управления
- **Принцип комплексности**, заключающийся в механизации и автоматизации процедур преобразования данных на всех этапах функционирования информационной системы

# Программное обеспечение и информационные технологии

Информационные системы также классифицируются:

- по функциональному назначению: *производственные, коммерческие, финансовые, маркетинговые и др.*
- по объектам управления: *информационные системы автоматизированного проектирования, управления технологическими процессами, управления предприятием (офисом, фирмой, корпорацией, организацией) и т. п.*

# Программное обеспечение и информационные технологии

- по характеру использования результатной информации: *информационно-поисковые*, предназначенные для сбора, хранения и выдачи информации по запросу пользователя; *информационно-советующие*, предлагающие пользователю определенные рекомендации для принятия решений (системы поддержки принятия решений); *информационно-управляющие*, результатная информация которых непосредственно участвует в формировании управляющих воздействий.

# Программное обеспечение и информационные технологии

Структуру информационных систем составляет совокупность отдельных ее частей, называемых подсистемами.

- **Функциональные подсистемы** реализуют и поддерживают модели, методы и алгоритмы получения управляющей информации. Состав функциональных подсистем весьма разнообразен и зависит от предметной области использования информационной системы, специфики хозяйственной деятельности объекта, управления.

# Программное обеспечение и информационные технологии

В состав ***обеспечивающих подсистем*** обычно входят:

- *информационное обеспечение*
- *техническое обеспечение*
- *программное обеспечение*
- *математическое обеспечение*
- *лингвистическое обеспечение*

# Программное обеспечение и информационные технологии

**Организационные подсистемы** по существу относятся также к обеспечивающим подсистемам, но направлены, в первую очередь, на обеспечение эффективной работы персонала, и поэтому они могут быть выделены отдельно. К ним относятся:

- *кадровое обеспечение*
- *эргономическое обеспечение*
- *правовое обеспечение*
- *организационное обеспечение*

# Программное обеспечение и информационные технологии

- Под **технологией** понимается совокупность методов, способов и приемов, применяемых для получения определенного вида продукции
- Информационные технологии относятся к области информационной деятельности людей
- **Технология** — это точно рассчитанный процесс получения предсказуемого (предопределенного) результата. Это свойство является важнейшей характеристикой технологии, отличающей его от других процессов, например, эксперимента, где результат не может быть предопределенно предсказан.

# Программное обеспечение и информационные технологии

Информационные технологии широко используемые в наши дни:

- *Поиск информации*
- *Помощь в принятии управленческих решений*
- *Управление технологическими процессами*
- *Автоматизированное проектирование*
- *Геоинформационные технологии*
- *Информационные технологии в обучении*



# **АЛГОРИТМИЗАЦИЯ ВЫЧИСЛИТЕЛЬНЫХ ПРОЦЕССОВ И ПРОЦЕССОВ ОБРАБОТКИ ДАННЫХ**

# Основные понятия и определения

- **Алгоритмизация** – это процесс построения алгоритма решения задачи, результатом которого является выделение этапов процесса обработки данных, формальная запись содержания этих этапов и определение порядка их выполнения
- **Алгоритм** (algorithm) – точное предписание, определяющее процесс преобразования исходных данных в конечный результат

# Основные понятия и определения

- **дискретность** – возможность разбиения алгоритма на отдельные элементарные действия;
- **определенность** ( детерминированность ) алгоритма обеспечивает однозначность результата (повторяемость получаемого результата при многократных расчетах с одними и теми же исходными данными ) и исключает возможность искажения или двусмысленного толкования предписания;
- **результативность** – обязательное получение за конечное число шагов некоторого результата, а при невозможности получения результата – сигнала о том, что данный алгоритм неприменим для решения поставленной задачи ;
- **массовость** – возможность получения результата при различных исходных данных для некоторого класса сходных задач.

# Основные понятия и определения

- **Алгоритмический язык** – набор символов и правил образования и истолкования конструкций из этих символов для записи алгоритмов.
- **Программа** (program) – данные, их описание и алгоритм, записанный на языке программирования. Программа описывает операции, которые нужно выполнить для решения поставленной задачи.

# Основные понятия и определения

- **Операторами** называются действия, предписываемые программой, а элементарное предписание, предусматривающее выполнение какой-либо операции, называют командой. Общее название программы определяется, как правило, реализуемой ей задачей (управляющие, ввода/вывода, диагностические и пр.). Обычно программы хранятся во внешней памяти ЭВМ.

# Основные понятия и определения

- **Программирование** (programming) – процесс создания программ. Программирование неразрывно связано с языками программирования.
- **Языки программирования** ( programming language) – формализованные языки для написания программ, исполняемых на ЭВМ. До сих пор язык программирования является искусственным, в нем синтаксис и семантика строго определены.
- **Переменная** – это объект, который в ходе выполнения программы может менять свое значение.

# Основные понятия и определения

- ***Система программирования*** (в настоящее время более употребим термин **стек технологий**) – средство автоматизации программирования, включающее язык программирования, транслятор этого языка, документацию, а также средства подготовки и выполнения программ.
- ***Транслятор*** – это программа, которая переводит с одного языка на другой.
- ***Интерпретатор*** – это программа, которая сразу выполняет переводимые команды.
- ***Компилятор*** – это программа, которая переводит конструкции алгоритмического языка в машинные коды.

# Основные понятия и определения

Формы представления алгоритма

- *словесная*
- *формульно-словесная*
- *псевдокод*
- *графическая*
- *языки программирования*



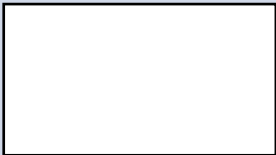
# Основные понятия и определения

- Схема алгоритма представляет собой последовательность блоков, предписывающих выполнение определенных действий, и связи между ними.
- Выделение составных частей алгоритма должно определяться внутренней логикой процесса вычислений.
- Схема алгоритма может выполняться с разной степенью детализации.
- Схема, в которой определены ввод и вывод информации и учитываются особенности языка программирования, называется *схемой программы*.


# Основные понятия и определения

- Графическая запись алгоритма должна выполняться в соответствии с государственными стандартами (ГОСТ 19.701–96 «Государственный стандарт единой системы программной документации» (ЕСПД)).


# Основные понятия и определения

Элемент схемы	Символ	Описание
Процесс		Символ отображает функцию обработки данных любого вида (выполнение определенной операции или группы операций, приводящее к изменению значения, формы или размещения информации или к определению, по которому из нескольких направлений потока следует двигаться).

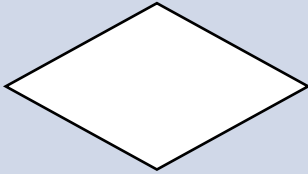
# Основные понятия и определения

Элемент схемы	Символ	Описание
Линии потока		<p>Потоки данных или потоки управления в схемах показываются линиями. Направление потока слева направо и сверху вниз считается стандартным.</p> <p>В случаях, когда необходимо внести большую ясность в схему (например, при соединениях), на линиях используются стрелки. Если поток имеет направление, отличное от стандартного, стрелки должны указывать это направление.</p> <p>Линии в схемах должны подходить к символу либо слева, либо сверху, а исходить либо справа, либо снизу. Линии должны быть направлены к центру символа.</p>


# Основные понятия и определения

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


# Основные понятия и определения

Элемент схемы	Символ	Описание
Решение		Символ отображает решение или функцию переключательного типа, имеющую один вход и ряд альтернативных выходов, один и только один из которых может быть активизирован после вычисления условий, определенных внутри этого символа. Соответствующие результаты вычисления могут быть записаны по соседству с линиями, отображающими эти пути.

# Основные понятия и определения

Элемент схемы	Символ	Описание
<b>Ввод-вывод</b>		Символ отображает данные, носитель данных не определен.

# Основные понятия и определения

Элемент схемы	Символ	Описание
Подготовка		Символ отображает модификацию команды или группы команд с целью воздействия на некоторую последующую функцию (установка переключателя, модификация индексного регистра или инициализация программы).



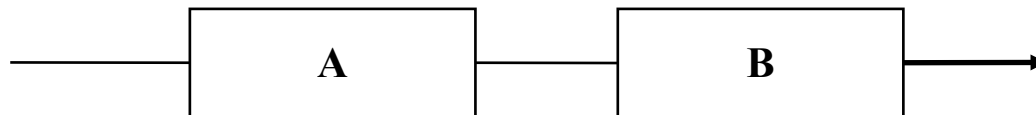
# Базовые структуры программирования

Алгоритм любой сложности может быть представлен комбинацией трех базовых структур:

- следование (линейная);
- ветвление (альтернатива, *если – то – иначе*);
- цикл (повторение).

# Базовые структуры программирования

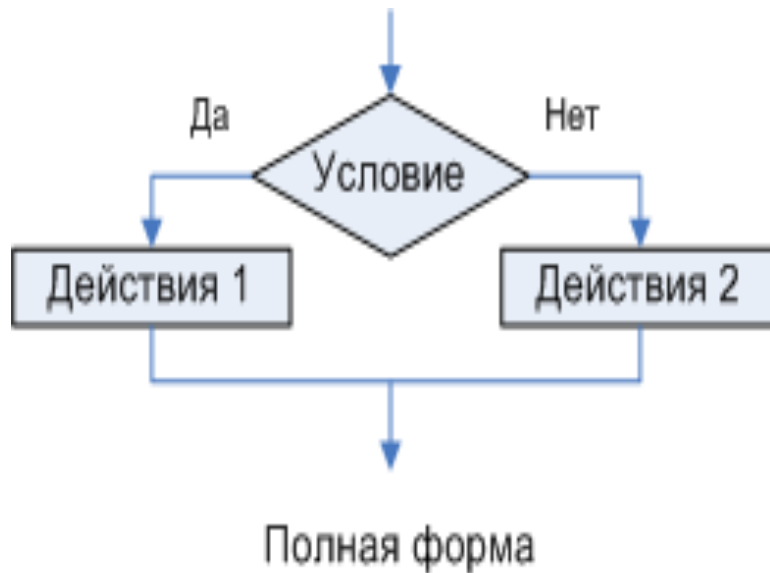
- Базовая структура **следование** означает, что несколько операторов должны быть выполнены последовательно друг за другом и только один раз за время выполнения данной программы.
- Совокупность связанных базовых структур следование называется линейным вычислительным алгоритмом.



# Базовые структуры программирования

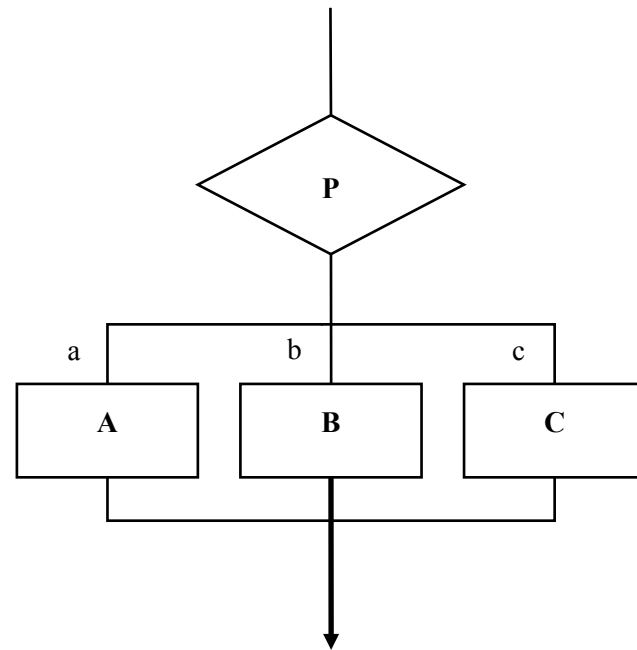
- Второй базовой структурой является **ветвление**. Эта структура обеспечивает, в зависимости от результата проверки условия (истина или ложь), выбор одного из альтернативных путей работы алгоритма, причем каждый из путей ведет к общему выходу. Возможные пути выполнения алгоритма помечают соответствующими метками: истина /ложь, да /нет, 1/0 и т. д.
- Может оказаться, что для одного из выбранных путей действий предпринимать не нужно. Такая структура получила название обход или структура *если – то*.

# Базовые структуры программирования



# Базовые структуры программирования

- Если в алгоритме имеется три и более направления ветвления, то его можно представить в виде совокупности нескольких базовых структур *если – то – иначе*. Такую разновидность структуры разветвление часто называют ***множественный выбор***.



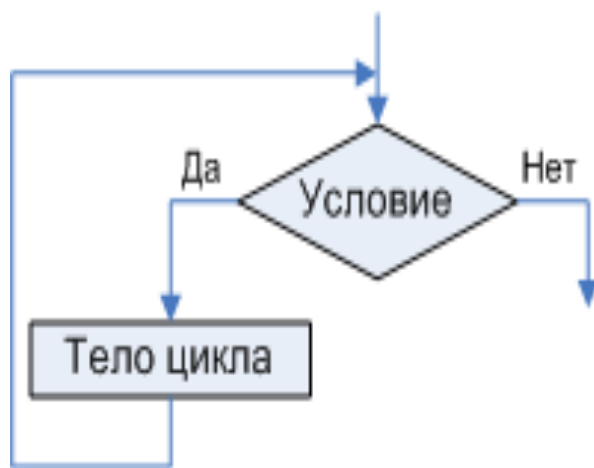
# Базовые структуры программирования

- Третья базовая структура **цикл** обеспечивает повторное выполнение или, другими словами, циклическую работу операторов.
- Различают две разновидности этой структуры: *цикл – пока* (цикл с предусловием) и *цикл – до* (цикл с постусловием).
- Группа операторов, повторяющаяся в цикле, называется телом цикла.
- Основное отличие структуры **цикл – пока** от структуры **цикл – до** заключается в том, что в первой структуре операторы тела цикла в зависимости от условия могут не выполняться совсем, тогда как в структуре *цикл – до* тело цикла будет выполняться хотя бы один раз.

# Базовые структуры программирования

- Легко заметить, что в структуре *цикл – пока* проверка выполнения условия осуществляется перед выполнением операторов тела цикла, а в структуре *цикл – до* осуществляется после прохождения тела цикла.
- *Цикл с параметром* является разновидностью цикла с предусловием (*цикл - пока*). Особенностью данного типа цикла является то, что в нем имеется параметр, начальное значение которого задается в заголовке цикла, там же задается условие продолжения цикла и закон изменения параметра цикла.

# Базовые структуры программирования



Цикл с предусловием



Цикл с постусловием



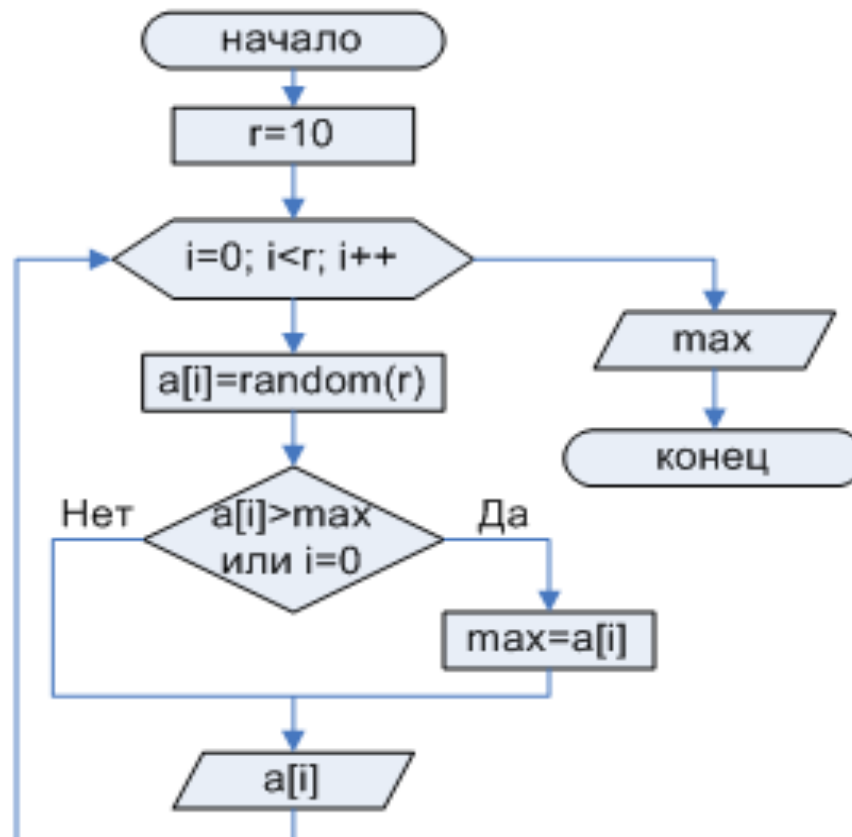
Цикл с параметром



# Примеры базовых алгоритмов

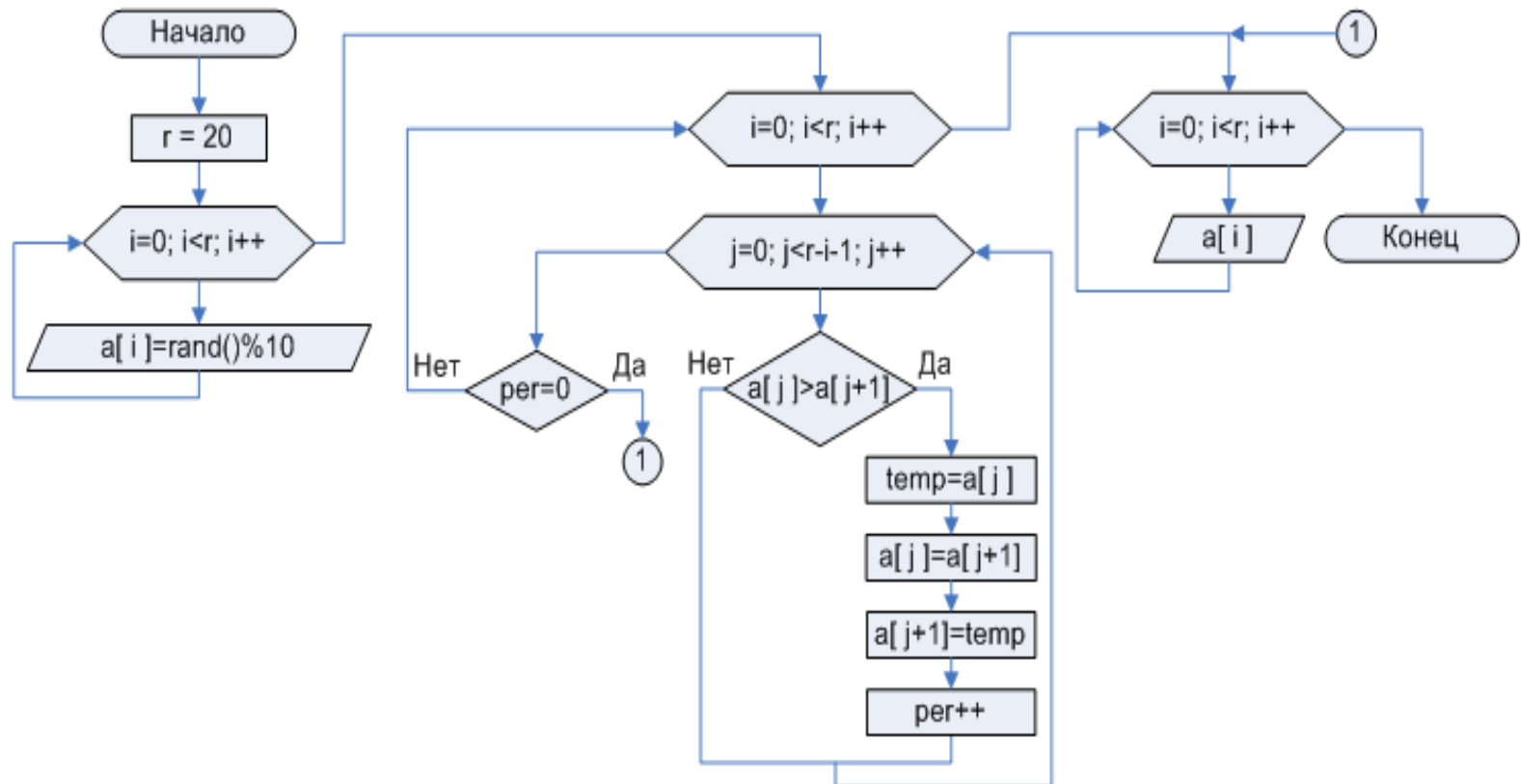
(один из вариантов реализации)

Поиск максимального элемента массива.



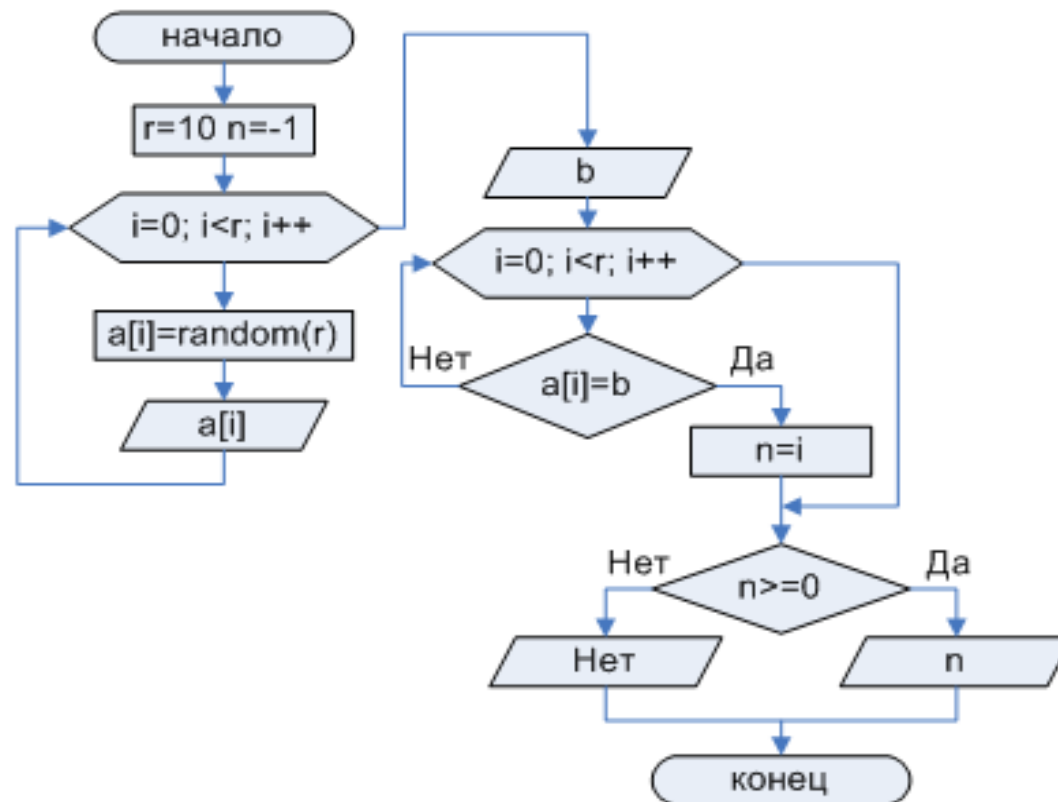
# Примеры базовых алгоритмов

Сортировка одномерного массива по возрастанию методом прямого обмена.



# Примеры базовых алгоритмов

Линейный поиск номера элемента массива с заданным значением.



**ПРОГРАММИРОВАНИЕ.  
ИНСТРУМЕНТАЛЬНЫЕ СРЕДСТВА  
РАЗРАБОТКИ ПРОГРАММ. ОБЩАЯ  
ХАРАКТЕРИСТИКА ЯЗЫКА  
ПРОГРАММИРОВАНИЯ C++**

# Развитие языков программирования

- Первые языки программирования появились в 20-40 годы XX столетия, как и первые ЭВМ, были довольно примитивны и ориентированы на численные расчеты.
- Программы, написанные на ранних языках программирования, представляли собой линейные последовательности элементарных операций с регистрами, в которых хранились данные.

# Развитие языков программирования

- 50-е годы XX века ознаменовались появлением языков программирования так называемого «высокого уровня». Различие состоит в повышении эффективности труда разработчиков за счет абстрагирования или отвлечения от конкретных деталей аппаратного обеспечения
- Одна инструкция (оператор) языка высокого уровня соответствовала последовательности из нескольких низкоуровневых инструкций, или команд.
- **Императивный** подход.

# Развитие языков программирования

- В 60- х г.г. возникает новый подход к программированию, который до сих пор успешно конкурирует с императивным, а именно, **декларативный подход**.
- Суть подхода состоит в том, что программа описывает результат (его свойства), а не методы его достижения.
- Высокая степень абстракции также является преимуществом данного подхода.
- Одним из путей развития декларативного стиля программирования стал **функциональный подход**, возникший с появлением и развитием языка LISP. Такой подход дает возможность прозрачного моделирования текста программ математическими средствами.

# Развитие языков программирования

- В 70-е годы возникла еще одна ветвь языков декларативного программирования, связанная с проектами в области искусственного интеллекта, а именно, языки логического программирования.
- Согласно **логическому подходу** к программированию, программа представляет собой совокупность правил или логических высказываний. Кроме того, в программе допустимы логические причинно-следственные связи, в частности, на основе операции импликации.
- Недостатками логического подхода в концептуальном плане являются специфичность класса решаемых задач и сложность эффективной реализации для принятия решений в реальном времени.



# Развитие языков программирования

- Важным шагом на пути к совершенствованию языков программирования стало появление **объектно-ориентированного подхода** к программированию и соответствующего класса языков.
- В рамках данного подхода программа представляет собой описание объектов, их свойств (или атрибутов), совокупностей (или классов), отношений между ними, способов их взаимодействия и операций над объектами (или методов).
- Наиболее известным примером объектно-ориентированного языка программирования является язык C++, его прямым потомком и логическим продолжением является язык C#.

# Развитие языков программирования

- Развитием событийно управляемой концепции объектно-ориентированного подхода стало появление в 90-х г. г. целого класса языков программирования, которые получили название **языков сценариев** или скриптов. В рамках данного подхода программа представляет собой совокупность возможных сценариев обработки данных, выбор которых инициируется наступлением того или иного события.
- События могут инициироваться как операционной системой (в частности, Windows), так и пользователем.

# Развитие языков программирования

- Еще одним весьма важным классом языков программирования являются ***языки поддержки параллельных вычислений***.
- Программы, написанные на этих языках, представляют собой совокупность описаний процессов, которые могут выполняться как в действительности одновременно, так и в псевдопараллельном режиме.

# Развитие языков программирования

- Другой весьма значимой областью применения языков параллельных вычислений являются системы реального времени, в которых пользователю необходимо получить ответ от системы непосредственно после запроса. Системы такого рода отвечают за жизнеобеспечение и принятие ответственных решений.
- Обратной стороной достоинств рассматриваемого класса языков программирования является высокая стоимость разработки программного обеспечения

# Инструментальные средства разработки программ

- При разработке программного продукта задействуется довольно большой спектр инструментального ПО, которое можно разбить на четыре группы
  - a) необходимые**
  - b) часто используемые**
  - c) специализированные**
  - d) интегрированные среды**

# Инструментальные средства разработки программ

характеристики, универсальные для всех программ:

- фирма-производитель, автор (зачастую имя производителя значит больше, чем все остальное).
- название продукта;
- номер последней версии;
- класс продукта, который установил для него производитель;
- тип дистрибуции программы (с открытыми кодами/бесплатная/условно-бесплатная/платная) и стоимость;
- наличие и тип поддержки, ее стоимость;
- доступность и качество документации;
- простота и понятность интерфейса;
- наличие пробных версий (для платных программ);
- сайт программы и возможность ее скачивания;
- размер дистрибутива и его состав.

# Инструментальные средства разработки программ

- **Microsoft Visual Studio**
- Одним из наиболее передовых инструментальных комплексов для быстрой разработки приложений является интегрированная среда Microsoft Visual Studio.

# Инструментальные средства разработки программ

- Мастер приложений предоставляет пользовательский интерфейс, с помощью которого, на основе шаблона проекта, создается проект, производящий исходные файлы и каталоги для нового приложения. Мастер предоставляет структуру программы, основные меню, панели инструментов, значки и соответствующие операторы **#include**. Мастера приложений Visual C++ работают вместе с исполняющей средой и библиотеками приложений для создания начальных версий программ.



# Инструментальные средства разработки программ

## CLR

- Шаблон "Библиотека классов" (C++);
- Шаблон консольного приложения CLR (C++);
- Шаблон пустого проекта CLR (C++);
- Шаблон библиотеки элементов управления Windows (C++);
- Шаблон "Приложение Windows Forms";
- Шаблон "Служба Windows" (C++).

# Инструментальные средства разработки программ

## **ATL**

- Проект библиотеки ATL.

## **MFC**

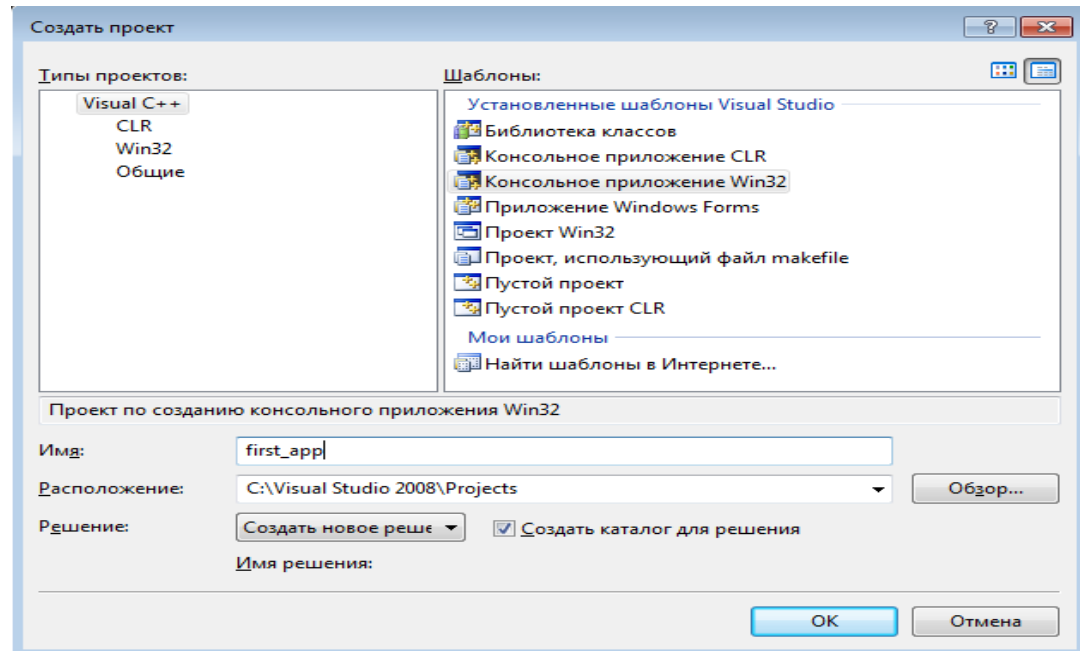
- Элемент управления ActiveX MFC;
- Приложение MFC;
- Библиотека DLL MFC.

## **WIN32**

- Проект консольного приложения Win32;
- Проект Win32.

# Инструментальные средства разработки программ

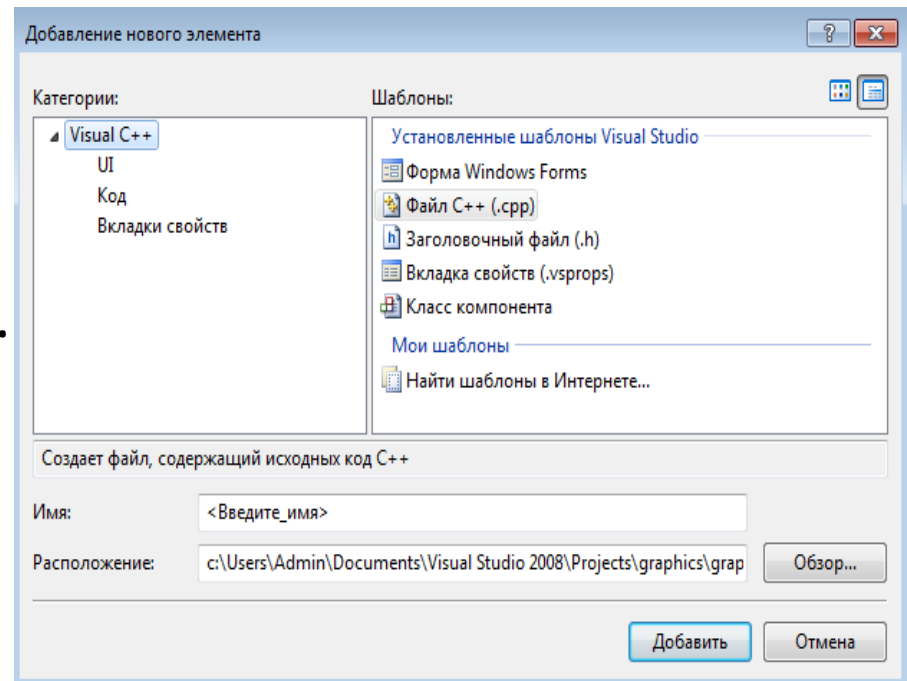
Выбрав пункт меню File | New | Project (Файл | Создать | Проект) вы увидите окно выбора типа проекта. Здесь можно указать основу какого из возможных проектов для вас должна создать среда. При создании проекта необходимо задать его имя.



# Добавление новых файлов в проект

Добавление новых файлов к существующему проекту осуществляется несколькими способами.

- В обозревателе решений по нажатию правой кнопкой на проект и выбором пункта контекстного меню «Добавить» | «Создать элемент».
- Через меню «Проект»  
«Добавить новый элемент».



# Редактор кода

- Редактор — это текстовый процессор интегрированной среды разработки (IDE). При использовании для редактирования текста он называется текстовым редактором. При использовании для редактирования исходного кода на языке разработки Visual Studio — используется термин "редактор кода".
- Можно открыть несколько редакторов кода и просматривать код в различных формах или модулях, а также выполнять копирование и вставлять фрагменты из одного редактора кода в другой.

# Редактор кода

Редактор кода имеет элементы пользовательского интерфейса.

- ***Панель кода***
- ***Поле индикаторов***
- ***Поле структуры***
- ***Горизонтальные и вертикальные полосы прокрутки***

# Редактор кода

Команды редактирования, с помощью комбинаций клавиш:

- ***Ctrl + «-»*** — перемещение, навигация курсора;
- ***Ctrl + Shift + «-»*** — в обратном порядке;
- ***Ctrl + ]*** — перемещение от начала к концу операторных скобок { };
- ***Ctrl + K -> Ctrl + K*** — создать закладку в коде;
- ***Ctrl + K -> Ctrl + N*** — к следующей закладке;
- ***Ctrl + K -> Ctrl + P*** — к предыдущей;
- ***Ctrl + K -> Ctrl + L*** — удалить все закладки;

# Редактор кода

Команды редактирования, с помощью комбинаций клавиш:

- ***Ctrl + G*** — перейти к строке с номером (подсветку номеров строк можно включить в Tools -> Options -> Text Editor -> Select your editor(or All Languages) -> Display -> Line Numbers);
- ***Ctrl + M -> Ctrl + M*** — скрыть и раскрыть код внутри текущего блока;
- ***Ctrl + M -> Ctrl + O*** — скрыть код для всего файла;
- ***Ctrl + M -> Ctrl + L*** — раскрыть код для всего файла;
- ***Ctrl + M -> Ctrl + P*** — не показывать логическую структуру файла ;



# Редактор кода

Команды редактирования, с помощью комбинаций клавиш:

- ***Ctrl + Space*** — вызвать помощник кода (intellisense);
- ***Ctrl + Alt + X*** — показать Toolbox;
- ***Ctrl + Стрелка вверх/вниз*** — удобное перемещение по коду;
- ***Ctrl + K + C*** и ***Ctrl + K + U*** — закомментировать и раскомментировать код;
- ***Ctrl + Shift + Space*** — показать синтаксис метода.

# Компиляция, отладка, запуск

- Интегрированная среда разработки (IDE) Visual Studio облегчает компиляцию проекта и запуск результирующего приложения.
- Для отладки проекта во время его выполнения можно использовать встроенный в IDE отладчик.

## Компиляция и запуск текущего проекта.

- Нажмите клавишу **F5**, находясь в интегрированной среде разработки (IDE) Visual Studio. Среда разработки скомпилирует проект и запустит приложение в отладчике Visual Studio.

# Компиляция, отладка, запуск

## Компиляция и запуск текущего проекта из меню.

- В меню Отладка интегрированной среды разработки Visual Studio выберите команду Начать отладку. Среда разработки скомпилирует проект и запустит приложение в отладчике Visual Studio.

## Компиляция и запуск текущего проекта без отладки.

- Нажмите клавиши CTRL + F5 в среде разработки Visual Studio. Среда разработки скомпилирует проект и запустит приложение.

# Компиляция, отладка, запуск

Компиляция и запуск текущего проекта из меню без отладки.

- В меню Отладка интегрированной среды разработки Visual Studio выберите команду Запуск без отладки. Среда разработки скомпилирует проект и запустит приложение.

Если выбрана команда Запуск, приложение запустится, и будет выполняться до точки останова. Выполнение можно прервать в любой момент, чтобы просмотреть или изменить значения переменных, либо для выполнения других операций, связанных с проверкой работы программы.

# Компиляция, отладка, запуск

При выборе команды «Выполнять до текущей позиции» приложение запустится и будет выполняться либо до точки останова, либо до текущего положения курсора, если точка останова встречена не будет. Положение курсора определяется в окне исходного кода. В некоторых случаях прерывания не происходит. Это означает, что код, на котором стоит курсор, так и не был достигнут в ходе выполнения.

# Общая характеристика языка программирования C++

Язык Си, созданный Денисом Ричи в начале 70-х годов в Bell Laboratory американской корпорации AT&T, является одним из универсальных языков программирования. Язык Си считается языком системного программирования, хотя он удобен и для написания прикладных программ.

# Общая характеристика языка программирования С++

## Преимущества языка Си:

- переносимость программ на компьютеры различной архитектуры и из одной операционной системы в другую
- лаконичность записи алгоритмов
- логическую стройность программ
- возможность получить программный код, сравнимый по скорости выполнения с программами, написанными на языке ассемблера

# Общая характеристика языка программирования C++

- Язык C++ был разработан в начале 1980-х гг. Бьерном Страуструпом из компании AT&T Bell Laboratories. C++ основан на языке Си. Два символа "++" в названии – это игра слов, символами "++" в языке Си обозначается операция инкремента.
- Таким образом, C++ был задуман как язык Си с расширенными возможностями. Большая часть языка Си вошла в C++ как подмножество, поэтому многие программы на Си можно скомпилировать с помощью компилятора C++.
- C++ является расширением языка Си с объектно-ориентированными возможностями.



# Общая характеристика языка программирования C++

- Национальный Институт Стандартизации США (American National Standards Institution, ANSI) разработал "официальные" стандарты для многих языков программирования, в том числе для Си и C++.
- Эти стандарты стали общепринятыми, и они имеют очень большое значение.
- Программу, целиком написанную на **ANSI C++**, гарантированно можно запустить на любом компьютере, для которого имеется компилятор ANSI C++.

# Общая характеристика языка программирования C++

Исходный файл программы на языке Си имеет расширение \*.c или \*.cpp (расширение \*.cpp говорит о том, что в программе могут быть использованы возможности языка C++).

- Это обычный текстовый файл, в который записывают текст программы в любом текстовом редакторе, например, в Блокноте.

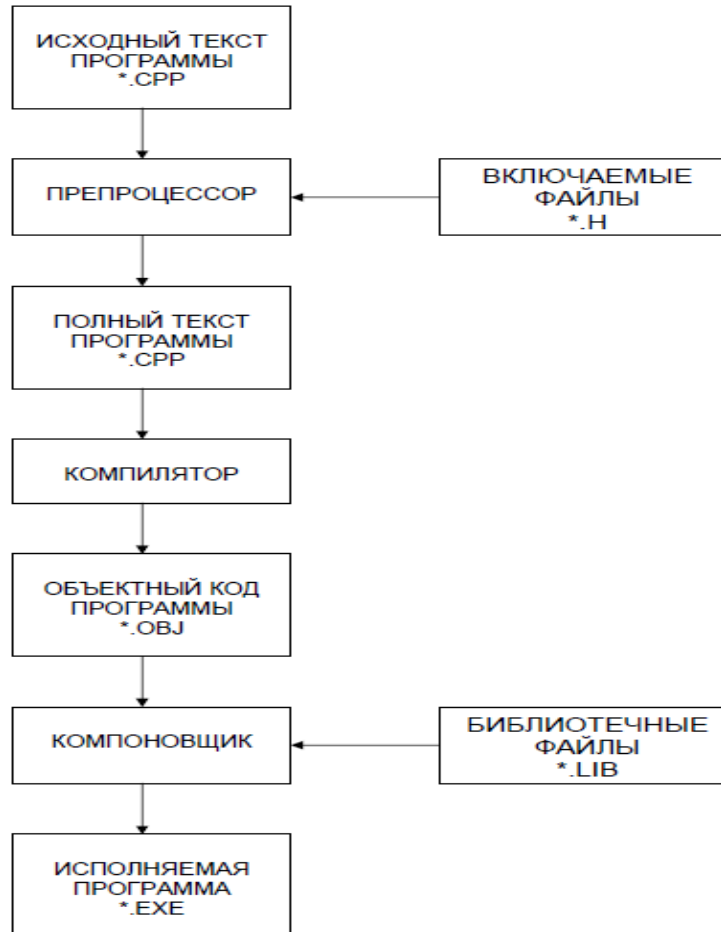
Программа на языке Си создается в два этапа:

- 1) **трансляция** – перевод текста программы в машинные коды;
- 2) **компоновка** – сборка частей программы и подключение стандартных функций.

# Общая характеристика языка программирования C++

- **Транслятор** переводит исходный файл в машинные коды и строит так называемый объектный файл с тем же именем и расширением \*.o. Хотя в нем уже записан машинный код, объектный файл еще нельзя запускать на компьютере, потому что в нем не хватает стандартных функций (например, для ввода и вывода данных).
- **Компоновщик** подключает стандартные функции, хранящиеся в библиотеках (они имеют расширение \*.a). В результате получается один файл с расширением \*.exe, который и представляет собой готовую программу.

# Общая характеристика языка программирования C++



# Общая характеристика языка программирования C++

В алфавит языка входят

- Прописные латинские буквы      A..Z
- Строчные латинские буквы      a..z
- Арабские цифры      0..9
- Символ подчеркивания      \_ (рассматривается как буква)
- Знаки пунктуации и специальные символы, приведенные в таблице на следующем слайде.
- Пробельные символы (пробел, символы табуляции, перевода строки, возврат каретки, перевод страницы). Любая последовательность пробельных символов при компиляции рассматривается как один пробел.

# Общая характеристика языка программирования C++

Символы	Наименование	Символы	Наименование
,	запятая	{	открывающая фигурная скобка
.	точка	}	закрывающая фигурная скобка
;	точка с запятой	<	меньше
:	двоеточие	>	больше
?	знак вопроса	[	открывающая квадратная скобка
'	апостроф	]	закрывающая квадратная скобка
!	восклицательный знак	#	номер или решетка
	прямая черта	%	процент
/	слеш	&	амперсанд
\	обратный слеш	^	НЕ-логическое
~	тильда	-	минус
*	звездочка	=	равенство
(	открывающая скобка	"	кавычки
)	закрывающая скобка	+	плюс

# Общая характеристика языка программирования C++

Все слова (идентификаторы), встречающиеся в программах, можно разделить на три категории:

***Служебные слова языка***

***Библиотечные идентификаторы***

***Идентификаторы, введенные программистом***

# Общая характеристика языка программирования C++

## Служебные слова

<b>asm</b>	<b>continue</b>	<b>float</b>	<b>new</b>	<b>signed</b>	<b>try</b>
<b>auto</b>	<b>default</b>	<b>for</b>	<b>operator</b>	<b>sizeof</b>	<b>typedef</b>
<b>break</b>	<b>delete</b>	<b>friend</b>	<b>private</b>	<b>static</b>	<b>union</b>
<b>case</b>	<b>do</b>	<b>goto</b>	<b>protected</b>	<b>struct</b>	<b>unsigned</b>
<b>catch</b>	<b>double</b>	<b>if</b>	<b>public</b>	<b>switch</b>	<b>virtual</b>
<b>char</b>	<b>else</b>	<b>inline</b>	<b>register</b>	<b>template</b>	<b>void</b>
<b>class</b>	<b>enum</b>	<b>int</b>	<b>return</b>	<b>this</b>	<b>volatile</b>
<b>const</b>	<b>extern</b>	<b>long</b>	<b>short</b>	<b>throw</b>	<b>while</b>



# Общая характеристика языка программирования C++

Идентификатором не может быть произвольная последовательность символов. По правилам Си++, идентификатор начинается с буквы или символа подчеркивания (" \_") и состоит только из английских букв, цифр и символов подчеркивания.

Комментарии:

`/* и */`

`//`

# Структура программы на языке Си

```
#include <stdio.h>  /* заголовок */

int main(void)  /* главная функция: начало программы */
{  /* открывающая скобка в начале программы */

    оператор программы;
    оператор программы;
    ...
    оператор программы;

}  /* закрывающая скобка в конце программы */
```

# Структура программы на языке C++

```
1  #include <iostream>
2
3  using namespace std;
4
5  int main() {
6      int x=10;
7      int y=25;
8      int z=x+y;
9
10     cout<<"Sum of x+y = " << z;
11 }
```

# Общая характеристика языка программирования C++

Для создания удобочитаемой программы следует придерживаться следующих правил:

- располагайте один оператор на строке;
- используйте пустые строки для "отделения" одной логической части программы от другой;
- используйте комментарии.

# Онлайн компиляторы C++

<http://cpp.sh/>

<https://repl.it/languages/cpp>

<https://www.jdoodle.com/online-compiler-c++/>



**Komleva.NV@rea.ru**