Contents

🎯 算法组培训中期考核规则

version 0.1

一、考核目的

本次中期考核旨在检验同学们在前半阶段培训中对于**编程能力、工程组织能力、文档编写、代码规范性以及 Git 使用习惯**的掌握情况。

考核采用**项目制**形式，要求同学们自主设计与实现一个工程化项目，并在 GitHub 上开源提交。本次考核占总成绩的 **10~20%**，具体的权重将在最终评分时确定。

本次考核不限定项目方向（无需局限于计算机视觉），仅要求体现工程化能力与编码质量。

二、提交内容与方式

1. 提交内容

每位同学必须提交：

- **一个 C++ 工程化项目（主项目）**
- GitHub 项目链接（公开仓库）
- 项目构建与运行方式需在 README 中明确说明

可额外提交其他项目（不限语言，不限形式），但仅作为附加展示，不计入主项目评分。

> **额外项目**可以是 Python 脚本、Java 程序、Web 应用等任何形式的项目，例如自己之前实现的小组项目或课程实验，也可以是学习语言或其他技术的笔记仓库。鼓励同学们根据个人兴趣与特长进行创作。尽管**额外项目**不计入评分，但建议同学们积极尝试不同方向与技术栈，丰富个人项目经验。对额外项目的良好实践也会在后续考核中作为加分项考虑。

2. 提交方式

- 将 GitHub 仓库链接通过指定 QQ 群内收集表提交
- 截止时间：**2025 年 12 月 14 日**

三、主项目硬性要求（C++ 工程项目）

本项目要求所有同学**必须使用 C++ 进行工程化项目开发，并体现工程思维、模块化设计与可维护性**。以下为必须满足的要求。

---

1. C++ 语言要求（必须体现 OOP）

你的项目必须满足：

✔ **至少包含 两个以上的类（Class）** 要求：

- 类不能只是空壳，需要有方法（public/private 成员函数）
- 至少一个类需要具有内部状态（成员变量）

示例类结构（示意）：

```
class Tokenizer {
public:
    Tokenizer();
```

```
    std::vector<std::string> tokenize(const std::string& input);
private:
    char delimiter;
};
```

✔ **至少具有一种类之间的关系** 包括但不限于：

- **继承（Inheritance）**
- **组合（Composition）**
- **聚合（Aggregation）**

示例（组合）：

```
class Application {
private:
    Logger logger;    // Application 组合了 Logger
    Config config;
};
```

✔ **不允许只写一个 main.cpp 把所有逻辑堆进去** 你的项目必须按模块拆分，例如：

```
src/
├── main.cpp
├── tokenizer.cpp
├── parser.cpp
include/
├── tokenizer.h
├── parser.h
```

一个合理的 C++ 项目应该做到：

- main.cpp 只负责组织模块流程
- 主要逻辑封装在类中
- 头文件放声明，cpp 放实现

---

2. 必须使用 CMake 组织项目（工程化构建）

项目根目录必须包含：

```
CMakeLists.txt
src/
include/
```

最基本要求的 CMake 结构如下：

```
project_name/
|—— CMakeLists.txt
|—— src/
|      ├—— main.cpp
|      └—— module1.cpp
|—— include/
        └—— module1.h
```

✔ **CMake 必须实现以下能力**

2.1 可以编译出可执行程序 例如：

```
add_executable(main
    src/main.cpp
    src/tokenizer.cpp
    src/parser.cpp
)
target_include_directories(main PUBLIC include)
```

2.2 杜绝“直接把所有文件写死在 main 里”

```

add_library(core
    src/tokenizer.cpp
    src/parser.cpp
)
target_include_directories(core PUBLIC include)

add_executable(app src/main.cpp)
target_link_libraries(app core)
```

2.3 若有组件/库，可拆分为 target（推荐）

---

3. 项目结构（必须清晰、有模块化）

主项目必须包含清晰的目录结构，至少包括：

```
project/
├── CMakeLists.txt
├── README.md
├── include/
│   └── *.h
├── src/
│   └── *.cpp
```

推荐可选结构：

```
├── tests/          # 单元测试
├── docs/           # 文档（如设计说明）
├── examples/       # 示例代码
```

目录结构应体现模块化，例如：

```
src/
├── core/
│   ├── tokenizer.cpp
│   ├── parser.cpp
├── utils/
│   └── logger.cpp
```

禁止：

- 文件全部堆在 `src/` 下无分类
- 单个文件超过几百行的大杂烩代码

---

4. 文档要求（README + 注释）

✔ **README.md 必须包含以下内容**

1. **项目简介**（这个项目是什么？解决什么问题?）
2. **主要功能说明**
3. **项目目录结构解释**
4. **构建方式与运行方式** 示例：

```
mkdir build
cd build
cmake ..
make
./app
```

5. (可选) 模块设计说明

✔ **注释要求**

- 主要类、公共接口必须写注释
- 内容解释逻辑或功能，而不是重复代码
- 不要求注释每一行，但逻辑块或函数必须有说明

示例注释:

```
// Parse tokens into an AST-like structure
Node Parser::parse(const std::vector<std::string>& tokens) {
    ...
}
```

---

5. Git 提交记录要求（必须具有开发过程）

要求:

✔ **不允许一次性提交所有代码** 需要至少 **5 条以上有效提交**，例如:

- 初始化项目结构
- 添加某个模块
- 修复 Bug
- 重构
- 文档更新

✔ **提交信息必须清晰** 推荐格式:

```
feat: add tokenizer class # 添加新的实现
fix: resolve crash on empty input # 修复 Bug
refactor: extract config loader into its own module # 重构代码
docs: update README with build steps # 更新文档
```

✔ **提交内容必须与 message 对应** 不能出现：

```
feat: add new module
(实际上改了 20 个文件，含各类改动)
```

提交应尽可能做到**单一职责。**

6． 项目必须能成功构建 + 能运行

无论项目内容是什么：

- make 后必须成功编译
- 运行后不得出现未处理的崩溃
- 必须有可观测结果（程序打印/文件输出/简单交互均可）

示例输出方式：

- 命令行输出
- 日志文件
- 生成结果文件

7． 内容不限（非视觉方向也完全允许）

项目主题完全自由：

- 工具类项目
- 模拟器
- 文本处理
- 路径规划
- 小型数据库
- 游戏逻辑
- 简单解释器
- 网络小工具
- 个人兴趣项目

只要满足上述工程化要求即可。可以参考的开源项目：

- 全网搜集 C/C++ 入门练手项目实战

四、C++ 工程项目评分细则表

1. 代码能力（C++ + OOP）——40 分

评分项	分值	评分标准
类设计数量与复杂度	10	0 分：无类设计；5 分：只有简单类；10 分：有多个类，结构清晰，职责明确
面向对象特性体现	10	0：无 OOP；5：部分封装；10：有类关系（继承/组合/聚合），结构合理
模块间接口设计	10	0：模块混乱；5：基本可用；10：接口清晰，解耦良好
代码风格与规范性	10	0：严重混乱；5：基本规范；10：变量命名统一、逻辑清晰、函数职责单一

小计：**40 分**

---

2. 工程组织结构（CMake + 模块化）——20 分

评分项	分值	评分标准
CMake 使用规范	10	0：构建失败；5：能构建但较混乱；10：target 分离、路径清晰
目录结构与模块化水平	10	0：单文件；5：简单拆分；10：合理分目录，src/include 清晰，模块分层

小计：**20 分**

---

3. 文档与注释——20 分

评分项	分值	评分标准
README 完整度	10	0：没有；5：基本说明；10：结构说明 + 构建方法 + 使用示例齐全
代码注释质量	10	0：无注释；5：部分关键点；10：重要类和函数都有解释，可读性好

小计：**20 分**

4. Git 提交与版本管理——20 分

评分项	分值	评分标准
提交数量与粒度	10	0：一次性提交；5：有 3~5 个提交；10：>5 次有效提交，粒度合理
提交信息规范性	10	0：混乱；5：部分规范；10：提交内容清晰且内容对应实际修改

小计：**20 分**

总分（主项目）：**100 分**

ex. 加分项（最多 +10 分）

加分项	分值	说明
测试（Unit Test）	+4	使用 GoogleTest / Catch2 / 自写测试均可
额外的文档（如设计文档、类图）	+4	docs/ 目录、流程图、架构图
项目展示视频	+2	简单演示即可，非必须【在收集表中提交，不用上传至 GitHub 仓库】

最高加分：**10 分**

总分：核心内容 100 分 + 加分项 10 分 = **110 分**

五、其他说明

1. 如果 github 仓库因某些原因无法使用，也可以使用 GitLab、Gitee 等其他公开代码托管平台提交，但请确保链接**可访问**。如果无法访问，将视为未提交。
2. 本考核主要检验工程能力，不以项目大小为主要评价依据。
3. 多次提交不同项目仅其中一个 C++ 工程作为主项目评分（在提交的收集表中需要选为主项目），其他项目仅作展示。尽管其他项目不会计入评分，但良好实践会作为后续入队考核的加分项考虑。
4. 可参考开源项目，但必须**自行实现主要功能**的代码。

5. 鼓励大家通过**第三方库**提升项目质量，但请务必在 README 中注明所使用的第三方库及其用途。
6. 不反对大家借助 ChatGPT 等工具辅助编程，但请务必理解代码逻辑，确保代码质量。
7. 在最终的入队考核中，将会要求大家对该项目进行讲解与答辩，请务必熟悉自己提交的代码。如果无法回答相关问题，可能怀疑代码非本人编写，成绩将**按 0 分计算**。
8. 如有任何疑问，请及时在 QQ 群中提问或联系培训负责人。
9. 后续如有规则更新或修改，将会在 QQ 群中通知，请大家保持关注。

祝大家考核顺利，期待看到各位的精彩项目！🚀🎉