# Парадигмы программирования

# Парадигма программирования

- это совокупность идей и понятий, определяющих стиль написания компьютерных программ (подход к программированию)
- не определяется однозначно языком программирования
- многие языки программирования допускают использование различных парадигм

# Парадигма предполагает использование определенных сущностей

- состояние данных и команды, изменяющие их (императивное программирование)
- математические функции без состояния, предоставляющие свой результат (функциональное программирование)
- объекты и взаимодействие между ними (объектно-ориентированное программирование)
- алгоритмы и контейнеры, оперирующие типами данных, переданными как параметр (обобщенное программирование)

# Императивная парадигма

- Основана на модели машины Тьюринга
- Пошаговое описание изменения состояния программы (данных)

# Императивное программирование

- Основные понятия:
  - Инструкция
  - Состояние
- Порожденные понятия:
  - Присваивание
  - Переход
  - Память
  - Указатель

  Вычисления описываются в виде инструкций, изменяющих состояние программы.

# Низкоуровневые языки

- Языки ассемблера

состоянием могут быть память, регистры и флаги,

а инструкциями— те команды, которые поддерживает

целевой процессор.

# Более высокоуровневые

- C

состояние— это только память

инструкции – могут быть сложнее и вызывать выделение и освобождение памяти в процессе своей работы

# Совсем высокоуровневые

- Python (если на нем программировать императивно)

состояние ограничивается лишь переменными,

команды могут представлять собой комплексные операции, которые на ассемблере занимали бы сотни строк

# Методы императивного программирования

- Структурное (Э. Дейкстра, Н.Вирт)

- Основные понятия:
  - Блок
  - Цикл
  - Ветвление

# Методы императивного программирования

- Процедурное

Основные понятия:

— Процедура

Порожденные понятия:

— Вызов
— Аргументы
— Возврат
— Рекурсия
— Перегрузка

# Методы императивного программирования

- Модульное программирование

Основные понятия:

— Модуль

— Импортирование

Порожденные понятия:

— Пакет

— Инкапсуляция

Впоследствии появились:

пакеты (тоже модульное программирование)

классы (это уже ООП),

шаблоны (обобщенное программирование)

# Функциональное программирование

- Основа – математическая модель «$\lambda$-исчисление» Чертча
- программы представляют из себя вычисление функций именно в их математическом понятии
- в идеале нет глобального состояния программы и от него ничего не должно зависеть

# Функциональное программирование

- Функции не должны иметь побочных эффектов (pure)
- данные стараются делать не изменяемыми (immutable)

- наиболее яркий представитель- Haskell

# Логическое программирование

- Основа – математическая модель логики предикатов

- задаётся набор фактов
- описываются правила вывода
- формулируется проблема

на основе этой информации компьютер выдаёт ответ

- Яркий представитель- язык Prolog

# Обобщенное программирование

- рассматривается как методология программирования, основанная на определении структур данных и алгоритмов через использование абстрактных описаний требований

# Обобщенное программирование

- заключается в таком описании данных и алгоритмов, которое можно применять к различным типам данных, не меняя само это описание
- в 1970-х годах в языках Клу и Ада
- C++, Java, языки для платформы .NET и др.

# Обобщённое программирование в C++

- основывается на понятии «шаблон»
- широко применяется в стандартной библиотеке шаблонов C++ (STL), а также в сторонних библиотеках (boost)

# Пример

```
<typename T>
 T max(T x, T y) {
    if (x < y)      return y;
       else         return x;
 }
  ...
  int a = max(10,15);
  ...
  double f = max(123.11, 123.12);
  ...
```

# Метапрограммирование

вид программирования, связанный с созданием

программ, которые порождают другие программы как результат своей работы (в частности, на стадии компиляции их исходного кода),

либо программ, которые меняют себя во время выполнения (самомодифицирующийся код)

# Самомодифицирующийся код

- Возможность изменять или дополнять себя во время выполнения превращает программу в виртуальную машину

# МЕТАПРОГРАММИРОВАНИЕ С ШАБЛОНАМИ

# Вычисление факториала на стадии компиляции

```
template <int n>
struct Factorial {
   enum { val = Factorial<n-1> :: val * n };
};
template <>
struct Factorial <0> {
   enum { val = 1 };
};
```

# Вычисление факториала на стадии компиляции

```
#include <iostream>
using namespace std;
int main() {
 cout<< Factorial <12> :: val<<endl;          //479001600
 double nums [Factorial <5>::val];
  assert(sizeof nums == sizeof (double)*120);
  return 0;
}
```

- Обладает полнотой по Тьюрингу, т.к. поддерживает:
  - выбор
  - циклы (посредством рекурсии)

Теоретически можно выполнять любые вычисления