ЛЕКЦИЯ 1

Введение в программирование.

Понятие функции

(Слайды 19,20, 39-44 можно пропустить)

Что такое программирование?

Разработка программного обеспечения

дискретная математика, теория вероятности, методы оптимизации, математическая статистика, линейная алгебра, схемотехника и т. д.

Разработка алгоритма и перевод его на ЯП

- Математика
- Английский язык
- Творчество

З н а н и е т е р м и н о л о г и и

Переменная

Тип данных

Объявление
Инициализация

Объект

Класс

наследование,
инкапсуляция,
полиморфизм

Функция

Параметры

Указатели
Ссылки

Что такое программирование?

Разработка программного обеспечения

- Постановка задачи (ТЗ)
- Разработка алгоритма
- Выбор инструментов
- Знание парадигмы ЯП
- Написание кода

- Память ПК
- Архитектура ОС

- Оптимизация кода
- Рациональный подход
- . . .

- Работа в команде
- Работа с документацией
- . . .

Это техническая профессия !!!
Учит решать проблемы методом автоматизации

Английский математик 19 века Шенкс потратил 20 лет жизни на вычисление числа π с точностью 707 верных значащих цифр.

Шенкс ошибся в 520-м знаке!!!

50 лет назад число π было вычислено с точностью 500 тысяч знаков, и потребовалось для этого *несколько часов работы* вычислительной машины серии ЕС.

Д/З. Вычислить число π с точностью 500 и 5000 цифр после запятой. Определить время вычислений на вашей архитектуре и конфигурации компьютера.

Вопрос:

Как определить время работы программы?

Объем информации резко возрос:

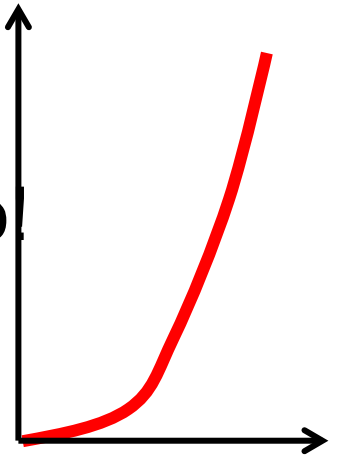
начало н.э. – 1750 годы удвоение знаний

1751 – 1900 годы удвоение знаний

1900 – 1950 годы в 8 – 10 раз

1950 – 2000 годы в 30 раз

сейчас – рост с устрашающей скоростью!



Современная тенденция - годовой рост объемов информации в 2 раза!!!

Информационный взрыв !!!

Экспоненциальный закон!!!

- быстрое сокращение времени удвоения объема накопленных научных знаний;
- огромные затраты на хранение, передачу и переработку информации, а также расходов на энергетику.

В августе 2020 в одном из зарубежных журналов напечатана статья Мелвина Вопсона, ученого из Великобритании, Портсмутский университет, где озвучено, что

- количество битов будет равно количеству атомов на Земле примерно через 150 лет;
- к 2245 году половина массы Земли будет преобразована в цифровую информационную массу;
- когда количество цифровых битов превзойдет количество всех атомов на Земле, мир превратится в суперкомпьютер размером с планету.

Вопсон считает, что

- информацию следует рассматривать как пятое состояние вещества!!!!
- через 130 лет энергия, необходимая для поддержания процесса создания цифровой информации, сравняется со всей энергией, которая производится сейчас на Земле.
- **90% мировой информации, существующей сегодня, было создано за последние десять лет!**

???????

Прогнозы

Объем данных, порождаемых человечеством:
- в 2018 году – 33 зеттабайтов информации
- в 2025 – 175 зеттабайтов (один зеттабайт — это 10^{21} байтов).

- За последние два года человечество сгенерировало больше информации, чем за всю предшествующую историю.
- К концу столетия объем информации в мире составит более четырех йоттабайт (один йоттабайт — это 10^{24} еттабайта).

Количество поисковых запросов, обрабатываемых GOOGLE в секунду:
- в 2018 году – 35 000
- август 2023 – более 100 000 (примерно 9 миллиардов поисковых запросов Google в день).

Должен ли программист знать, как работает ПК, «железо»?

Если программист понимает, как работает компьютер, его программы «правильнее», оптимальнее, лучше……!!!

Если писать программы для решения прикладных задач, использовать компилируемые языки, надо знать как процессор обрабатывает написанный код, как хранятся данные, как выполняются операции над данными, что такое память ПК и т.д.

Что такое программа?

Описание процесса обработки информации на языке программирования.

Неизменяемая информация – программа первоначальной загрузки ПК, базовая система вода/вывода

BIOS – программа, с которой начинает работать компьютер

Центральный процессор

Видеокарта

Клавиатура
Мышь

Жесткий диск

Материнская плата

ПЗУ (BIOS)

Оперативная память

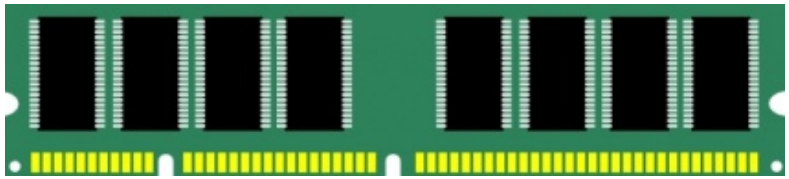
Часы реального времени

Тактовый генератор

Для временного хранения данных

Память компьютера

- это устройства, где компьютер хранит данные.



Оперативная память, в ней хранятся **исполняемые в данный момент** программы (и наша тоже, после того, как мы ее запустили)

Жесткий диск, на нем хранится наша программа **все остальное время**

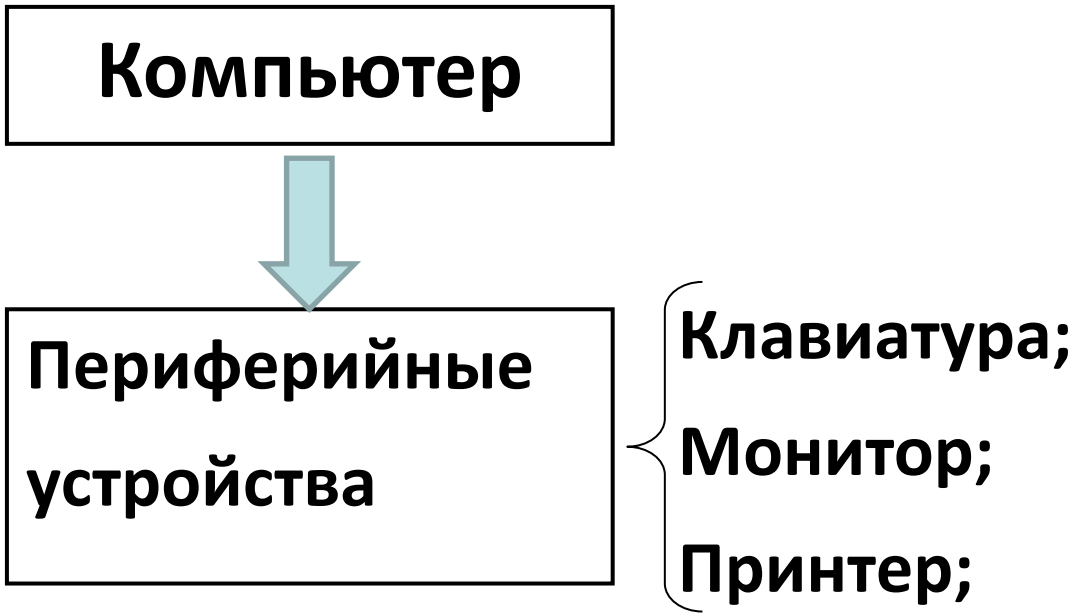
Центральный процессор (CPU)

- это электронное устройство, микросхема для управления всеми операциями ПК и выполнения арифметических и логических операций с данными.

Имеет регистровую, кэш (быструю) память и арифметико-логическое устройство.

Производительность процессора определяется :

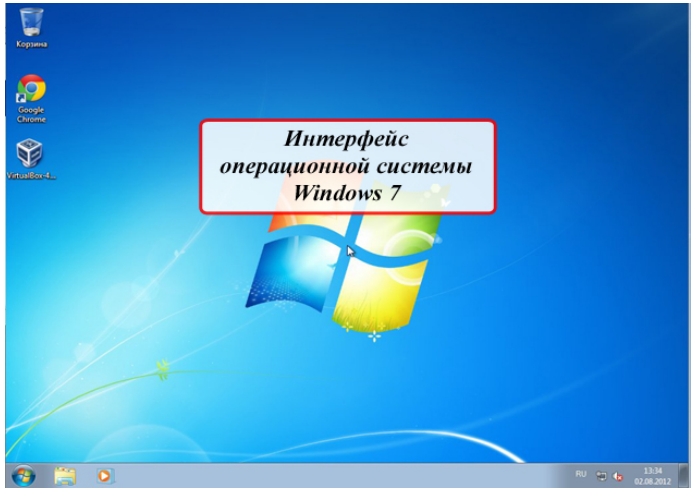
- тактовой частотой – число выполненных операций в секунду (МГц – миллион тактов в сек, ГГц – миллиард тактов в сек);
- разрядностью – количество байт информации, передаваемой за такт (8, 16, 32, 64). Если процессоры оперирует 32-разрядными целыми числами – их называют 32-разрядными процессорами. Стандартный целочисленный тип `int` занимает 32 двоичных разряда.



Драйвера - ?

Драйвер - это программа, обеспечивающая взаимодействие компьютера с оборудованием и устройствами.

Интерфейсы -?



Это совокупность технических, программных средств и правил взаимодействия устройств и программ

Графический интерфейс, интерфейс пользователя, дружеский интерфейс . . .

или сенсорная панель «тачпад»

Программный интерфейс – интерфейсы программ Google, Chrome, Opera.

Консольный интерфейс – взаимодействие с компьютером через ввод — вывод информации. В операционной системе Windows он называется «***Интерфейс командной строки***».

Вводим текст – поэтому ***текстовый интерфейс.***

Интерфейс функции -??? y = f (x);

Интерфейс (объявление, сигнатура) функции, возвращающей значение: ***тип функции, имя функции, типы параметров***

```
float sin(float);
```

Понятие алгоритма, средства записи алгоритмов

Персидский математик IX век н.э. - Абу-Джафар Мохаммед ибн-Мусы аль_Хорезми (**Algorithm**)

50 лет назад определение алгоритма:

Алгоритм – конечная последовательность точно определённых инструкций, приводящих к однозначному решению поставленной задачи.

Сейчас

Алгоритм – конечная совокупность точно заданных инструкций, описывающих порядок действий для решения задачи.

Виды представления алгоритмов:

- словесная запись алгоритмов;
- блок-схемы;
- алгоритмические языки.

Вычислительный блок

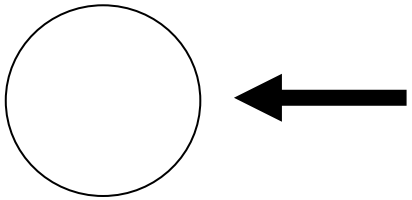
Логический блок

подпрограмма

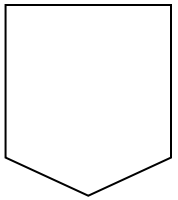
Начало-конец программы

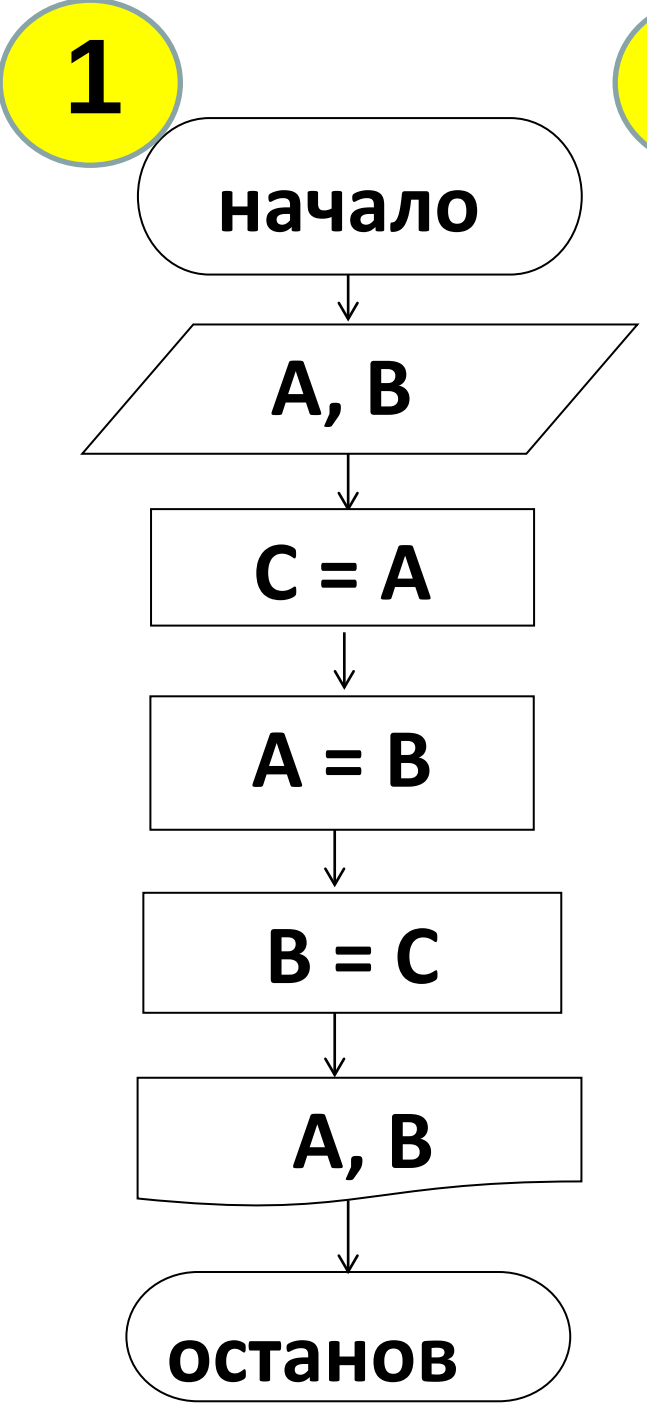
Ввод данных

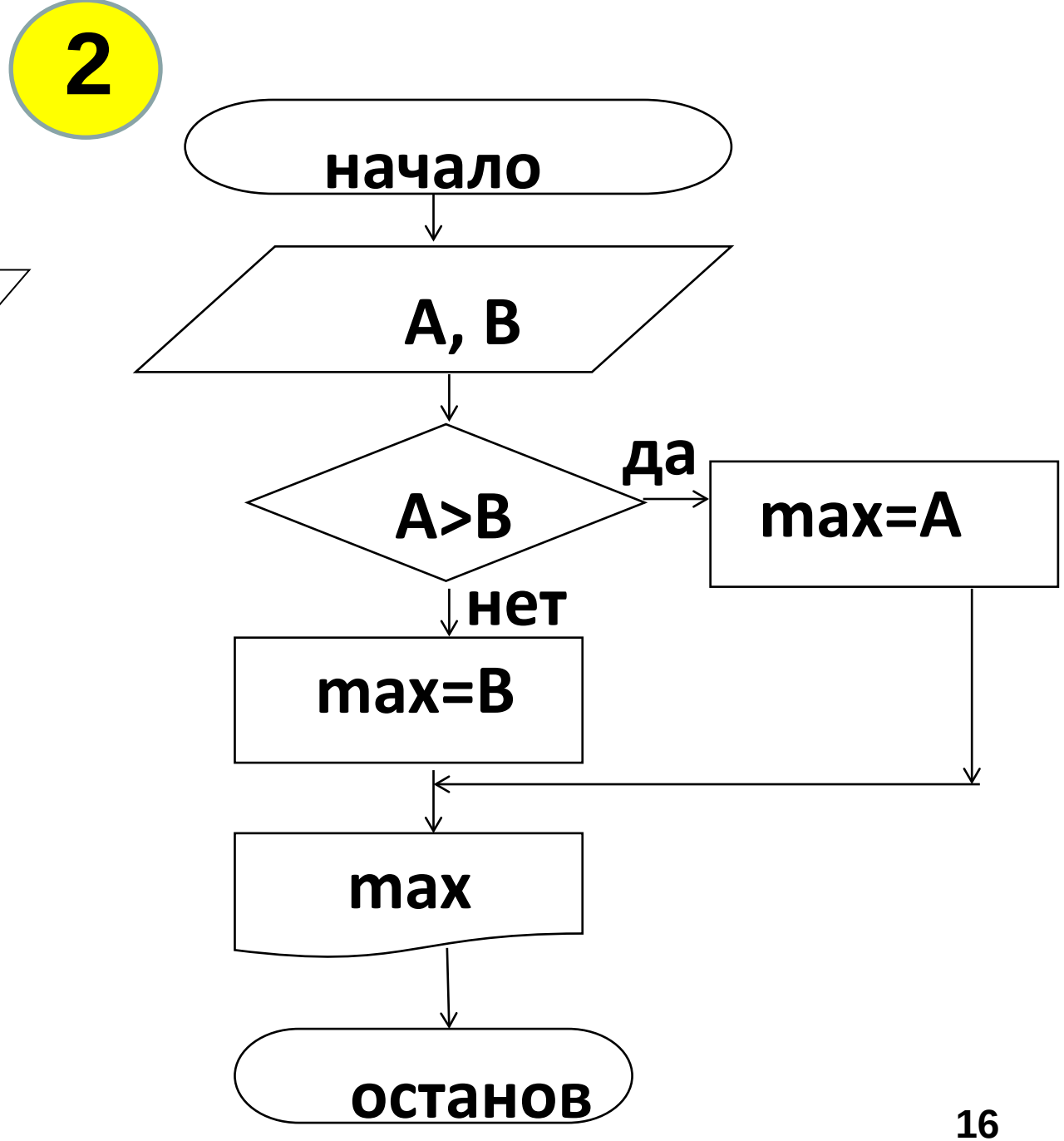
Блок вывода данных



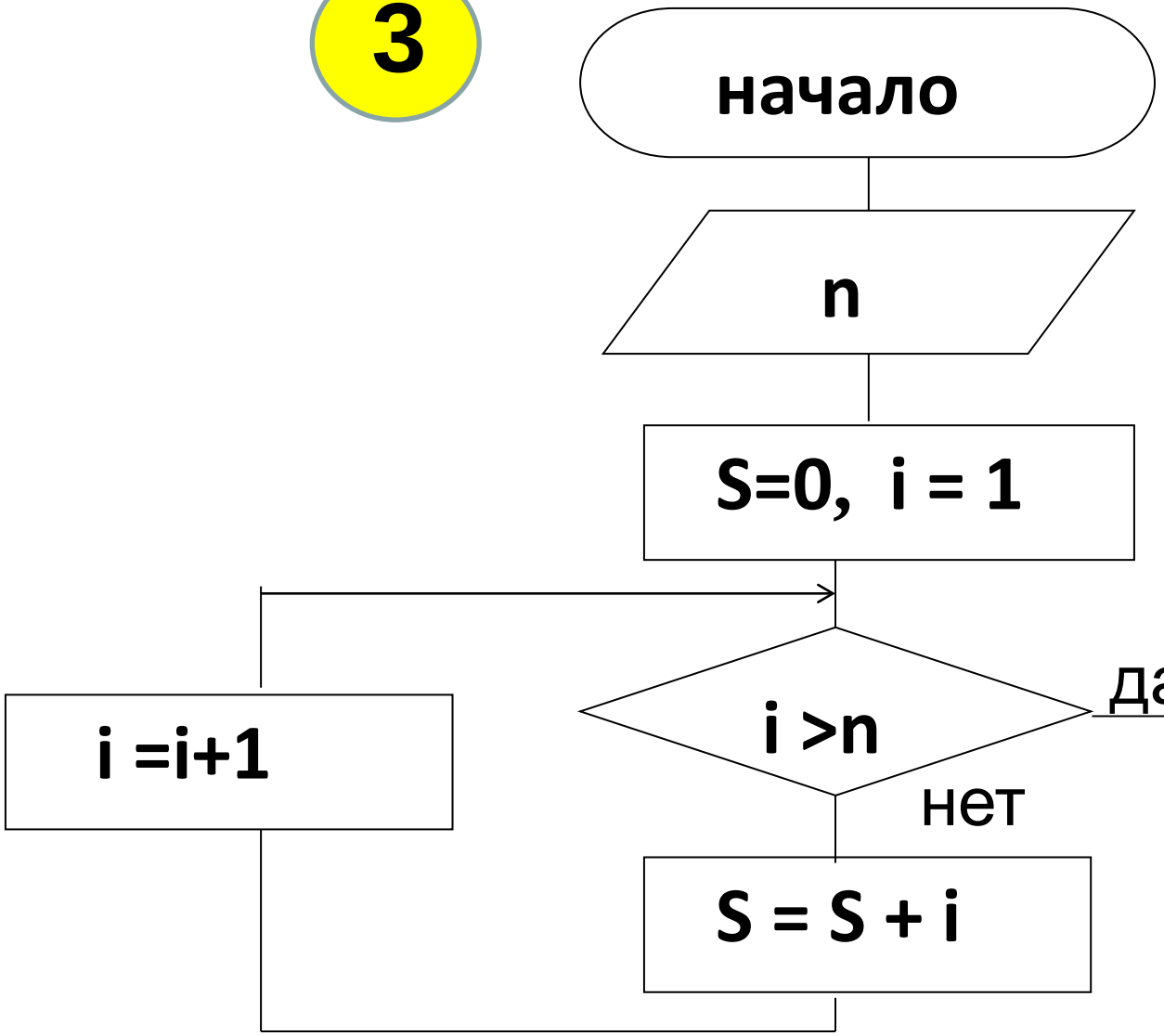
Ссылка на текущую и следующую страницы




1
начало
A, B
C = A
A = B
B = C
A, B
останов


2
начало
A, B
A>B
да
max=A
нет
max=B
max
останов

3



n=3

S=6

i – параметр цикла

Алгоритм «Опрос студентов» (не для написания кода ☺):

Парадигмы программирования

(совокупность идей и понятий, определяющих стиль написания компьютерных программ)

Императивное программирование (машинные коды, Ассемблеры, Fortran, Java, Python, JavaScript, C, C++) – содержит прямое указание, какие инструкции и в каком порядке нужно выполнить (процедурное, ООП программирование, ….)

Декларативное программирование – программы пишут в виде функций ( ) или некоторых ограничений и правил (Prolog), из которых компьютер генерирует способ получения результата. Обычно отсутствует изменение переменных или оно спрятано.

Преимущества - безопасный и поддерживаемый код, который легко параллелится, компиляторы декларативных языков имеют больше возможностей при оптимизации программ.

Парадигмы программирования

(совокупность идей и понятий, определяющих стиль написания компьютерных программ)

- Процедурное программирование (Basic, **C**, Fortran, Pascal)
- Объектно-ориентированное (Python, **C++**, Java, C#)
- Функциональное программирование (Haskell, Scala, Erlang)
- Метапрограммирование
- Обобщенное программирование
- Логическое программирование (Prolog)

➢ Процедурное программирование

Программа – набор подпрограмм (процедур и функций), каждая из которых выполняет определенный блок кода с нужными входящими данными.

Использует следующие принципы:

- локальные переменные;
- глобальные переменные;
- передача параметров (по значению, по адресу – указателю и ссылке);
- модульность.

Функции могут изменять внешние переменные, что может приводить к ошибкам.

Процедурное программирование хорошо подходит для легких программ без сложной структуры.

➢ Процедурное программирование

Pascal

```
Program Prim1;
function Sum (a, b : integer ): integer;
  begin
     Sum := a + b;
  end;

function Subtraction (a,b : integer ): integer;
  begin
     Subtraction := a - b;
  end;

begin
     writeln ('sum=', Sum ( 5, 2 ));
     writeln ('sub=', Subtraction ( 5, 2 ))
end.
```

```
#include <stdio.h>
```

```
int Sum (int a, int b)
{
    return a + b;
}
```

C (Си)

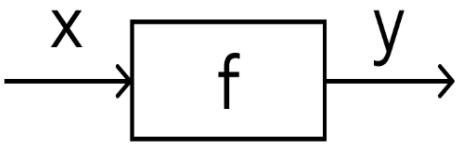
Ритчи Деннис М., Керниган Брайан У.
Язык программирования Си

```
int Sub (int a, int b)
{
    return a - b;
}
```

```
int main()
{
  printf( "sum=%d\nsub=%d", Sum(5,2), Sub(5,2));
    return 0;
}
```

Понятие функции (знать!)

Функция – это подпрограмма (некоторый код), которая вызывается к исполнению по имени, получает входные данные для работы через параметры (аргументы функции), обрабатывают их по некоторому скрытому от нас алгоритму и возвращает результат.



Функция может:

- получать данные из вызывающей программы и сюда же возвращать результат;
- получать данные с клавиатуры или из файла посредством ввода данных в переменные программы, выводить данные на экран или в файл.

Есть функции, встроенные в ЯП или содержащиеся в подключаемых к программе библиотеках, а есть пользовательские функции, которые пишет программист.

Объявление функции: **тип имя_функции (типы аргументов);**

```
int sum(int, int);
```

Определение функции:

тип имя_функции (описание аргументов)
{
тело функции – исполняемые инструкции (операторы)
}

```
int sum(int a, int b)
{
    return a + b;
}
```

Вызов функции: **имя_функции (аргументы);**

```
int s = sum(5, 10);
```

Понятие функции (знать!)

- Все переменные, объявленные в заголовке и в теле функции – локальны по отношению к функции, существуют только во время работы функции (под них выделяется память в момент вызова функции и только на время ее работы).
- Все переменные объявленные вне функции – глобальны по отношению к функции. Не следует внутри функции использовать глобальные переменные!
- Если функция объявлена с типом `void`, у нее нет значения, и ее работа заканчивается оператором `return;`
- Иначе функция имеет значение, которое возвращает в программу в точку вызова функции

```
return выражение;
```

Тип выражения должен совпадать с типом функции.

Примеры объявления функций. Какие алгоритмы они реализуют?

```
int max1(int, int);
int max2(int, int, int);
float max3(float, float);
float max4(float [100]);
float Sort(float [100]);
bool Find(float [100], float);
            //поиск заданного значения в
            массиве вещественных чисел
            размерностью 100 элементов
```

Понятие функции (знать!)

Детерминированные функции – если функции всегда производит один и тот же результат при одинаковых входных данных. Пример – вычисление синуса.

Недетерминированные функции – ее результат зависит не только от входных данных (значений аргументов). Пример – функция-генератор случайных чисел.

Функции с побочными эффектами – они могут менять что-то за рамками функции – значения глобальных переменных. Это «опасные» функции. Иногда они полезны, но общее принятое положение – минимизация побочных эффектов.

Вопрос: Детерминированные функции могут давать побочные эффекты?

Ответ: Могут!

Пример детерминированной функции sum

```c
#include <stdio.h>

int sum(int, int);

int main()
{
    int s = sum(5, 10);
    printf("sum = %d", s);
    return 0;
}

int sum(int a, int b)
{
    return a + b;
}
```

```
sum = 15
```

Пример побочного эффекта функции sum

```c
#include <stdio.h>

int count = 0; //глобальная переменная
int sum(int, int);

int main()
{
    int s = sum(5, 10);
    printf("sum = %d,\ncount = %d", s, count);
    return 0;
}

int sum(int a, int b)
{   count++;
    return a + b;
}
```

использование глобальной переменной – в принципе так делать нельзя

```
sum = 15,
count = 1
```

➢ Объектно-ориентированное программирование

ООП – программа рассматривается как набор объектов, взаимодействующих друг с другом. У объектов есть свойства (параметры) и поведение (методы класса).

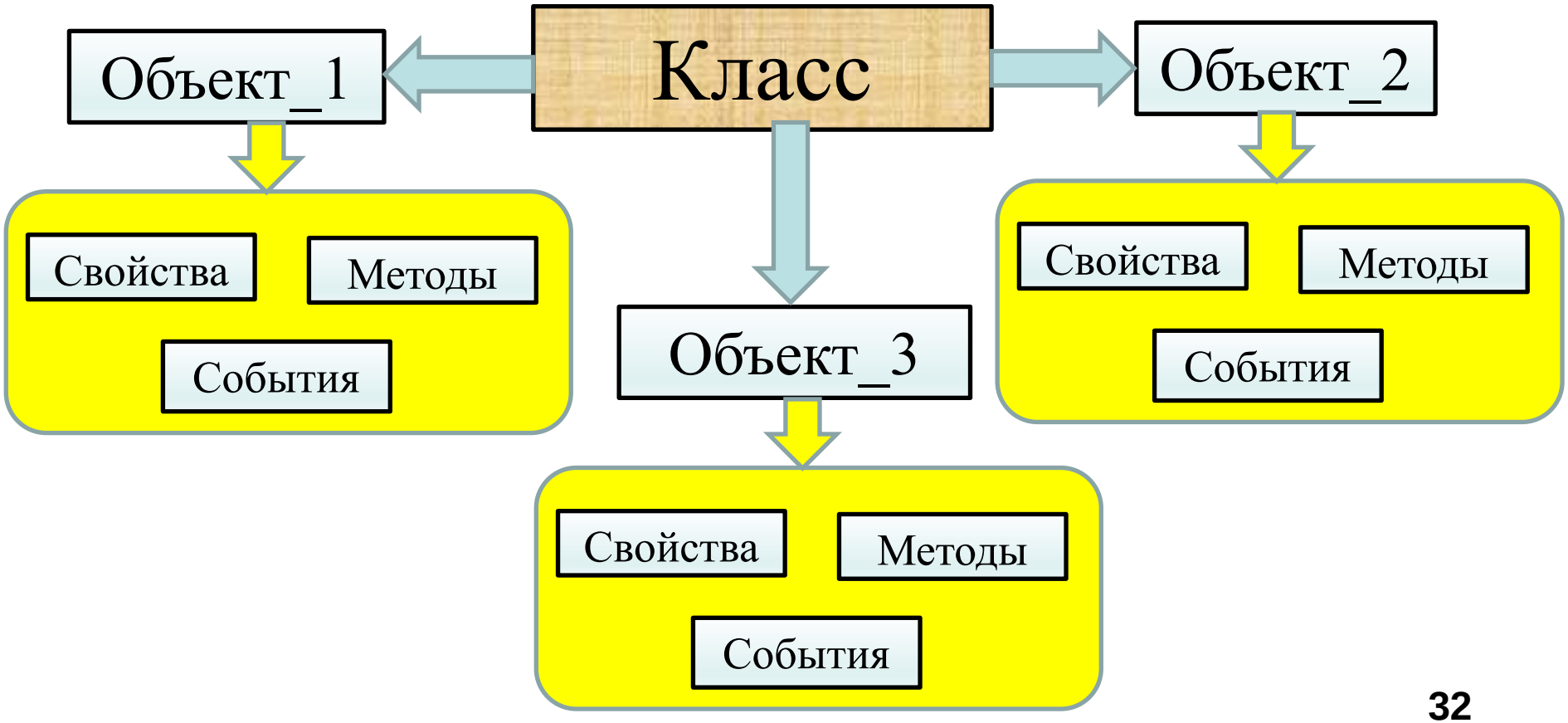
Класс – это шаблон для объекта, описывающий его свойства. Объекты – это экземпляры некоторого класса. Объекты взаимодействуют друг с другом по некоторым прописанным алгоритмам. В программе может быть код, не связанный с функционированием объектов. ООП – это императивное программирование.

К ООП можно приступать, освоив процедурный подход и изучив теорию ООП, порог входа в которую высокий.

➢ Объектно-ориентированное программирование

Принципы ООП

Абстракция	Инкапсуляция
Наследование	Полиморфизм



➢ Объектно-ориентированное программирование

C++

```
#include <iostream>
using namespace std;
class Arithmetic_Operations {
    int x, y;  //свойства, по умолчанию private
public:        //методы, доступны из любого места программы
    void get_xy()
    {   x = 5;  y = 2;  }

    int sum_xy ()
    {   return x + y;  }

    int sub_xy ()
    {   return x - y;  }
} s; // объявлен объект s класса Arithmetic_Operations

int main()
{   s.get_xy(); //обращаемся к объекту класса - вызываем метод
    cout << "sum=" << s.sum_xy() << endl;
    cout << "subtraction=" << s.sub_xy();
}
```

Что такое пространство имен:

C++

Пространство имен (namespace)
имеет имя Audience_4201

```
namespace Audience_4201 {
.   .    .
}
```

Вопрос:

Этому пространству имен что может принадлежать?

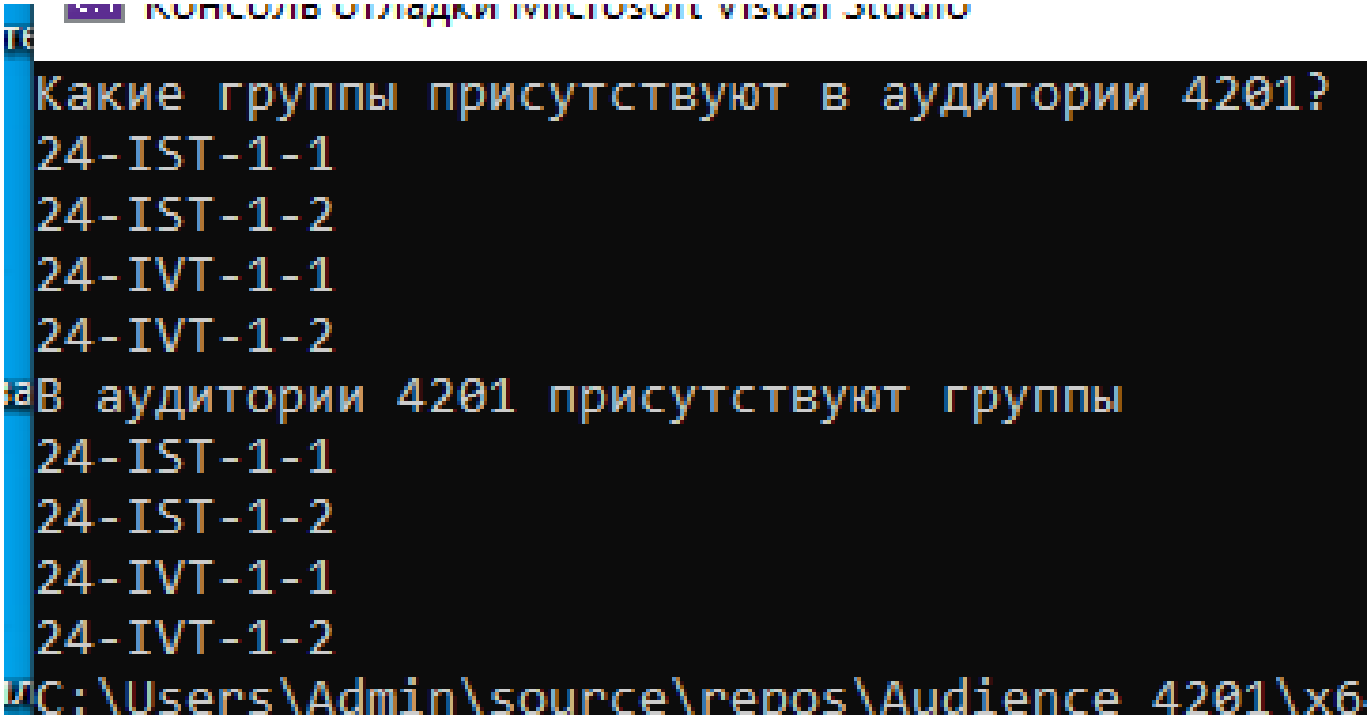
```
namespace Audience_4201 {
    char name_group[4][12];
    char group_list[20][50];
}
```

C++

С функциями из Си

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <locale.h>
namespace Audience_4201 {
    char name_group[4][12];
    char group_list[20][50];
}


int main()
{
    setlocale(LC_ALL, "Russian");
    printf("Какие группы присутствуют в аудитории 4201?\n");
    scanf("%s", Audience_4201::name_group[1]);
    scanf("%s", Audience_4201::name_group[2]);
    scanf("%s", Audience_4201::name_group[3]);
    scanf("%s", Audience_4201::name_group[4]);
    printf("В аудитории 4201 присутствуют группы\n");
    printf("%s\n%s\n%s\n%s", Audience_4201::name_group[1],
        Audience_4201::name_group[2],
   Audience_4201::name_group[3], Audience_4201::name_group[4]);
}
```


Какие группы присутствуют в аудитории 4201?
24-IST-1-1
24-IST-1-2
24-IVT-1-1
24-IVT-1-2
В аудитории 4201 присутствуют группы
24-IST-1-1
24-IST-1-2
24-IVT-1-1
24-IVT-1-2
C:\Users\Admin\source\repos\Audience 4201\x6

C++

С функциями из Си

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <locale.h>
namespace Audience_4201 {
    char name_group[4][12];
    char group_list[20][50];
}

using namespace Audience_4201;
int main()
{
    setlocale(LC_ALL, "Russian");
    printf("Какие группы присутствуют в аудитории 4201?\n");
    scanf("%s", name_group[1]);
    scanf("%s", name_group[2]);
    scanf("%s", name_group[3]);
    scanf("%s", name_group[4]);
    printf("В аудитории 4201 присутствуют группы\n");
    printf("%s\n%s\n%s\n%s", name_group[1], name_group[2],
   name_group[3], name_group[4]);
}
```

C++

```
#define _CRT_SECURE_NO_WARNINGS
#include <iostream>
#include <locale.h>
namespace Audience_4201 {
    char name_group[4][12];
    char group_list[20][50];
}
using namespace std;
using namespace Audience_4201;

int main()
{
   setlocale(LC_ALL, "Russian");
   cout << "Какие группы присутствуют в аудитории 4201? " << endl;
   cin >> name_group[1] >> name_group[2] >> name_group[3] >>
          name_group[4];
   cout << "В аудитории 4201 присутствуют группы " << endl <<
       name_group[1] << endl << name_group[2]<< endl <<
       name_group[3] << endl << name_group[4];
}
```

➢ Функциональное программирование

В функциональных языках (LISP, ML, Miranda, Haskell) вычисления реализуются в виде функций в их математическом понимании. Разработка программ заключается в создании из простых функций более сложных. При этом результат функции зависит только от исходных данных, не дает побочных эффектов, не влияет на внешние переменные.

Используется для создания высоконагруженных систем с высоким уровнем параллельных вычислений

➢ Метапрограммирование

Построение кода с динамическим изменением его поведения или структуры в зависимости от данных, действий пользователя или взаимодействия с другими системами (C, C++, Python, Java, JS, Ruby и т.д.).

Это разработка программ, которые в результате своей работы порождают другие программы.

Простейшие компилируемые техники – шаблоны (template), макросы и параметрический полиморфизм, оптимизирующая предкомпиляция (препроцессорная обработка), генерация исходного кода.

Обобщенное программирования (шаблоны)

```
template <typename T>
    T max(T first, T second) {
        if (second > first) {
            return second;
        }
        return first;
    }
}
```

Си

C++

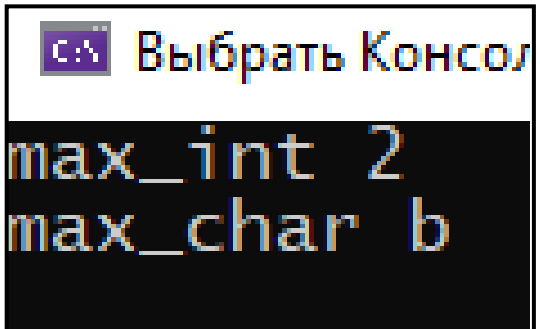
```
printf("max_int %d\n", max(1,2));
printf("max_char %c", max('a','b'));
```

```
cout << "max_int “ << max(1, 2) << endl;
cout << "max_char "<< max('a','b’) << endl;
```

В результате будут сгенерированы 2 функции для параметров 1, 2 и ‘a’, ‘b’

```
int max(int first, int second) {
        if (second > first) {
            return second;
        }
        return first;
    }
}
```

```
char max(char first, char second)
{
        if (second > first) {
            return second;
        }
        return first;
    }
}
```



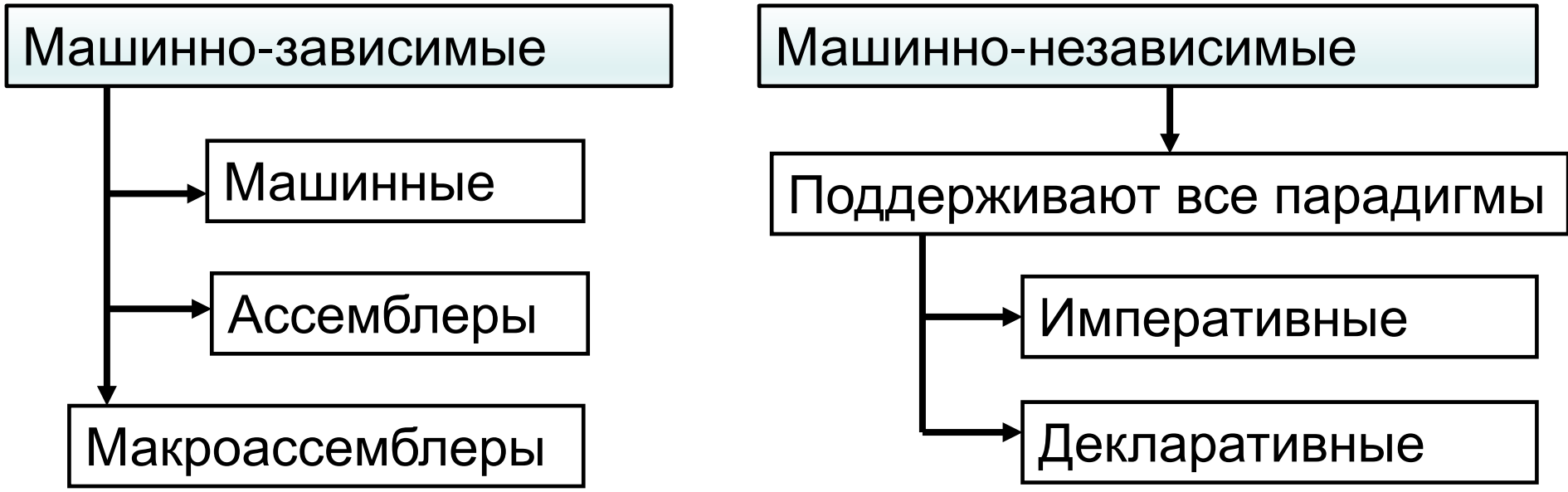
➢ Логическое программирование (Prolog)

Использует декларативное программирование и принцип устойчивости к изменениям: описывается задача и её свойства, а не алгоритм решения. Задача представляется в виде:

Проблема = Модель + Поиск
Вариация задачи = Базовая Модель + Доп. условие

Такой подход используется в базах данных. SQL — это декларативный язык запросов, а поиском ответа на этот запрос занимается СУБД. Для эффективной работы СУБД разработаны эффективные алгоритмы, данные хранятся в оптимизированном виде, используются индексы, методы оптимизации запросов и т.д.
Области применения: автоматический перевод, обработка текстов, экспертные системы, САПР, Data-mining системы, автоматическое управление, СУБД, символьные вычисления.

Классификация языков программирования



Императивные (Java, Python, JavaScript, C, C++, и т.д.) – дают указания, что должен сделать компьютер, и в каком порядке должны выполняться инструкции. Основные элементы – переменные и операторы присваивания, поддерживаются циклы, рекурсии.

Декларативные – программы пишутся в виде некоторых ограничений и правил, операторы – объявления или высказывания в символьной форме (Пролог).

➢ Процедурное программирование

Инструкции выполняются сверху вниз, последовательно, код может содержать подпрограммы

Основное внимание уделяется АЛГОРИТМАМ

Программа – набор подпрограмм,
в которых **скрыты алгоритмы**,
а **обмен данными** осуществляется **через параметры**

Разработаны встроенные в компилятор библиотеки

➢ Объектно-ориентированное программирование

Язык приспосабливают для решения задач

Создают **формы данных**, соответствующих специфике задачи

Основной акцент делается на ДАННЫЕ

Программа – набор **взаимодействующих друг с другом объектов**, при этом у каждого объекта в системе есть свойства и поведение.
Высокая производительность, написание больших проектов

Разработано большое количество библиотек классов

➢ Обобщенное программирование

Направлено на решение задач общего характера

Создание **кода программы, независимого от типа данных**

Основной акцент делается на **АЛГОРИТМЫ**

Разрабатывается функция (шаблон), например, сортировки, для неопределенного типа данных, которая впоследствии применяется для разных реальных типов.

Построение кода, который можно использовать многократно с объектами разных типов Поддерживается строго типизированными языками. (C++, Python, Java, Object Pascal, D, Eiffil, платформы .NET и т.д.).

Августовский рейтинг 2021, 2022 языков программирования TIOBE

Aug 2022	Aug 2021	Change	Programming Language	Ratings	Change
1	2	↑	Python	15.42%	+3.56%
2	1	↓	C	14.59%	+2.03%
3	3		Java	12.40%	+1.96%
4	4		C++	10.17%	+2.81%
5	5		C#	5.59%	+0.45%
6	6		Visual Basic	4.99%	+0.33%
7	7		JavaScript	2.33%	-0.61%
8	9	↑	Assembly language	2.17%	+0.14%
9	10	↑	SQL	1.70%	+0.23%
10	8	↓	PHP	1.39%	-0.80%
11	16	⇈	Swift	1.27%	+0.30%
12	12		Classic Visual Basic	1.27%	+0.04%
13	22	⇈	Delphi/Object Pascal	1.22%	+0.60%

Августовский рейтинг 2022, 2023 языков программирования TIOBE

May 2023	May 2022	Change	Programming Language	Ratings	Change
1	1		Python	13.45%	+0.71%
2	2		C	13.35%	+1.76%
3	3		Java	12.22%	+1.22%
4	4		C++	11.96%	+3.13%
5	5		C#	7.43%	+1.04%
6	6		Visual Basic	3.84%	-2.02%
7	7		JavaScript	2.44%	+0.32%
8	10	^	PHP	1.59%	+0.07%
9	9		SQL	1.48%	-0.39%
10	8	v	Assembly language	1.20%	-0.72%

Августовский рейтинг 2024, 2025 языков программирования TIOBE

Aug 2025	Aug 2024	Change	Programming Language	Ratings	Change
1	1		Python	26.14%	+8.10%
2	2		C++	9.18%	-0.86%
3	3		C	9.03%	-0.15%
4	4		Java	8.59%	-0.58%
5	5		C#	5.52%	-0.87%
6	6		JavaScript	3.15%	-0.76%
7	8	↑	Visual Basic	2.33%	+0.15%
8	9	↑	Go	2.11%	+0.08%
9	25	⇈	Perl	2.08%	+1.17%
10	12	↑	Delphi/Object Pascal	1.82%	+0.19%
11	10	↓	Fortran	1.75%	-0.03%
12	7	⇊	SQL	1.72%	-0.49%

Выводы:

- Нет смысла учить только синтаксис языка C/C++. Учить нужно программирование)))
- Если понять основы программирования – легко понять отличия в языках и выучить синтаксис другого языка.
- Программист может заниматься совершенно разным – писать вычислительные алгоритмы, писать компиляторы языков, быть «бэкенд» или «фронтенд» разработчиком, программировать «железо», заниматься моделированием процессов, писать БД, заниматься аналитикой, в том числе видео аналитикой, разрабатывать сайты ….

- **Язык C/C++ - сложный язык, но и один из самых актуальных и востребованных**

На 1 курсе студент задает вопросы:

Как начать учиться IT- специальности?
Какие книги читать?
У кого-то информатики в школе практически не было!

В современных книгах можно увидеть следующее:

«Настоящая книга не является вводным курсом в программирование; она предполагает определенное знакомство с основными понятиями программирования такими как переменные, операторы присваивания, циклы, функции»

или:

«предполагается рабочее владение основными элементами программирования; здесь не объясняется, что такое ЭВМ или компилятор, не поясняется смысл выражений типа N=N+1, а также терминология, например: Символические константы, и т.д.

Как повысить эффективность учёбы

1. Завести **словарь терминов**, на лекциях, дома заполнять его (например, термины разместить по алфавиту).
2. Записывать важные моменты на лекциях и практиках, т.к. в презентациях – фрагменты кодов, отмечены основные моменты, подробные объяснения идут устно (например, записывайте: слайд № – и комментарии, пояснения).
3. Учим английский, не переходим на кириллицу в кодах
4. Выполняем задания дома на **компьютере, это ваш репетитор!**
5. Не забываем:

– читаем сообщения об ошибках 

– о предупреждениях 

– можно кликнуть на код ошибки и зайти в Microsoft Документацию,

– если в коде подчеркивание красной волнистой чертой – ошибка!

зеленой волнистой чертой – предупреждение!

– пользуемся всплывающими подсказками

– гуглим, спрашиваем, в т.ч. у одногруппников – помогите !!!

Интегрированная среда разработки IDE (Integrated Development Environment)

Visual Studio – полнофункциональная среда разработки от компании Microsoft, легко войти в процесс разработки (бесплатно распространяется Visual Studio Community, с достаточным набором возможностей).

Code::Blocks поддерживает C, C++ и Fortran, являясь для них одной из лучших IDE. Для C и C++ - хорошая настраиваемость и гибкость, доступность множества плагинов. Функционал этой среды разработки можно расширять, в том числе с помощью пользовательских плагинов.

Dev C++ – популярная IDE для разработки на C/C++ на начальном уровне.

QT Creator – кроссплатформенная IDE, удобная и быстрая интегрированная среда разработки C++, предлагающая интересные возможности для разработчика.

Eclipse – свободная IDE, поддерживающая язык Си стандарта C99. Имеет модульную архитектуру, что даёт возможность подключения поддержки разных языков программирования и дополнительных возможностей.

Про Visual Studio C/C++

- Стандартный компилятор IDE Visual Studio не поддерживает в полной мере новые стандарты языка Си (C99, C11). Но почти все, что мы будем изучать, относится к той части стандартов, которая не меняется уже очень давно и поддерживается всеми компиляторами и IDE.

- В дисциплине «Цифровая обработка сигналов» и «Программирование микроконтроллеров» вы будете заниматься низкоуровневым программированием, например, контроллеров STM32, изучать архитектуру Risc-V и работать в IDE, поддерживающей стандарты Си, например Eclipse (*например, при работе с «железом» важно знать, как хранятся в регистрах значения и как их оттуда «взять» в нужном формате!!!)*

Вывод – Изучаем работу функций printf() и scanf() ☺

Язык C был создан Дэннисом Ритчи, сотрудником лаборатории Bell Labs (1969 – 1973), как язык системного программирования. Компилятор языка C был написан на самом C.

Вместе с Кеном Томпсоном написали бестселлер «Язык программирования Си».

На C были разработаны ОС UNIX, часть Windows, MacOS.

легкий доступ к аппаратным средствам компьютера

компиляторы, ОС, экранные интерфейсы, консольные утилиты, игровые движки, можно программировать контроллеры, системы реального времени

Сочетает лучшие свойства ассемблеров и языков высокого уровня. Перспективной замены C как системного языка - нет.

C++ был создан Бьёрном Страуструпом, сотрудником компании AT&T в начале 80-х гг.

Символами «++» обозначается операция инкремента (увеличение значения переменной на 1) .

Язык C вошел в C++, программы на C компилируются с помощью компилятора C++ .

совместим со стандартным языком C

добавлены конструкции ООП, основанные на концепции классов из языка Simula, и строгая проверка типов

Управляющие структуры, позволяющие создавать хорошо структурированные программы :

- оператор ветвления `if (condition) operator1 else operator2`
- оператор выбора варианта `switch`
- оператор повторения с предусловием `while, for`
- оператор повторения с постусловием `do { } while`

Стандартные типы данных:

- целый `int`
- символьный `char`
- вещественный `float, double`

Пользовательские типы данных:

Объявления типов	

Массивы	`int x[20];`
	`float temptrature[365];`
	`char subject[30];`

структуры	`struct person`
	`{ string name;`
	`int age};`

Файлы	`FILE* f_Person;`

Списки (очереди, стеки, деревья)	`struct comp {`
	`int Data;`
	`comp *next;`
	`};`

В Си слабый контроль типов, который позволяет создавать высокоэффективные программы

В C/C++ хорошо реализован механизм указателей на переменные и функции.

Указатель – это переменная для хранения машинного адреса некоторой переменной или функции.

тип **имя переменной**

```
int*            p_int;
float*          p_weight;
struct person*  p_struct_person;
```

Поддерживается арифметика указателей, позволяющая осуществлять доступ к адресам памяти, как и в языке Ассемблер.

В C/C++ например, нет:

- встроенных систем ввода/вывода
- встроенных средств для работы с символьными строками
- встроенных средств динамического распределения памяти и т.д.

Есть библиотеки, входящие в системное окружение языка, подключаемые с помощью директивы

```
#include <stdio.h>
```

Некоторые отличия

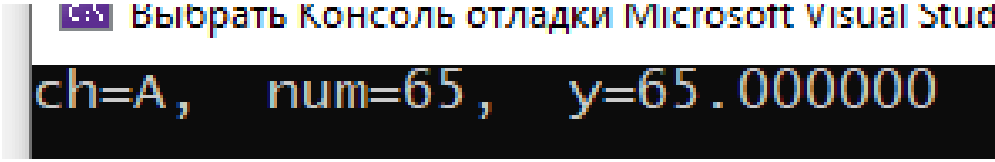
C	C++
Код читается проще	Код выглядит сложнее
Проще портировать код на другие платформы	Из-за особенностей языка код портировать сложнее
Есть структуры и переменные типа «структура»	Есть классы и объекты
Наследование — это структура внутри структуры	Классическое наследование в стиле ООП
void — обязательное слово	Использовать слово void не обязательно
Можно объявлять глобальную переменную несколько раз	Глобальная переменная объявляется только один раз
Исключения нужно конструировать самому	Есть обработка исключений
Нет перегрузки операторов	Операторы можно перегружать
Почти всё стандартно и предсказуемо – *при хорошем знании языка!*	Легко написать код, в котором непонятно, где ошибка

C – позволяет писать низкоуровневые программы на языке высокого уровня, это процедурное программирование.

- Низкий уровень – позволяет работать с аппаратными типами данных, поддерживаемыми микропроцессором (1, 2, 4, 8 байт), реализованы логические операции над битами, сдвиги, работа с адресами и регистрами.
- Высокий уровень – структурные типы данных, операторы структурного программирования, указатели, функции, большой набор операций (16 приоритетов).

Это слабо типизированный язык

```
char ch;
int num;
float y;
y = num = ch = 'A';
printf("\nch=%c, num=%d, y=%f", ch, num, y);
```



Будет работать? `ch = num = y = 'A';`

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;
3   (x = 7) + (c = 10 + x);
4   ++(n = (k = 5) % 2);
5   if (-1 > (unsigned int) 1)  *p++;
```

BCE