

Ch. 5 从需求到设计

2017.4

§ 5.0 过程与环节

- 从“做什么”到“怎么做”
- 从“怎么做”到“做出来”
- 从“做出来”到“行不行”

目录

- § 5.1 从一份需求开始
- § 5.2 软件的设计思路

从一份需求开始

目 录

1 引言

1. 1编写目的

1. 3定义

1.3.1 关键字

1.3.2 术语

1. 4参考资料

2 任务概述

2. 1目标

3 功能需求

4. 性能需求

4.1精度

4.2时间特性要求

4.3 启动关闭性能

4.4故障处理要求

4.4.1 软件故障

4.4.2 硬件故障

5 运行需求

5.1 用户界面

5.2硬件环境

5.3软件环境

5.3.1 操作系统

5.3.2 开发环境

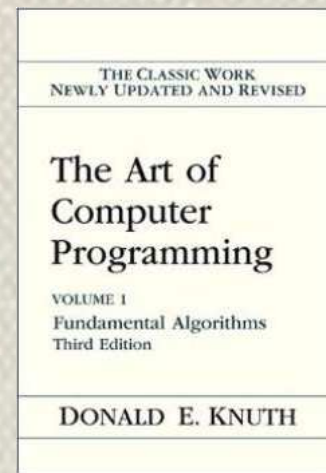
5.3.3 接口通信

Design

- **Design** is the creation of a plan or convention for the construction of an object, system or measurable human interaction (as in architectural blueprints, engineering drawings, business processes, circuit diagrams, and sewing patterns). Design has different connotations in different fields (see design disciplines below). In some cases, the direct construction of an object (as in engineering, management, coding, and graphic design) is also considered to use design thinking.

什么是设计

- 设计：指[设计师](#)有目标有计划的进行技术性的创作与创意活动。
- 设计的任务不只是为生活和商业服务，同时也伴有艺术性的创作。

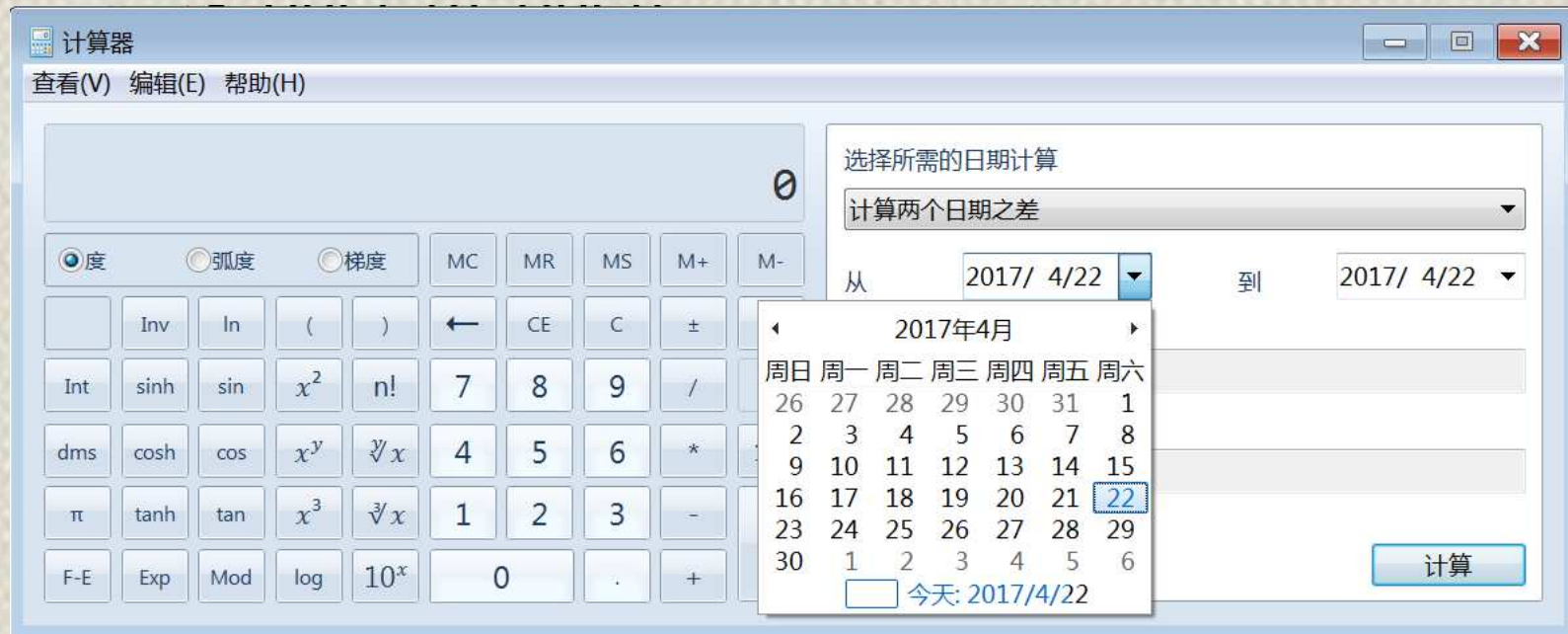


为什么要做设计？

- 需求过程本身就是设计
- 需求要被整理
- 需求需要转化（什么是模态对话框？）
- 用一些固定的设计更清晰、高效
- 一般性问题和特殊性问题
- 项目总是有些特殊问题，特殊问题怎么解决？

特殊问题

- 什么是特殊问题？
- $op(a, b); op(a)$



都设计什么？

结构设计：定义软件系统的整体结构，

数据设计：数据结构、数据库、文件的定义。

过程设计：把结构成份（模块）转换成软件的过程性描述

UI/UE设计：是对系统边界的描述，是用户和系统进行交互的工具。

界定清楚

- 产品设计： 着重于用户体验方面的设计
- 技术设计： 技术方案、数据结构、代码结构等

是不是一定要设计？

- 某些问题其实可以不必进行设计
- 根据经验动手先做，经验来源
- 不断优化
- 其实是有设计过程的

设计会不会发生变化？

- 确定后不再进行变化
- 一直在变化，不稳定的需求
- 需要本身要求具有较强的扩展性
- 潜在的变化

目录

- § 5.1 从一份需求开始
- § 5.2 软件的设计思路

软件的设计思路

- 自顶向下与自底向上
- 模块化的设计思路

模块化思维

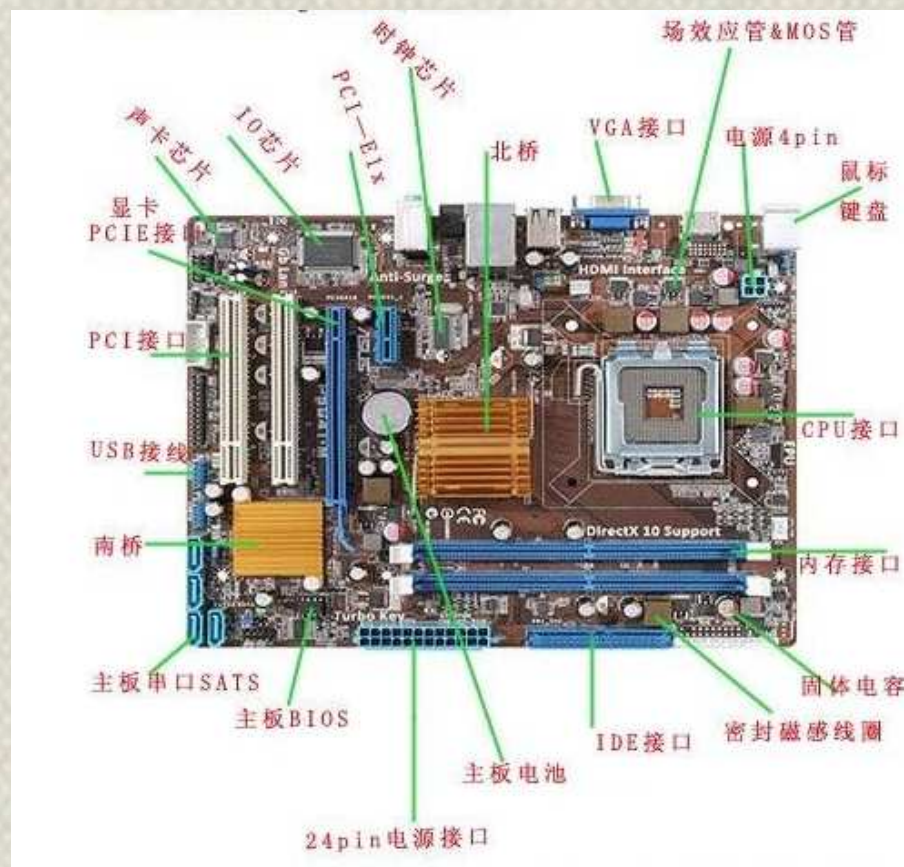
- 整体化思维与模块化思维



模块化



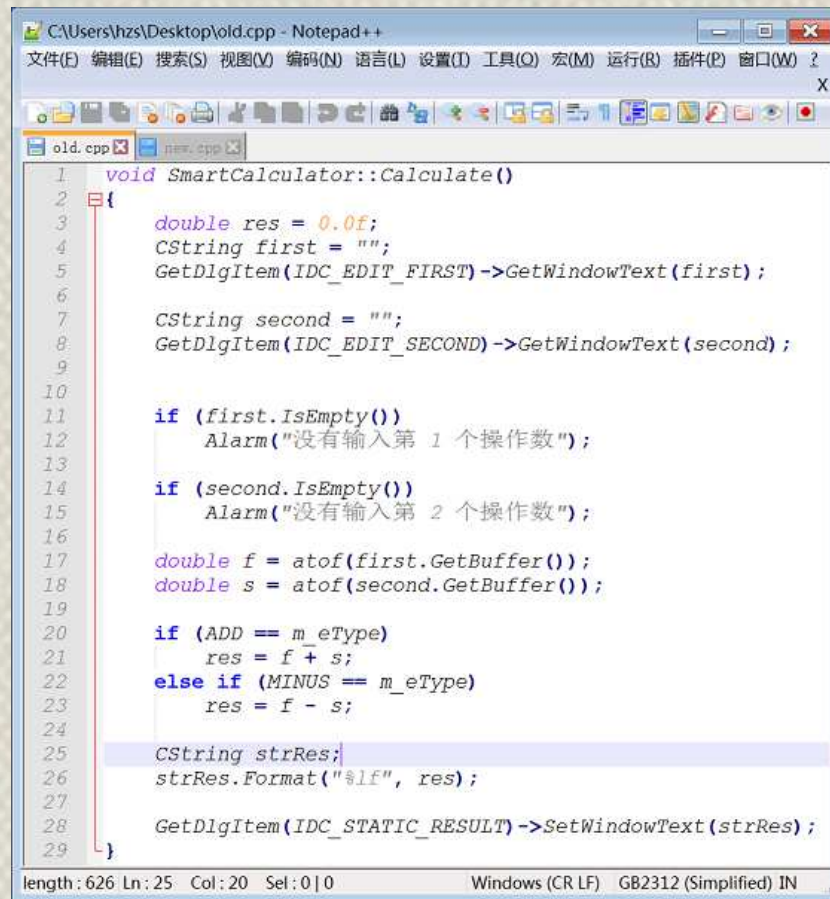
模块化



软件模块化方法

- 函数（过程）
- 面向对象的封装
- 源码包
- 静态库、动态库
- 组件

未经处理



```
1 void SmartCalculator::Calculate()
2 {
3     double res = 0.0f;
4     CString first = "";
5     GetDlgItem(IDC_EDIT_FIRST)->GetWindowText(first);
6
7     CString second = "";
8     GetDlgItem(IDC_EDIT_SECOND)->GetWindowText(second);
9
10
11     if (first.IsEmpty())
12         Alarm("没有输入第 1 个操作数");
13
14     if (second.IsEmpty())
15         Alarm("没有输入第 2 个操作数");
16
17     double f = atof(first.GetBuffer());
18     double s = atof(second.GetBuffer());
19
20     if (ADD == m_eType)
21         res = f + s;
22     else if (MINUS == m_eType)
23         res = f - s;
24
25     CString strRes;
26     strRes.Format("%lf", res);
27
28     GetDlgItem(IDC_STATIC_RESULT)->SetWindowText(strRes);
29 }
```

length: 626 Ln: 25 Col: 20 Sel: 0|0 Windows (CR LF) GB2312 (Simplified) IN

函数封装

```
C:\Users\hzs\Desktop\new.cpp - Notepad++
文件(F) 编辑(E) 搜索(S) 视图(V) 编码(N) 语言(L) 设置(O) 工具(T)
宏(M) 运行(R) 插件(P) 窗口(W) ?

old.cpp new.cpp
1 void SmartCalculator::Calculate()
2 {
3     double res = 0.0f;
4     double f = this->getFirst();
5     double s = this->getSecond();
6
7     if (ADD == m_eType)
8         res = f + s;
9     else if (MINUS == m_eType)
10        res = f - s;
11
12    out(res);
13 }
```

```
C:\Users\hzs\Desktop\new.cpp - Notepad++
文件(F) 编辑(E) 搜索(S) 视图(V) 编码(N) 语言(L) 设置(O) 工具(T) 宏(M) 运行(R) 插件(P) 窗口(W) ?
窗口(W) 2

old.cpp new.cpp
15 double SmartCalculator::getFirst()
16 {
17     CString first = "";
18     GetDlgItem(IDC_EDIT_FIRST)->GetWindowText(first);
19     if (first.IsEmpty())
20         Alarm("没有输入第 1 个操作数");
21     return atof(first.GetBuffer());
22 }
```

```
C:\Users\hzs\Desktop\new.cpp - Notepad++
文件(F) 编辑(E) 搜索(S) 视图(V) 编码(N) 语言(L) 设置(O) 工具(T) 宏(M) 运行(R) 插件(P) 窗口(W) ?
窗口(W) 2

old.cpp new.cpp
24 double SmartCalculator::getSecond()
25 {
26     CString second = "";
27     GetDlgItem(IDC_EDIT_SECOND)->GetWindowText(second);
28     if (second.IsEmpty())
29         Alarm("没有输入第 2 个操作数");
30     return atof(second.GetBuffer());
31 }
```

```
C:\Users\hzs\Desktop\new.cpp - Notepad++
文件(F) 编辑(E) 搜索(S) 视图(V) 编码(N) 语言(L) 设置(O) 工具(T) 宏(M) 运行(R) 插件(P) 窗口(W) ?
窗口(W) 2

old.cpp new.cpp
32 void out(double res)
33 {
34     CString strRes;
35     strRes.Format("%lf", res);
36     GetDlgItem(IDC_STATIC_RESULT)->SetWindowText(strRes);
37 }
```


多态设计

```
Calculator.java Operation.java Add.java Minus.java
1
2
3 public class Calculator
4 {
5     static Operation op = null;
6
7     public static void main(String[] args)
8     {
9         double f = 100.0f;
10        double s = 20.0f;
11
12        double value = op.calculate(f, s);
13
14        System.out.println(value);
15    }
16
17    static void ButtonDownAdd()
18    {
19        op = new Add();
20    }
21
22    static void ButtonDownMinus()
23    {
24        op = new Minus();
25    }
26 }
```

```
Calculator.java Operation.java Add.java Minus.java
1 public interface Operation
2 {
3     public double calculate(double f, double s);
4 }
```

```
Calculator.java Operation.java Add.java Minus.java
1 public class Add implements Operation
2 {
3     public Add(){}
4     public double calculate(double f, double s)
5     {
6         return f + s;
7     }
8 }
```

```
Calculator.java Operation.java Add.java Minus.java
1 public class Minus implements Operation
2 {
3     public Minus(){}
4     public double calculate(double f, double s)
5     {
6         return f - s;
7     }
8 }
```

模块化思路

- 本身不是模块化的
- 可以用模块化的方式去工作