日常记录及信息分享服务软件

软件代码规范说明



**课 程**：\_\_\_\_ \_ 软件工程 \_\_ \_\_\_\_ \_\_\_

**题 目**：\_ 基于App的类博客式分享交流软件设计与开发 \_

**专业班级**：\_\_\_\_ \_ \_\_软件工程 1802\_\_\_\_\_\_ \_\_\_

**组 长**：\_\_\_\_\_ \_ \_ G13-陈玲曦-31801349\_ \_\_\_\_ \_\_\_

**组 员**：\_\_\_\_ \_ \_\_ G13-童峻涛-31801341\_\_ \_\_ \_\_\_\_

**组 员**：\_\_\_\_ \_ \_ G13-刘书宇-31801323\_\_ \_ \_ \_\_

|  |  |  |
| --- | --- | --- |
| 文件状态：  [ ] 草稿  [ ] 正式发布  [√] 正在修改 | 文件标识： | SE2020-G13-软件总体设计说明书 |
| 当前版本： | 0.1.1 |
| 作者： | 陈玲曦、童峻涛、刘书宇 |
| 完成日期： | 2020-12-16 |

软件代码规范说明

说明：

《软件代码规范说明》 描述了软件在编程开发过程中所涉及的代码拜尼马格式规范以及注释规范。该文档可能还要用《软件测试报告》和《软件代码走查说明》以补充。

**版本记录**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 编号 | 修订日期 | 版本/状态 | 修订人 | 发布日期 | 备注 |
| 01 | 2020-1216 | 0.1.1 | 童峻涛 | 2020-12-16 | 报告的初始版本制作 |
|  |  |  |  |  |  |

目录

[软件代码规范说明 2](#_Toc59024033)

[1引言 5](#_Toc59024034)

[1.1标识 5](#_Toc59024035)

[1.2系统概述 5](#_Toc59024036)

[1.3文档概述 5](#_Toc59024037)

[1.4基线 6](#_Toc59024038)

[2引用文件 6](#_Toc59024039)

[3代码规范细则 7](#_Toc59024040)

[3.1排版要求 7](#_Toc59024041)

[3.1.1程序块缩进 7](#_Toc59024042)

[3.1.2程序块之间空行 7](#_Toc59024043)

[3.1.3长语句与长表达式 7](#_Toc59024044)

[3.1.4循环、判断等长表达式或语句 7](#_Toc59024045)

[3.1.5语句对齐 7](#_Toc59024046)

[3.1.6函数、过程和结构等语句块 7](#_Toc59024047)

[3.1.7操作符前后空格 7](#_Toc59024048)

[3.2注释 8](#_Toc59024049)

[3.2.1有效注释量 8](#_Toc59024050)

[3.2.2说明性文件 8](#_Toc59024051)

[3.2.3注释与代码一致 8](#_Toc59024052)

[3.2.4变量、常量注释 8](#_Toc59024053)

[3.2.5注释与代码之间空行 8](#_Toc59024054)

[3.2.6其他 8](#_Toc59024055)

[3.3标识符命名 8](#_Toc59024056)

[3.3.1命名清晰 8](#_Toc59024057)

[3.3.2 特殊命名需注释 8](#_Toc59024058)

[3.3.3命名规范与系统风格一致 8](#_Toc59024059)

[3.3.4其他 8](#_Toc59024060)

[3.4结构 9](#_Toc59024061)

[3.4.1可读性 9](#_Toc59024062)

[3.4.2公共变量说明、赋值 9](#_Toc59024063)

[3.5可测性 9](#_Toc59024064)

[3.5.1调试开关 9](#_Toc59024065)

[3.5.2打印信息 9](#_Toc59024066)

[3.5.3单元测试 9](#_Toc59024067)

[3.5.4集成测试 9](#_Toc59024068)

[3.5.5版本维护 9](#_Toc59024069)

[3.5.6其他 9](#_Toc59024070)

[4注解 9](#_Toc59024071)

[附录 10](#_Toc59024072)

**根据《GB8567－06计算机软件产品开发文件编制指南》中项目实现阶段结合实际情况调整后的《软件代码规范说明》内容如下：**

# 1引言

## 1.1标识

标题：《软件代码规范说明》

版本号：0.1.1

说明：本文档支持Office、WPS等word文档阅读软件

## 1.2系统概述

1. **软件概述**

我们小组开发的记录分享软件（Day）能够在手机上记录生活的点点滴滴，同时可以发布分享自己的遇到的有趣的新鲜事，收藏、点赞别人的动态。考虑到用户使用的积极性与粘度，我们计划加入打卡模块，通过在线阅读、点赞、发表等完成基本任务打卡，根据排行月榜提供专属奖励。

1. **项目概述**
2. 项目名称：移动端记录分享服务型App应用程序（Day）
3. 项目用途：在学生繁忙的生活中提供一些便利与休闲娱乐
4. 任务提出者：杨枨老师
5. 项目开发者：陈玲曦、刘书宇、童峻涛
6. 用户：浙大城市学院学生
7. 课程名称：《软件工程》
8. 承办小组：G13小组
9. 相关文档：项目的提出和介绍；软件项目计划；软件可行性分析报告；软件需求分析报告；软件总体设计报告；软件详细设计报告；软件测试报告；项目总结报告。

## 1.3文档概述

本文档的内容包括需求、合格性规定、需求可追踪性等，其中对需求模块进行详细分析。

本文档的使用除了项目开发团队的内部使用外，还需提交用户和客户组织负责人审查批准。

本文档的使用应遵守国家先关法律法规，未经允许不得对外公开，需要有一定的保密性和私密性。

## 1.4基线

基线是软件生存期各个开发阶段的工作成果。

重要的检查点是里程碑，重要的需要客户确认的里程碑是基线。

主要基线如下表所示：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 基线名称 | 基线缩写 | 基线标识 | 计划建立日期 | 计划完成时间 |
| 软件需求基线 | SRBL | SR | 2020/11/07 | 2020/11/18 |
| 概要设计基线 | PDBL | PD | 2020/11/19 | 2020/11/26 |
| 详细设计基线 | DDBL | DD | 2020/11/27 | 2020/12/03 |
| 代码基线 | SCBL | SC | 2020/12/04 | 2020/12/20 |
| 测试基线 | STBL | ST | 2020/12/21 | 2020/12/30 |
| 运行基线 | PRBL | PR | 2020/12/31 | 2021/01/13 |

# 2引用文件

[1]《GB T-8567-2006计算机软件文档编制规范》

[2]张海藩，牟永敏.软件工程导论(第6版)[M]北京：清华大学出版社

[3]梁勇. JAVA语言程序设计(第10版)[M]北京：机械工业出版社

[4]《Java语言编程规范--华为技术有限公司》

[5]《Flutter中文网开发者代码规范》. [EB/OL]. <https://flutterchina.club/> --2020/12/16

# 3代码规范细则

## 3.1排版要求

### 3.1.1程序块缩进

程序块要采用缩进风格编写，缩进的空格数为4个。

说明：对于由开发工具自动生成的代码可以有不一致。

### 3.1.2程序块之间空行

相对独立的程序块之间、变量说明之后必须加空行。

示例：如下例子不符合规范。

1. **if** (!valid(ni))
2. {
3. … // program code
4. }
5. String ni = " ";

应如下书写：

1. **if** (!valid(ni))
2. {
3. … // program code
4. }
6. String ni = " ";

### 3.1.3长语句与长表达式

较长的语句（>80字符）要分成多行书写，长表达式要在低优先级操作符处划分新行，操作符放在新行之首，划分出的新行要进行适当的缩进，使排版整齐，语句可读。

1. perm\_count\_msg.head.len = NO7\_TO\_STAT\_PERM\_COUNT\_LEN
2. + STAT\_SIZE\_PER\_FRAM \* sizeof( \_UL );
4. act\_task\_table[frame\_id \* STAT\_TASK\_CHECK\_NUMBER + index].occupied
5. = stat\_poi[index].occupied;
7. act\_task\_table[taskno].duration\_true\_or\_false
8. = SYS\_get\_sccp\_statistic\_state( stat\_item );
10. report\_or\_not\_flag = ((taskno < MAX\_ACT\_TASK\_NUMBER)
11. && (n7stat\_stat\_item\_valid (stat\_item))
12. && (act\_task\_table[taskno].result\_data != 0));

### 3.1.4循环、判断等长表达式或语句

循环、判断等语句中若有较长的表达式或语句，则要进行适应的划分，长表达式要在低优先级操作符处划分新行，操作符放在新行之首。

1. **if** ((taskno < max\_act\_task\_number)
2. && (n7stat\_stat\_item\_valid (stat\_item)))
3. {
4. … // program code
5. }
7. **for** (i = 0, j = 0; (i < BufferKeyword[word\_index].word\_length)
8. && (j < NewKeyword.word\_length); i++, j++)
9. {
10. … // program code
11. }
13. **for** (i = 0, j = 0;
14. (i < first\_word\_length) && (j < second\_word\_length);
15. i++, j++)
16. {
17. … // program code
18. }

### 3.1.5语句对齐

对齐只使用空格键，不使用TAB键。

说明：以免用不同的编辑器阅读程序时，因TAB键所设置的空格数目不同而造成程序布局不整齐，

不要使用BC作为编辑器合版本，因为BC会自动将8个空格变为一个TAB键，因此使用BC合入的版本大多会将缩进变乱。

### 3.1.6函数、过程和结构等语句块

函数或过程的开始、结构的定义及循环、判断等语句中的代码都要采用缩进风格，case语句下的情况处理语句也要遵从语句缩进要求。

### 3.1.7操作符前后空格

在两个以上的关键字、变量、常量进行对等操作时，它们之间的操作符之前、之后或者前后要加空格；进行非对等操作时，如果是关系密切的立即操作符（如－>），后不应加空格。

说明：采用这种松散方式编写代码的目的是使代码更加清晰。

由于留空格所产生的清晰性是相对的，所以，在已经非常清晰的语句中没有必要再留空格，如果语句已足够清晰则括号内侧(即左括号后面和右括号前面)不需要加空格，多重括号间不必加空格，因为在C/C++语言中括号已经是最清晰的标志了。

在长语句中，如果需要加的空格非常多，那么应该保持整体清晰，而在局部不加空格。给操作符留空格时不要连续留两个以上空格。

**示例：**

(1) 逗号、分号只在后面加空格。

1. **int** a, b, c;

(2)比较操作符, 赋值操作符 ‘=’、 ‘+=’，算术操作符’+’、’%’，逻辑操作符 ’&&’、’&’，位域操作符 ’<<’、’^’ 等双目操作符的前后加空格。

1. **if** (current\_time >= MAX\_TIME\_VALUE)
2. a = b + c;
3. a \*= 2;
4. a = b ^ 2;

(3) ’!’、’~’、’++’、’–‘、’&’（地址运算符）等单目操作符前后不加空格。

1. p = ‘a’; // 内容操作”“与内容之间
2. flag = !isEmpty; // 非操作”!”与内容之间
3. p = &mem; // 地址操作”&” 与内容之间
4. i++; // “++”,”–”与内容之间

(4)’->’、’.’前后不加空格。

1. p->id = pid; // “->”指针前后不加空格

(5) if、for、while、switch等与后面的括号间应加空格，使if等关键字更为突出、明显。

1. **if** (a >= b && c > d)

## 3.2注释

### 3.2.1有效注释量

一般情况下，源程序有效注释量必须在20％以上。

说明：注释的原则是有助于对程序的阅读理解，在该加的地方都加了，注释不宜太多也不能太少，注释语言必须准确、易懂、简洁。

### 3.2.2说明性文件

说明性文件（如头文件.h文件、.inc文件、.def文件、编译说明文件.cfg等）头部应进行注释，注释必须列出：版权说明、版本号、生成日期、作者、内容、功能、与其它文件的关系、修改日志等，头文件的注释中还应有函数功能简要说明。

示例：下面这段头文件的头注释比较标准，当然，并不局限于此格式，但上述信息建议要包含在内。

1. /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*
2. Copyright (C), 1996-1998, xxxxx. Co., Ltd.
3. File name: // 文件名
4. Author: Version: Date: // 作者、版本及完成日期
5. Description: // 用于详细说明此程序文件完成的主要功能，与其他模块
6. // 或函数的接口，输出值、取值范围、含义及参数间的控
7. // 制、顺序、独立或依赖等关系
8. Others: // 其它内容的说明
9. Function List: // 主要函数列表，每条记录应包括函数名及功能简要说明
10. 1. ….
11. History: // 修改历史记录列表，每条修改记录应包括修改日期、修改
12. // 者及修改内容简述
13. 1. Date:
14. Author:
15. Modification:
16. 2. …
17. \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

### 3.2.3注释与代码一致

边写代码边注释，修改代码同时修改相应的注释，以保证注释与代码的一致性。不再有用的注释要删除。

### 3.2.4变量、常量注释

对于所有有物理含义的变量、常量，如果其命名不是充分自注释的，在声明时都必须加以注释，说明其物理含义。变量、常量、宏的注释应放在其上方相邻位置或右方。

示例：

1. /\* active statistic task number \*/
2. define MAX\_ACT\_TASK\_NUMBER 1000
3. define MAX\_ACT\_TASK\_NUMBER 1000 /\* active statistic task number \*/

### 3.2.5注释与代码之间空行

将注释与其上面的代码用空行隔开。

示例：如下例子，显得代码过于紧凑。

1. /\* code one comments \*/
2. program code one
3. /\* code two comments \*/
4. program code two

应如下书写:

1. /\* code one comments \*/
2. program code one
4. /\* code two comments \*/
5. program code two

### 3.2.6其他

1. 避免在一行代码或表达式的中间插入注释。

说明：除非必要，不应在代码或表达中间插入注释，否则容易使代码可理解性变差。

2. 通过对函数或过程、变量、结构等正确的命名以及合理地组织代码的结构，使代码成为自注释的。

说明：清晰准确的函数、变量等的命名，可增加代码可读性，并减少不必要的注释。

3. 在代码的功能、意图层次上进行注释，提供有用、额外的信息。

说明：注释的目的是解释代码的目的、功能和采用的方法，提供代码以外的信息，帮助读者理解代码，防止没必要的重复注释信息。

示例：如下注释意义不大。

1. /\* if receive\_flag is TRUE \*/
2. **if** (receive\_flag)

而如下的注释则给出了额外有用的信息。

1. /\* if mtp receive a message from links \*/
2. **if** (receive\_flag)

4. 注释格式尽量统一，建议使用“/\* …… \*/”。

5. 注释应考虑程序易读及外观排版的因素，使用的语言若是中、英兼有的，建议多使用中文，除非能用非常流利准确的英文表达。

说明：注释语言不统一，影响程序易读性和外观排版，出于对维护人员的考虑，建议使用中文。

## 3.3标识符命名

### 3.3.1命名清晰

标识符的命名要清晰、明了，有明确含义，同时使用完整的单词或大家基本可以理解的缩写，避免使人产生误解。

小组开发过程中采用驼峰命名法。

说明：较短的单词可通过去掉“元音”形成缩写；较长的单词可取单词的头几个字母形成

缩写；一些单词有大家公认的缩写。

示例：如下单词的缩写能够被大家基本认可。

temp 可缩写为 tmp ;

flag 可缩写为 flg ;

statistic 可缩写为 stat ;

increment 可缩写为 inc ;

message 可缩写为 msg ;

### 3.3.2 特殊命名需注释

命名中若使用特殊约定或缩写，则要有注释说明。

说明：应该在源文件的开始之处，对文件中所使用的缩写或约定，特别是特殊的缩写，进行必要的注释说明。

### 3.3.3命名规范与系统风格一致

自己特有的命名风格，要自始至终保持一致，不可来回变化。

说明：个人的命名风格，在符合所在项目组或产品组的命名规则的前提下，才可使用。（即命名规则中没有规定到的地方才可有个人命名风格）。

### 3.3.4其他

1. 除非必要，不要用数字或较奇怪的字符来定义标识符。

2. 在同一软件产品内，应规划好接口部分标识符（变量、结构、函数及常量）的命名，防止编译、链接时产生冲突。

说明：对接口部分的标识符应该有更严格限制，防止冲突。如可规定接口部分的变量与常量之前加上“模块”标识等。

3. 用正确的反义词组命名具有互斥意义的变量或相反动作的函数等。

4. 除了编译开关/头文件等特殊应用，应避免使用EXAMPLE\_TEST之类以下划线开始和结尾的定义。

## 3.4结构

### 3.4.1可读性

1. 注意运算符的优先级，并用括号明确表达式的操作顺序，避免使用默认优先级。

说明：防止阅读程序时产生误解，防止因默认的优先级与设计思想不符而导致程序出错。

2. 避免直接使用数字作为标识符

避免使用不易理解的数字，用有意义的标识来替代。涉及物理状态或者含有物理意义的常量，不应直接使用数字，必须用有意义的枚举或宏来代替。

### 3.4.2公共变量说明、赋值

1. 仔细定义并明确公共变量的含义、作用、取值范围及公共变量间的关系。

说明：在对变量声明的同时，应对其含义、作用及取值范围进行注释说明，同时若有必要还应说明与其它变量的关系。

2. 当向公共变量传递数据时，要十分小心，防止赋与不合理的值或越界等现象发生。

说明：对公共变量赋值时，若有必要应进行合法性检查，以提高代码的可靠性、稳定性。

## 3.5可测性

### 3.5.1调试开关

在同一项目组或产品组内，要有一套统一的为集成测试与系统联调准备的调测开关及相应打印函数，并且要有详细的说明。

说明：本规则是针对项目组或产品组的。

### 3.5.2打印信息

在同一项目组或产品组内，调测打印出的信息串的格式要有统一的形式。信息串中至少要有所在模块名（或源文件名）及行号。

说明：统一的调测信息格式便于集成测试。

### 3.5.3单元测试

编程的同时要为单元测试选择恰当的测试点，并仔细构造测试代码、测试用例，同时给出明确的注释说明。测试代码部分应作为（模块中的）一个子模块，以方便测试代码在模块中的安装与拆卸（通过调测开关）。

说明：为单元测试而准备。

### 3.5.4集成测试

在进行集成测试/系统联调之前，要构造好测试环境、测试项目及测试用例，同时仔细分析并优化测试用例，以提高测试效率。

说明：好的测试用例应尽可能模拟出程序所遇到的边界值、各种复杂环境及一些极端情况等。

### 3.5.5版本维护

1. 用调测开关来切换软件的DEBUG版和正式版，而不要同时存在正式版本和DEBUG版本的不同源文件，以减少维护的难度。

2. 软件的DEBUG版本和发行版本应该统一维护，不允许分家，并且要时刻注意保证两个版本在实现功能上的一致性。

### 3.5.6其他

1. 在编写代码之前，应预先设计好程序调试与测试的方法和手段，并设计好各种调测开关及相应测试代码如打印函数等。

说明：程序的调试与测试是软件生存周期中很重要的一个阶段，如何对软件进行较全面、高率的测试并尽可能地找出软件中的错误就成为很关键的问题。因此在编写源代码之前，除了要有一套比较完善的测试计划外，还应设计出一系列代码测试手段，为单元测试、集成测试及系统联调提供方便。

2. 调测开关应分为不同级别和类型。

说明：调测开关的设置及分类应从以下几方面考虑：针对模块或系统某部分代码的调测；针对模块或系统某功能的调测；出于某种其它目的，如对性能、容量等的测试。这样做便于软件功能的调测，并且便于模块的单元测试、系统联调等。

3. 编写防错程序，然后在处理错误之后可用断言宣布发生错误。

示例：假如某模块收到通信链路上的消息，则应对消息的合法性进行检查，若消息类别不是通信协议中规定的，则应进行出错处理，之后可用断言报告。

# 4注解

Flutter： Google 开源的 UI 工具包，帮助开发者通过一套代码库高效构建多平台精美应用，支持移动、Web、桌面和嵌入式平台。

Spring Boot：由Pivotal团队在2013年开始研发、2014年4月发布第一个版本的全新开源的轻量级框架。它基于Spring4.0设计，不仅继承了Spring框架原有的优秀特性，而且还通过简化配置来进一步简化了Spring应用的整个搭建和开发过程。

# 附录

[1] Flutter [EB/OL].https://flutterchina.club/ -2020/11/04

[2] Spring Boot [EB/OL]. https://spring.io/projects/spring-boot -2020/11/04