

# Технологии программирования

**Разработал:**  
учитель информатики  
первой категории  
МБОУ МО г. Нягань «Гимназия»  
Юдина И.И.

# Технологии программирования

совокупность методов и средств,  
используемых в процессе разработки  
программного обеспечения.

# Первый этап

## «Стихийное программирование»

(от момента появления первых вычислительных машин до середины 60 годов XX в.)

Практически отсутствовали сформулированные технологии, и программирование фактически было искусством.

Первые программы имели простейшую структуру.



Они состояли из собственной программы на машинном языке и обрабатываемых ею данных.

# Второй этап

## Структурный подход к программированию

(60-70 годы XX в.)

### В основу положены следующие положения:

- программы должны состояться мелкими шагами; размер шага определяется количеством решений, применяемых программистом на этом шаге;
- сложная задача должна разбиваться на достаточные простые, легко воспринимаемые части, каждая из которых имеет только один вход и один выход;
- логика программы должна опираться на минимальное число достаточно простых базовых структур.

# Третий этап

## Объектно-ориентированное программирование

(с середины 80 до конца 90 годов XX в.)

Определяется как технология создания сложного программного обеспечения, основанная на представлении программы в виде совокупности **объектов**, каждый из которых является экземпляром определенного типа (класса), а **классы** образуют иерархию с наследованием свойств.

# Четвертый этап

## Компонентный подход и CASE-технологии

(с середины 90 годов XX в. до нашего времени)

Особенностью этого этапа является создание и внедрение автоматизированных технологий разработки и сопровождения программного обеспечения, которые были названы **CASE-технологиями** (Computer-Aided Software/System Engineering – разработка программного обеспечения/программных систем с использованием компьютерной поддержки).

Существуют CASE- технологии, поддерживающие как структурный, так и объектный (в том числе и компонентный) подходы к программированию.



# Оптимизация программ

## Оптимизация арифметических выражений

- 1) Некоторые, медленно выполняемые операции, легко заменить на более быстрые.

Сложение выполняется быстрее, чем умножение, поэтому умножение на **небольшое целое** число следует заменить сложением. Например,

$$3 * I = I + I + I$$

Если же в выражении ни все числа являются целыми, то при замене может быть утеряна точность.

Ошибка округления действительных чисел имеет тенденцию накапливаться, а не уменьшаться, так если  $R$  – действительное число, а  $I$  – целое, то запись  $I * R$  будет правильной, чем  $R + R + R + \dots$

$$\underbrace{\hspace{2cm}}_{I \text{ раз}}$$

# Оптимизация программ

## Оптимизация арифметических выражений

- 2) **Преобразование уравнений может привести к исключению операций.**

Например, выражение  $X=2*Y+(1+A)/P+2*T$  можно заменить на  $X=2*(Y+T)+(1+A)/P$ , что позволяет исключить одну операцию умножения.

- 3) **Поскольку деление является более медленной операцией всюду, где возможно, его следует заменять умножением.**

Умножение выполняется в 2 раза быстрее деления.

- 4) **Функция извлечения квадратного корня реализуется обычно гораздо быстрее и точность при этом выше, чем при операции возведении в степени.**

Медленный способ  $A ** 0.5$ , быстрый способ  $\sqrt{A}$



# Оптимизация программ

## Оптимизация арифметических выражений

- 5) Умножение выполняется значительно быстрее, чем возведение в степень, поэтому если показатель степени **небольшое целое** число, то операцию возведения в степень следует заменить несколькими операциями умножения.

Например,

