Лекция 1

ОСНОВЫ ТЕХНОЛОГИИ ПРОГРАММИРОВАНИЯ

Лекция 1. Основы технологии программирования

План

1. История создания вычислительной техники
2. Понятие технологии программирования
3. Основные современные парадигмы и технологии программирования
4. Литература

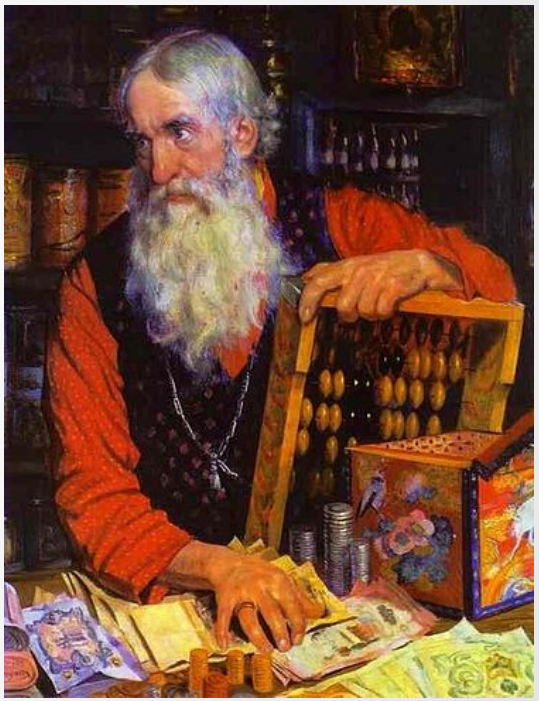
1. История создания вычислительной техники

Абак (счетная доска)

Появились предположительно в 3 тыс. до н.э. в Древнем Вавилоне

Счеты

Появились как развитие китайского счетного приспособления **суаньпань** приблизительно в XIV веке и активно используются до сих пор!!!



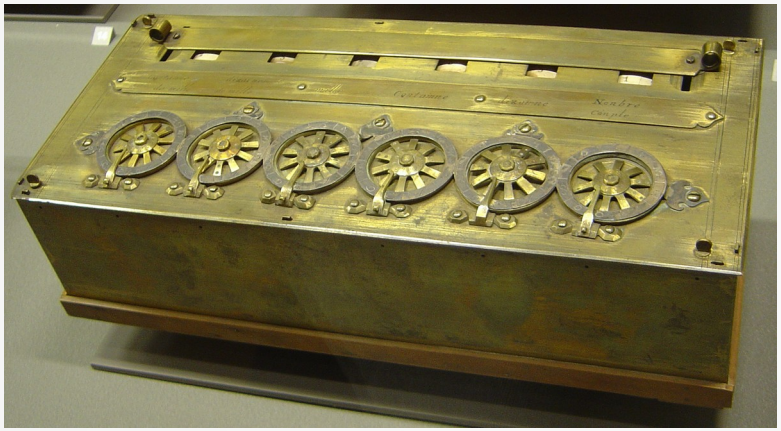
Купец
Б. М. Кустодиев, 1918 год

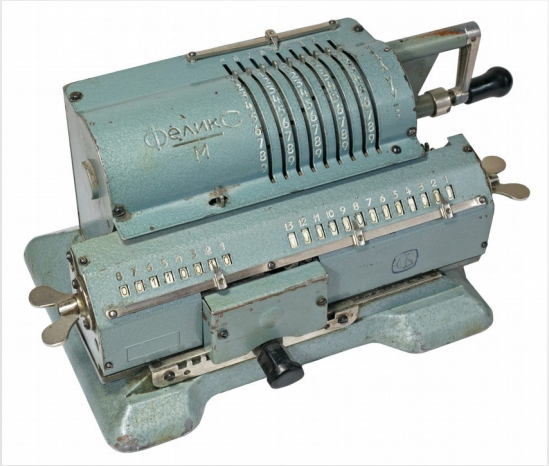
1. История создания вычислительной техники

Блез Паскаль
(1623-1662)

Паскалина (суммирующая машина Паскаля)

Создана французским ученым Блезом Паскалем в 1642 году

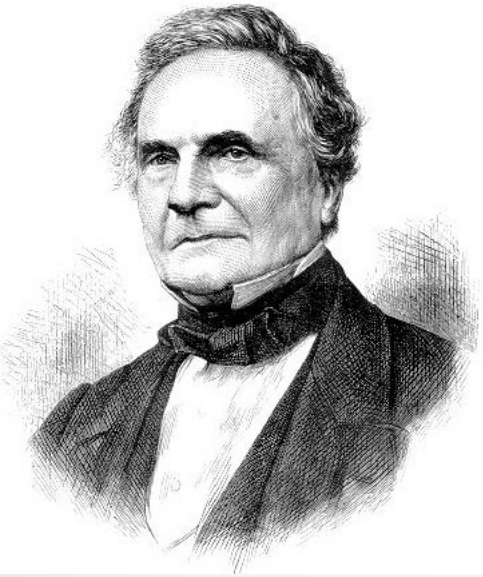




Арифмометр

Портативная механическая вычислительная машина, предназначенная для точного умножения и деления. Выпускалась в СССР до 1978 года.

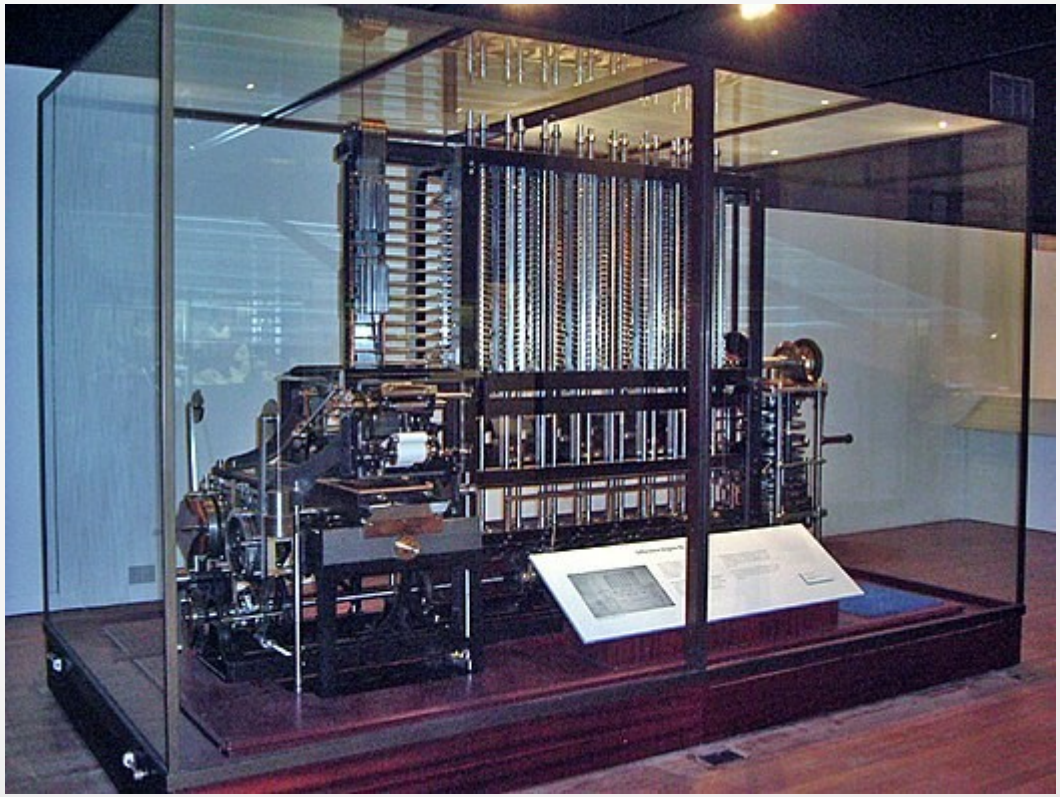
1. История создания вычислительной техники



Чарльз Бэббидж
(1791-1871)

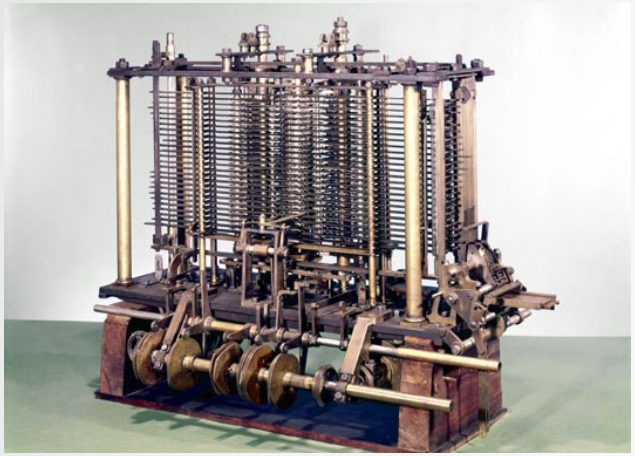
Разностная машина Бэббиджа

Автоматическое вычисление вычислять значения многочленов до шестой степени с точностью до 18-го знака. Предложена Ч. Бэббиджем в 1822 году



Современная реконструкция разностной машины Бэббиджа

1. История создания вычислительной техники



Аналитическая машина Бэббиджа

Прообраз современного компьютера, состоящий их арифметического устройства («мельницы»), памяти («склада») и устройства ввода-вывода, реализованного с помощью перфокарт, которые могли быть использованы как для ввода данных в машину, так и для сохранения результатов вычислений, если памяти было недостаточно.

Ткацкий станок с картами Жаккара

Ада Лавлейс (1815-1852)

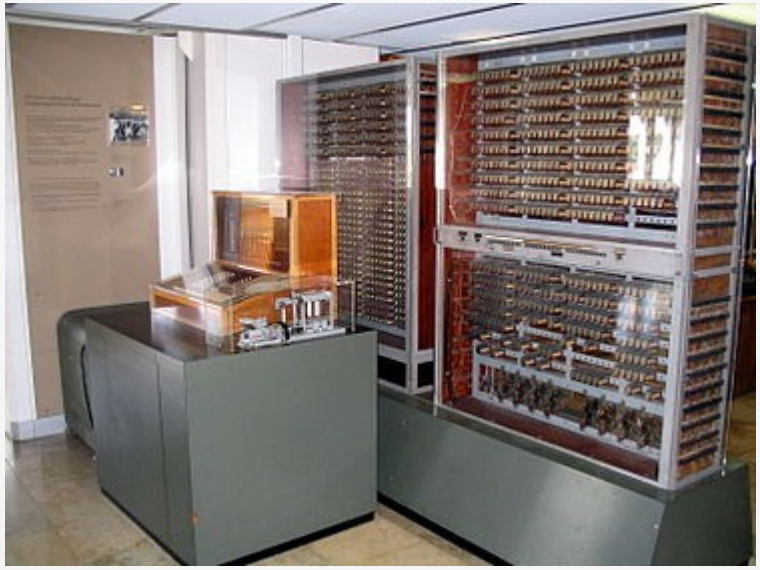
Ада Лавлейс считается первым программистом, так как она впервые описала алгоритм вычисления чисел Бернулли на аналитической машине

1. История создания вычислительной техники

Конрад Цузе
(1910-1995)

Z3

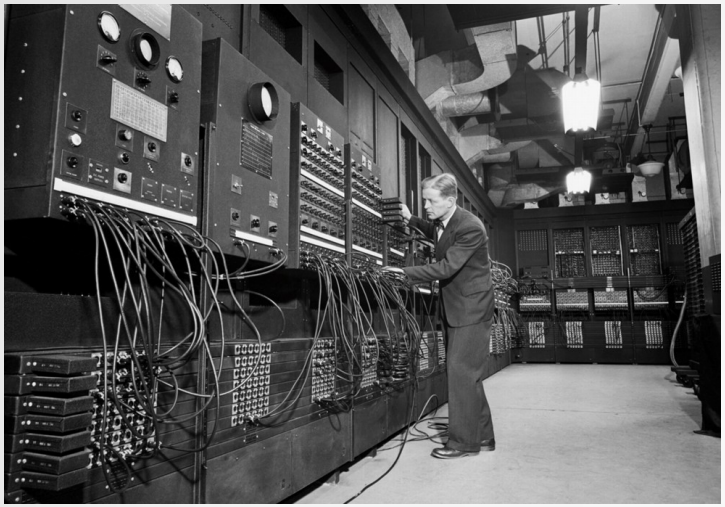
Первая работоспособная полнофункциональная программно управляемая и свободно программируемая вычислительная машина (фактически – первый электро-механический компьютер). Построена в 1941 году в Германии).



Тактовая частота: 5,3 Гц.

К. Цузе разработал первый в мире высокоуровневый язык программирования, названный им **Планкалкюль** (Plankalkül – исчисление планов).

1. История создания вычислительной техники



ENIAC (Electronic Numerical Integrator and Computer)

Первый электронный цифровой вычислитель общего назначения, который можно было перепрограммировать для решения широкого спектра задач. Построен в 1945 году в США).

Основные характеристики

- Вес – 30 тонн.
- Вычислительная мощность – 357 операций умножения или 5000 операций сложения в секунду.
- Тактовая частота – 100 кГц (сложение выполнялось за 1 такт, умножение – за 14 тактов).

1. История создания вычислительной техники

LEO

Первый коммерческий компьютер, который использовался для обработки коммерческих данных (использовался для расчета цен на продукты питания). Построен в 1951 году в Великобритании).

Тактовая частота: 500 КГц.

Altair 8800

Первый в мире персональный компьютер (микрокомпьютер). Разработан в США в 1975 году.

Тактовая частота: 2 МГц

2. Понятие технологии программирования

Программа – это данные, предназначенные для управления конкретными компонентами системы обработки информации в целях реализации определенного алгоритма (ГОСТ)

Программное обеспечение (ПО) – это программа или совокупность программ, предназначенных для управления компьютером.

Классификация ПО по сфере применения:

- научное;
- для бытовых устройств;
- для обеспечения работы оборудования;
- для бизнеса;
- для организации работы сети;
- ...

Классификация ПО назначению:

- прикладное;
- системное;
- инструментальное.

Программирование – процесс разработки ПО, сочетающий в себе элементы науки (математики), инженерии и искусства. В узком смысле программирование – кодирование алгоритма с помощью некоторого языка программирования (алгоритмического языка). В широком – длительный и творческий процесс создания программ.

2. Понятие технологии программирования

Технология программирования (ТП) – это последовательность процессов, исполняемых разработчиками ПО (программистами) для создания алгоритма решения задачи, ее кодирования на языке программирования, тестирования и отладки, а также сдачи заказчику и последующего сопровождения.

Иначе говоря, ТП – это совокупность средств и методов разработки ПО и порядок их применения.

Технологическая цепочка программирования содержит следующие основные этапы:
1) постановка задачи;
2) разработка или поиск алгоритма решения задачи;
3) спецификация исходных данных программы;
4) спецификация функций программы;
5) проектирование программы;
6) кодирование на языке программирования;
7) тестирование и отладка программы;
8) опытная эксплуатация.

2. Понятие технологии программирования

Постановка задачи – делится на неформальную и формальную. На этапе неформальной постановки задачи происходит диалог между заказчиком и разработчиком, главным результатом которого является одинаковое понимание задачи и результатов ее решения. При необходимости это фиксируется в виде формального документа (технического задания), где отражены все требования заказчика к будущему ПО.

Разработка или поиск алгоритма решения задачи – этап, во время которого разрабатывается алгоритм решения задачи (если она новая), или выбирается оптимальный существующий.

Спецификация исходных данных программы – этап, на котором форматы используемых данных, способы их хранения и обработки и т. п.

Спецификация функций программы – разработка описания функциональности программы.

Проектирование программы – этап, на котором создается детальное описание программы, ее функционала, структуры и т. п. Как правило, для этого используются специализированные языки формального или графического описания, например UML.

2. Понятие технологии программирования

Кодирование на языке программирования – непосредственно программная реализация алгоритма.

Тестирование и отладка программы – процесс поиска и устранения возможных ошибок в ПО.

Опытная эксплуатация – процесс проверки программы с целью выявления в ней недостатков и ошибок с последующим устранением.

3. Основные современные парадигмы и технологии программирования

Фактически **технология программирования** – это апробированная стратегия создания программ, которая излагается в виде определенных методик с описаниями проектных процедур и операций.

К наиболее распространенным современным технологиям разработки программного обеспечения относятся следующие:

1) **процедурное программирование;**
2) **структурное программирование;**
3) **объектно-ориентированное программирование;**
4) **визуальное программирование.**

3. Основные современные парадигмы и технологии программирования

В каждой из этих технологий используется одна или несколько **парадигм программирования** – совокупностей принципов, методов и понятий, определяющих способ разработки программ.

Парадигмы программирования представляют собой различные подходы к написанию программ (философии). Для использования каждой из них необходимы: свой тип мышления, особая школа обучения, приемы и способы программирования, определяемые используемым языком.

Существуют четыре основные парадигмы, описывающие большинство современных методов программирования:

- **императивная;**
- **аппликативная;**
- **основанная на системе правил;**
- **объектно-ориентированная**.

3. Основные современные парадигмы и технологии программирования

Императивная парадигма – наиболее распространенный и естественный подход, соответствующий архитектуре стандартного компьютера, выполняющего команды последовательно.

Программа на императивном языке состоит из последовательности выполняемых друг за другом операторов:

```
Оператор_1;
Оператор_2;
Оператор_3;
…
```

К императивным языкам относятся, например, Fortran, C, Pascal, Ada и др.

3. Основные современные парадигмы и технологии программирования

Пример программы на императивном языке программирования BASIC

```
10 CLS 'Очистка экрана
20 PRINT "Добро пожаловать!" 'Заголовок в первой строке
30 'Цикл, выводящий линию под заголовком, на всю ширину экрана
40 FOR I=1 TO 80
50 PRINT "=";
60 NEXT I
65 'Ввод символьных данных от пользователя (комментарий добавлен после ввода нижних строк)
70 INPUT "Имя:  ",N$
80 INPUT "Фамилия: ",S$
90 INPUT "Отчество: ",T$
95 'Вырезаем копию первых символов из имени и отчества
100 N2$=LEFT$(N$,1)
110 T2$=LEFT$(T$,1)
120 'Выводим результат
130 PRINT "Ваше имя кратко: ";S$;" ";N2$;". ";T2$;"."
140 INPUT "Повторить программу? (Y/N) ",U$
150 IF U$="Y" THEN GOTO 10
160 END
```

3. Основные современные парадигмы и технологии программирования

Аппликативная парадигма – базируется на идее рассмотрения функции, которую выполняет программа. Другими словами, какую функцию необходимо применить к начальному состоянию вычислительной машины, чтобы получить желаемый результат?

Языки, поддерживающие такую парадигму, называются **аппликативными** или **функциональными**. Их синтаксис, как правило, выглядит следующим образом:

Функция_n (... функция_2 (функция_1 (данные))...)

К функциональным относятся такие языки, как Haskell, Lisp, ML и др.

3. Основные современные парадигмы и технологии программирования

Примеры программ на аппликативных языках программирования

Haskell

```
calc :: String -> Float
 calc = head . foldl f [] . words
   where
     f :: [Float] -> String -> [Float]
     f (x:y:zs) "+"    = (y + x):zs
     f (x:y:zs) "-"    = (y - x):zs
     f (x:y:zs) "*"    = (y * x):zs
     f (x:y:zs) "/"    = (y / x):zs
     f (x:y:zs) "FLIP" = y:x:zs
     f (x:zs)   "ABS"  = (abs x):zs
     f xs       y      = read y : xs
```

Lisp

```
(SmartPlus 1 2)
==> 3
(type-of (SmartPlus 1 2 ))
==> (INTEGER 0
461168601842738790 3)
(SmartPlus 1 1.2)
==> 2.2
(type-of (SmartPlus 1 1.2))
==> SINGLE-FLOAT
(SmartPlus 2 2/3)
==> 8/3
(type-of (SmartPlus 2 2/3))
==> RATIO
(SmartPlus "abc" 20)
==> NIL
(type-of (SmartPlus "abc" 20))
==> NULL
```

3. Основные современные парадигмы и технологии программирования

Парадигма, основанная на системе правил. Языки, основанные на этой парадигме, осуществляют проверку наличия необходимого разрешающего условия и в случае его обнаружения выполняют соответствующее действие.

Операторы в программе, в случае использования этой парадигмы, выполняются не в той последовательности, в которой они определены в программе. Порядок выполнения определяют разрешающие условия. Синтаксис таких языков выглядит следующим образом:

условие_1 –> действие_1 условие_2 –> действие_2 ...
условие_n –> действие_n

Наиболее известным языком, основанным на системе правил, является язык логического программирования Prolog.

3. Основные современные парадигмы и технологии программирования

Пример программы на логическом языке программирования Prolog

```
parent("Tom","Jake").
parent("Janna","Jake").
parent("Tom","Tim").
male("Tom").
male("Tim").
male("Jake").
female("Janna").

brother(X,Y):-
parent(Z,X),parent(Z,Y),male(X),male(Y),X\=Y.
```

3. Основные современные парадигмы и технологии программирования

Объектно-ориентированная парадигма. В этой модели строятся сложные объекты данных. Для операций над ними определяется некоторый ограниченный набор методов. Создаваемые объекты могут наследовать свойства более простых объектов. Благодаря такой возможности объектно-ориентированные программы имеют высокую эффективность, свойственную программам, написанным на императивных языках. Возможность разработки различных классов, которые используют ограниченный набор объектов данных, обусловливает гибкость и надежность, которые свойственны аппликативному языку.

К объектно-ориентированным языкам относятся Java, C#, Python, C++ (не в полной мере) и др.

3. Основные современные парадигмы и технологии программирования

Пример программы на объектно-ориентированном языке C#

```
// assembly: System.dll
// assembly: System.Drawing.dll
// assembly: System.Windows.Forms.dll
using System;
using System.Drawing;
using System.Windows.Forms;

namespace WindowsForms
{
    public class Program
    {
        [STAThread]
        public static void Main()
        {
            new DemoForm().ShowDialog();
        }
    }
```

```
public class DemoForm : Form
    {
        Label label = new Label();

        public DemoForm()
        {
            label.Text = "Hello World!";
            this.Controls.Add(label);
            this.StartPosition =
                    FormStartPosition.CenterScreen;
            this.BackColor = Color.White;
            this.FormBorderStyle =
                    FormBorderStyle.Fixed3D;
        }
    }
}
```

3. Основные современные парадигмы и технологии программирования

На ранних этапах развития программирования программы писались в виде последовательностей машинных команд (машинного кода) и какие-либо технологии программирования отсутствовали.

Пример программа «Hello, world!» для процессора архитектуры x86
(в шестнадцатеричном формате):

BB 11 01 B9 0D 00 B4 0E 8A 07 43 CD 10 E2 F9 CD 20 48 65 6C 6C 6F 2C 20 57 6F 72 6C 64 21

Позже для разработки ПО начал использоваться машинно-зависимый низкоуровневый язык программирования ассемблер.
Пример этой же программы на языке ассемблера для операционной системы Linux:

```
ection     .text
global      _start                          ;must be declared for linker (ld)
_start:                                     ;tell linker entry point
    mov     edx,len                         ;message length
    mov     ecx,msg                         ;message to write
    mov     ebx,1                           ;file descriptor (stdout)
    mov     eax,4                           ;system call number (sys_write)
    int     0x80                            ;call kernel
    mov     eax,1                           ;system call number (sys_exit)
    int     0x80                            ;call kernel
section     .data
msg     db  'Hello, world!',0xa             ;our dear string
len     equ $ - msg                         ;length of our dear string
```

3. Основные современные парадигмы и технологии программирования

С увеличением размеров программ стали выделять их обособленные части и оформлять их как **подпрограммы**. Часть таких подпрограмм объединялась в **библиотеки**, из которых подпрограммы можно было включать в рабочие программы и затем вызывать из рабочих программ. Это положило начало **процедурному программированию** – большая программа представлялась совокупностью процедур-подпрограмм. Одна из подпрограмм являлась главной и с нее начиналось выполнение программы.

Процедурный подход потребовал структурирования будущей программы, разделения ее на отдельные процедуры. При разработке отдельной процедуры о других процедурах требовалось знать только их назначение и способ вызова. Появилась возможность перерабатывать отдельные процедуры, не затрагивая остальной части программы, сокращая при этом затраты труда и машинного времени на разработку и модернизацию программ.

К первым языкам, поддерживающим процедурный подход, относятся Fortran, Algol, Cobol и др.

3. Основные современные парадигмы и технологии программирования

Следующим шагом в углублении структурирования программ стало **структурное программирование**, при котором программа рассматривались как последовательности канонических структур: **линейных участков**, состоящих из последовательно выполнявшихся операторов, **циклов** и **разветвлений**.

Благодаря отказу от использования оператора безусловного перехода (goto) и использования специальных стилей форматирования появилась возможность читать и проверять исходный текст программы как последовательный текст, что повысило производительность труда программистов при разработке, отладке и сопровождении программ.

К языкам, поддерживающим структурное программирование, относятся C, C++, Pascal, Python и др.

3. Основные современные парадигмы и технологии программирования

Все **универсальные языки программирования**, несмотря на различия в синтаксисе и используемых ключевых словах, реализуют одни и те же канонические структуры: операторы присваивания, циклы и разветвления. Во всех современных языках присутствуют **базовые типы данных** (целые и вещественные арифметические типы, строковые и т. п.), имеется возможность использования агрегатов данных, в том числе массивов и структур (записей).

Однако, при разработке ПО для решения конкретной прикладной задачи желательно наличие большей концептуальной близости текста программы к описанию задачи. Этого можно добиться двумя основными способами:

1) созданием языка программирования (**языка-оболочки**), содержащего как можно больше типов данных, и выбором для каждого класса задач некоторого подмножества этого языка (например, PL/1);

2) построением расширяемого языка (**языка-ядра**), содержащего небольшое допускающего расширение ядро, дополняющее язык типами данных и операторами, отражающими концептуальную сущность конкретного класса задач (C++).

3. Основные современные парадигмы и технологии программирования

Объектно-ориентированный подход (ООП) к программированию базируется на следующих основные идеях:

• программа представляет собой модель некоторого реального процесса, части реального мира;

• модель реального мира или его части может быть описана как совокупность взаимодействующих между собой объектов;

• объект описывается набором параметров, значения которых определяют состояние объекта, и набором операций (действий), которые может выполнять объект;

• взаимодействие между объектами осуществляется посылкой специальных сообщений от одного объекта к другому, при этом сообщение, полученное объектом, может потребовать выполнения определенных действий, например, изменения состояния объекта;

• объекты, описанные одним и тем же набором параметров и способные выполнять один и тот же набор действий, представляют собой **класс** однотипных объектов.

ООП предполагает, что при разработке программы должны быть определены классы используемых в программе объектов и построены их описания, затем созданы экземпляры необходимых объектов и определено взаимодействие между ними.

К языкам, поддерживающим ООП, относятся C++, C#, Obejct Pascal, Python и др.

3. Основные современные парадигмы и технологии программирования

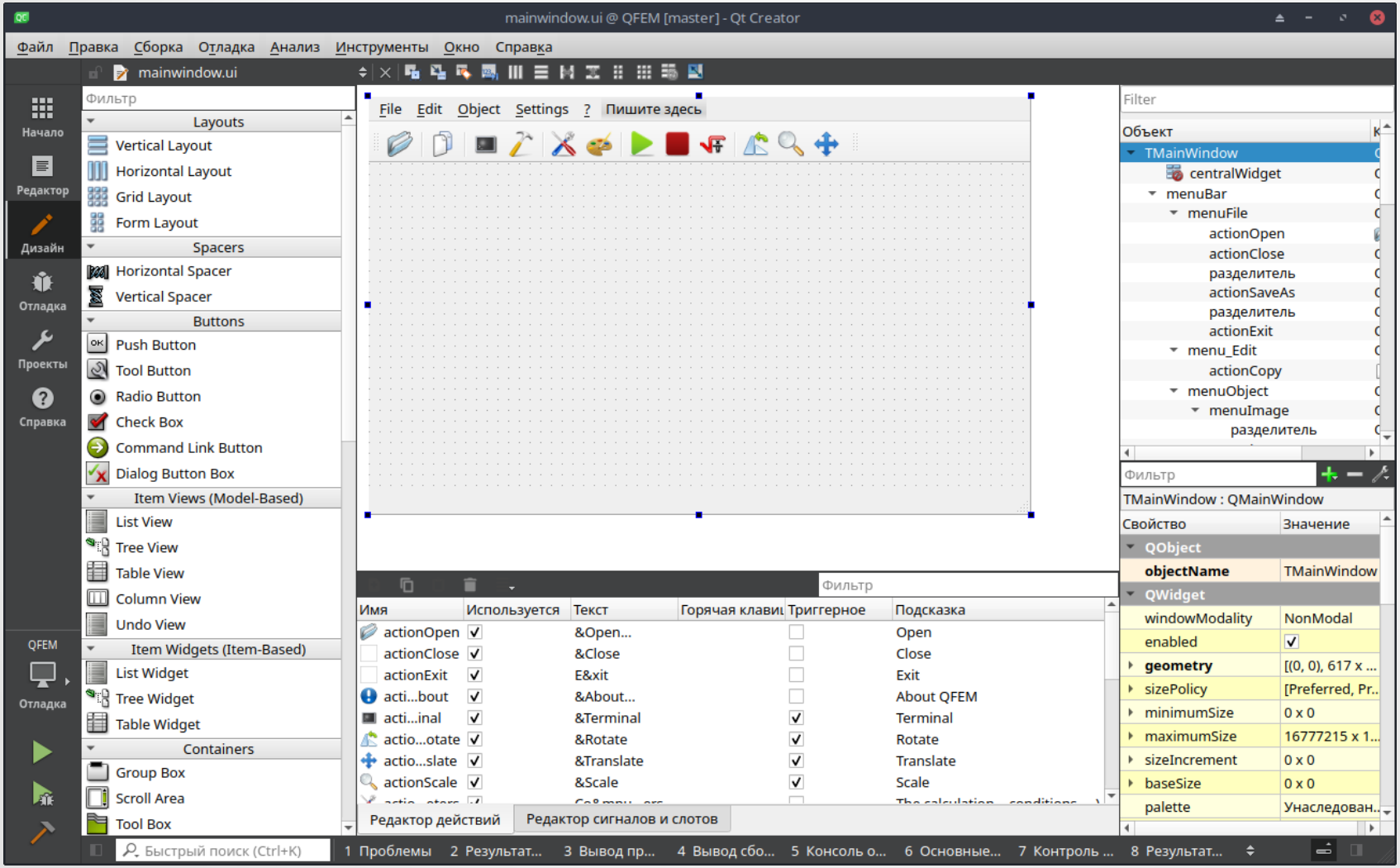
Визуальное программирование является в настоящее. время одной из наиболее популярных технологий программирования. Оно заключается в автоматизированной разработке с использованием специального инструментария графического интерфейса пользователя (GUI – Graphical User Interface) для программ, работающих в среде ОС с графическим интерфейсом (например, Windows или Linux).

На начальном этапе визуального программирования, как правило, создаются экранные формы, соответствующие главному и дочерним окнам программы, а также определяются их свойства и поведение. Затем, с использованием объектно-ориентированного программирования, реализуется функциональная составляющая программы (ее бизнес-логика).

К наиболее известным средам визуального программирования относятся такие коммерческие системы, как Microsoft Visual Studio, Embarcadero RAD Studio, бесплатные Qt Creator, Eclipse и др.

3. Основные современные парадигмы и технологии программирования

Пример использования среды Qt Designer для разработки GUI программы



4. Литература

1. Камаев В.А. Технологии программирования: Учебник / В.А. Камаев, В.В. Костерин. – 2-е изд., перераб. и доп. – М.: Высш. шк., 2006. – 454 с.
2. Прата С. Язык программирования С: Лекции и упражнения. – М.: Вильямс, 2006. – 960 с.
3. Страуструп Б. Язык программирования С++. Специальное издание. Пер. с англ. – М. : Издательство Бином, 2011. – 1136 с.
4. Скотт М. Эффективный и современный С++: 42 рекомендации по использованию С++ 11 и С++14.: Пер. с англ. – М.: ООО "ИЛ. Вильяме", 2016. – 304 с.
5. Шлее М. Qt 4.5. Профессиональное программирование на С++. – Спб. : БХВ-Петербург, 2010. – 896 с.