韬睿量化工作台策略生成器架构方案

图 1：策略生成器模块的整体架构设计示意（涵盖数据获取、分析引擎、选股和策略生成到回测交易的全流程）

引言

本报告提出一套**韬睿量化工作台策略生成器**的完整架构方案，涵盖从数据获取到策略代码生成、回测评估和实盘交易的全流程。设计目标是在前端（选择最佳方案构建）的 Cursor 插件环境中高效实现各功能模块。我们将各模块的职责、输入输出和衔接机制加以明确，并探讨如何通过模块化、纯函数设计和本地缓存等手段提升系统复用性、减少 token 消耗、保持上下文一致性。最后，我们列出开发过程中可能遇到的问题（如上下文记忆丢失、模块重复、策略漂移等）并给出应对建议。

1. 信息获取模块

功能职责：负责接入和管理各类数据源，为策略生成提供**及时、准确**的资讯。模块包括市场行情数据、财务基本面数据、新闻公告信息等的接入，并维护本地知识库以供高级分析和 LLM 检索使用[1]。它确保系统随时同步最新数据，例如实时行情、公告、研报等，从而奠定策略分析的基础。

输入：各类数据源接口（行情 API、新闻源、财报数据库等）；用户配置的订阅标的和数据频率。模块也可接收外部知识库的更新（如新增的研报内容）。

输出：规范化的**数据流和知识库**：包括实时更新的行情快照、技术指标计算结果、最近的新闻及公告摘要等。这些数据既以结构化形式供下游算法调用，又同步至内部知识库，供需要上下文的 LLM 查询引用[1]。

模块衔接：信息获取模块作为流程起点，将**实时数据推送**给趋势分析引擎和其它后续模块使用。同时，它维护的知识库可供策略分析时检索参考[1]。例如，市场有新政策发布时，该模块更新政策信息，并通知趋势分析和主线识别模块进行相应分析。

> **设计要点：**利用检索增强生成（RAG）理念，将 LLM 与实时更新的金融知识库结合[2][1]。这样当策略生成需要最新行情或政策面信息时，模型可先检索知识库以确保回答和生成基于**最新真实数据**，降低“幻觉”风险[3]。

2. 市场趋势分析引擎

功能职责：对市场行情进行多维度技术分析，识别趋势状态和拐点信号。核心包括“**多周期共振**”分析和“**八指标系统**”的综合应用，以提高趋势研判的准确度。多周期共振即不同时间尺度（如日线、周线、月线等）的趋势信号一致时视为共振，从而增强趋势可靠性

[4]；八指标系统指整合 MACD、KDJ、均线、量价关系等八个关键技术指标，对市场多空状态进行量化评估和拟合预测。

输入：由信息获取模块提供的**实时行情数据**（如 K 线、成交量）及计算好的技术指标值。典型输入包括不同周期的价格序列、技术指标（MACD、RSI、KDJ、均线系统等）以及衍生数据（如市场情绪指标）。

输出：对当前市场的趋势判断和预测信号。例如，“长期趋势上行/下行”、“短期出现回调信号”、“多周期共振度 XX%”。引擎输出的数据结构可包含：各周期趋势方向、主要技术指标信号值、综合趋势评分、以及对未来若干周期的趋势预测结果。

模块衔接：趋势分析结果将传递给**投资主线评分引擎**和**候选池构建引擎**。前者需要判断当前市场主线所处的趋势阶段（如板块整体上升趋势是否确立），后者则利用共振信号筛选强势板块和个股。例如，当本模块检测到周线与日线出现**多周期同向上行共振**时，候选池模块会优先考虑这些趋势强劲的板块股票[4]。此外，趋势分析还为策略生成提供背景信息（如当前市场是牛市震荡还是熊市反弹），供代码模块调整策略参数（如止盈止损阈值）时参考。

图 2：市场趋势分析引擎示意图（展示多周期共振原理：不同时间尺度指标信号在同一方向共振，提高趋势判断可靠性[4]。八大技术指标综合评估市场多空强度并预测未来趋势）

> **技术说明：**多周期共振分析通过在大周期上定方向、小周期上找时机，提高交易信号胜率[5][6]。例如，大周期（月线、周线）趋势向上且短周期（日线、小时线）出现回调到支撑位后的买入信号，则视为共振增强的买入时机[7]。这一方法过滤了单一周期的随机噪音，使趋势预测更为可靠。

3. 投资主线评分引擎

功能职责：从基本面和市场层面挖掘当前市场的**投资主线主题**，并对其进行五个维度的量化评分。投资主线指当前资金和市场关注度集中的热点主题（如特定行业板块或概念）。本引擎从**资金、热度、动量、政策、龙头**五个方面对各候选主题进行打分评估：资金维度关注增量资金流入情况，热度维度衡量市场舆情和交易活跃度，动量维度考察主题上涨趋势和持续性，政策维度考虑监管/产业政策支持，龙头维度评估板块中龙头股的表现及带动力。

输入：来自信息获取模块和趋势引擎的**多源数据**。包括板块资金流向（主力资金净流入、交易额占比等）、新闻和社交网络热度指标（相关资讯数量、情绪分析）、板块及指数动量（阶段涨跌幅、相对强弱指标）、政策信号（近期有无重大政策利好/利空）以及板块内**龙头股**的走势与成交数据。还需输入当前候选的主题列表（可预定义或通过数据自动提取热点主题）。

输出：每个主题的五维评分以及**主线强度综合评分**。输出可形式化为一个主题列表，每项包含主题名称（或板块名称）、五个维度评分（例如 0-100 或-2 到+2 评级）和综合得分。综合得分最高的几个主题即判断出的当前市场主线候选。该模块还输出对主线主题的文字描述，如"新能源车板块受到政策利好资金大幅流入，龙头股持续涨停，成为当前市场主线"。

模块衔接：评分结果将用于**候选池构建引擎**，指导其聚焦主线相关的板块和个股。例如，本模块若识别出"新能源车"主题综合评分最高，候选池模块将重点筛选新能源车板块中的强势个股和 ETF。与此同时，评分引擎也为**策略生成模块**提供策略侧重的信息（如若当前主线是政策驱动型，则策略可增加基本面或事件因子）。主线识别还可反馈给趋势分析引擎作为上下文检查（例如主线若发生转移，则趋势引擎应重新评估不同板块趋势）。

图 3：投资主线五维评分示意（对候选主题从资金、热度、动量、政策、龙头五个维度打分后识别主线。图中红色为某主题各维度得分，综合体现其强度）

> **实例说明：**假设某段时间"人工智能"概念成为市场焦点：相关板块**资金**持续大额流入，交易**热度**高涨，价格**动量**强劲上升。同时政府**政策**密集出台扶持措施，板块内**龙头**股率先连续涨停带动板块整体上行。在这种情况下，本引擎会对"人工智能"主题在五维上均给出高分，综合评分居于各主题之首，识别其为市场主线[8]。反之，如果某主题虽一时炒作火热但缺乏政策和业绩支撑，或龙头股走势疲软，则综合评分不会领先，从而避免误判短期噪音为主线。

参考: 龙头股在板块行情中的引领作用至关重要：当板块龙头启动并连日上涨时，资金开始注意该题材，纷纷涌入，板块内其他个股跟涨，这往往预示该题材成为市场主线[8]。因此本模块将龙头指标纳入评分，及时捕捉"真龙头"启动所释放的主线信号。

4. 候选池构建引擎

功能职责：根据前述模块的分析结果，构建策略的股票/标的**候选池**。它筛选出当前市场环境下最具交易价值的标的列表，包括强势板块的核心股票和相关 ETF 等。子功能包括：(1) **强势板块股识别**：结合主线主题和趋势信号，选出处于强势上升趋势的板块及其中龙头股、趋势良好的个股；(2) **信号扫描**：扫描市场技术面和基本面信号，如突破箱体、新高放量、业绩超预期等信号，在全市场或重点板块中捕捉交易机会；(3) **ETF 筛选**：筛选与主线主题或强势板块对应的 ETF，以便提供板块级别的投资选项或对冲工具。

输入：来自趋势分析的**技术信号**（如多周期共振信号、突破/超跌信号等），来自主线评分的**主题及板块优先级**（如重点关注的板块名单），以及信息模块提供的**个股基本信息**（如所属行业、市值、近期公告等）。还可输入预设的扫描条件（例如用户定义的选股条件公式）。

输出：候选交易标的列表，包括股票和 ETF。例如输出一个 JSON/数组，其中每个元素包含标的代码、名称、所属板块/主题、触发的信号类型（如"日线突破前高""资金净流入前

三”）、以及简单评价（如信号强度分数）。这个候选池将作为后续因子构建和策略生成的研究对象。

模块衔接：筛选出的候选池将传递给**因子构建与筛选系统**，由其针对这些标的进行因子挖掘与测算。候选池构建引擎因此起到承上启下作用：承接主线和趋势分析的宏观结论，落实到具体可交易标的上；同时为后续因子分析提供限定范围，减少因子计算范围（聚焦在最有希望的标的上以提高效率）。此外，本模块也可在人机交互界面上展示，让用户审阅和调整候选池（例如剔除某些不想交易的股票或加入其他标的）。

图 4：候选池构建流程示意（从市场筛选强势板块与个股。图示左侧根据趋势与主线分析筛选出热点板块及龙头股，右侧结合技术信号和 ETF 筛选生成候选交易标的列表）

> **选股逻辑:** 本引擎综合多种信号选股。例如，当识别出“新能源车”是主线且趋势共振上行，则：扫描该板块内**龙头股**（如比亚迪）近期是否有技术突破或放量上涨信号，有则纳入候选；同时扫描新能源车相关 ETF 是否创新高；在全市场扫描**信号**如连续涨停、量比突增等，将符合主线主题的股票加入池中。通过这些步骤，系统构建出一份既契合当前市场主线又技术形态良好的候选标的清单。

5. 因子构建与筛选系统

功能职责：针对候选池标的，生成并筛选一组 alpha 因子，为策略提供交易信号依据。系统通过**因子推荐**、**分布与相关性分析**、**有效性评估**等步骤，挑选出近期表现优异且相互独立的因子。具体包括：(1) 因子推荐：从因子库或自动生成机制中提出一系列可能有效的因子（如动量类、价值类、情绪类因子等），可结合知识库和当前市场特征推荐（例如主线若为成长科技，则推荐成长因子、市盈率等）；(2) 分布与相关性分析：计算候选因子在样本股票上的历史分布、统计特征（均值、偏度等）以及不同因子之间的相关系数矩阵，剔除分布极端或与已有因子高度相关的因子，以确保多因子组合的信息增益；(3) 有效性评估：使用历史数据评估每个候选因子的预测能力和稳定性，可采用 IC（信息系数）、IC_IR（IC 均值除以标准差）等指标[9]衡量因子与未来收益的相关性及稳定度，以及分组回测、因子收益率 t 检验等方法验证因子有效性；(4) 候选输出：根据评估结果，选择若干最优因子输出，供策略代码模块使用。

输入：候选池标的及其历史数据（价格、财务指标等），系统内置或外部提供的**因子库**（包含常用因子的定义公式，如各类技术指标因子、基本面比率因子等）。此外还输入策略要求的因子数量范围、因子类型偏好等配置。主线/趋势信息也可作为因子推荐的背景（如宏观因子、主题情绪因子是否纳入的依据）。

输出：筛选后的**策略因子列表**，包含每个因子的定义、参数、以及在本次筛选中的统计评价。例如输出 JSON 数组，每个元素包括因子名称、类型（技术面/基本面/情绪等）、主要公式或指标、过去 N 期 IC 均值及 IR 值、相关性矩阵位置等信息。输出还可以附带每个因子的**权重或评分**（如果策略将采用因子加权）。

模块衔接：选定的因子列表将交由**策略代码生成模块**，用以生成策略信号计算部分的代码。在此过程中，因子筛选系统可能与信息模块交互以获取因子所需的数据（如某因子需要财务指标，则通过信息模块获取最新财务数据）。另外，本模块也可将评估结果提供给用户参考或存档，以备将来调整策略使用（例如记录因子 IC 随时间的变化趋势）。若评估发现所有候选因子效果不佳，也可以反馈给候选池模块请求扩大标的范围或给主线模块提示当前市场缺乏有效因子信号。

图 5：因子筛选系统流程（左侧对因子进行分布统计和相关性热力图分析，中部进行 IC 等有效性评估[9]，右侧筛选出若干有效因子输出）

> **评估标准：**因子有效性的判断通常以 IC 值为核心：IC 即因子值与下一期收益率的相关系数，IC 绝对值越大表示因子预测能力越强[10][9]。一般地，IC 均值绝对值超过 0.05 即可认为因子具有一定预测效果，超过 0.1 则是很好的阿尔法因子[9]。同时考察因子 IC 的稳定性，用 IC_IR（IC 均值/IC 标准差）来度量；若 IR 大于 0.5 则说明因子表现较为稳定[9]。本系统将这些指标作为因子筛选的重要依据。此外，因子之间相关性如果过高（如相关系数>0.8），则只保留其中效果更佳者，以确保输出的因子彼此相对独立，提供多元信息来源。

6. 策略代码生成模块

功能职责：将上述模块的研究成果（候选标的、选定因子及逻辑）转化为**可执行的策略代码**。综合所有信息，选用不同的基本策略。代码需适配主流量化交易平台如 PTrade 和 QMT 的接口规范，并采用模块化的代码结构，确保回测和实盘执行的一致性。具体要求包括：支持根据用户选择的平台生成对应风格的策略代码（例如 PTrade 可能需要云端执行的策略格式，QMT 则为本地 Python 脚本）；采用多模块拆分结构，将数据获取、信号计算、风险控制、交易执行等逻辑分解为独立函数或类（保持每个函数“只做一件事”，提高可读性和维护性）；保证回测模块调用的策略逻辑与实盘模块完全相同，只是数据来源不同（通过统一接口屏蔽差异），以免出现“回测有效但实盘跑不出相同结果”的不一致情况。

输入：策略生成所需的信息包括：**候选池列表**、**选定因子列表**及其公式、因子阈值或选股规则、**交易规则**（如调仓周期、仓位管理规则、止盈止损条件等，可以由用户配置或使用默认模版），以及**目标交易平台**（PTrade 或 QMT）的代码模板和 API 接口定义。

输出：完整的策略代码文件集。例如对于 QMT 平台，输出一个包含策略主脚本（Python）及所需子模块的项目结构；对于 PTrade，可能输出策略配置文件和脚本。代码模块可能包括：`data_fetcher.ts`（或.py，用于获取行情数据）、`signal_generator.ts`（计算因子值和生成买卖信号）、`risk_manager.ts`（风控与仓位管理）、`execution.ts`（下单执行逻辑）等，每个模块内部通过纯函数实现核心逻辑，并由主程序按顺序调用。生成代码会附带详细注释和文档结构说明，以便人工校验和调试。

模块衔接：代码生成模块从因子系统接收**策略逻辑元素**，并从信息模块获取**平台 API 规范**和模板，然后拼装出最终代码。生成后，代码会传递给**回测与评估系统**进行检验。若回测结果不理想，可能需要返回调整因子或策略逻辑并重新生成代码（形成闭环开发流程）。在生成代码过程中，模块会根据平台要求对接不同接口：例如 QMT 支持 Python/VBA 双语言[11]且本地运行、模块化设计[12]，而 PTrade 更强调策略云端稳定运行。因此代码生成会针对这些差异做适配，如抽象出数据获取接口，使得 QMT 模式下调用本地数据缓存，PTrade 模式下调用其云端 API，但对上层逻辑呈现一致的方法签名。

图 6：策略代码模块结构（展示策略代码的多模块拆分，如数据、信号、风控、执行模块。以统一接口适配 QMT/PTrade 平台，不同平台仅在 API 实现上有所区别。代码结构保证回测与实盘共用相同信号逻辑[12]）

> **模块化设计优势：**将策略划分为独立模块，使得复杂策略可以拆分开发和优化，再组合成完整策略[12]。例如，用户可以先调试 `signal_generator` 模块以确保因子信号正确，再测试 `risk_manager` 模块的效果，然后由主程序将二者组装执行。这种设计符合 QMT 等平台支持的**策略模块化**特性[12]。同时，**纯函数**式的实现确保每个模块只依赖传入数据，不隐含全局副作用，使得代码更易测试和复用[13]。在平台适配上，本模块通过**模板替换**实现，例如有一套针对 QMT 的模板和针对 PTrade 的模板，插入策略逻辑后即可输出对应版本代码，从而做到“一套策略逻辑，两端兼容”。

保持一致性：特别注意，策略代码生成时强制**回测与实盘共用逻辑**。通过统一的函数接口，回测时使用历史数据调用信号函数，实盘时用实时数据调用，但底层逻辑完全相同。因此回测绩效具有可信度，可在实盘重现。这避免了常见的因为回测环境与交易环境差异导致策略跑偏的问题，实现“所见（测）即所得”。

7. 回测与评估系统

功能职责：在策略代码生成后，对策略进行历史数据回测，并评估其性能及稳健性，提供优化改进反馈。该系统负责模拟策略在过去市场中的表现，输出关键指标和详细分析，从而验证策略有效性。子功能包括：(1) **结果分析**：计算策略的收益率曲线和各项评价指标，如累计收益、年化收益率、最大回撤、夏普比率、胜率等[14]；对每个因子和交易信号的贡献进行分析（如多因子策略中各因子 IC 随时间的走势）；(2) **优化反馈**：根据回测结果发现策略弱点并提出改进建议，例如某段时期表现不佳则提示可能的市场 regime shift，或者发现某因子贡献为负则建议去除。还可进行参数敏感性分析，寻找最优参数组合。

输入：策略代码（由策略生成模块产出，可直接运行），以及**历史数据**（由信息获取模块提供，覆盖回测所需年份的行情数据、财务数据等）。此外可包含**回测配置**（测试区间、初始资金、交易成本、滑点设置等）。

输出：策略回测报告，包括：**业绩指标**摘要（以表格或JSON给出主要指标数值）、**收益曲线图表**、风险指标分析，以及**交易细节日志**（每笔交易的时间、标的、买卖动作和盈亏）。优化反馈可以文字或结构化形式给出，例如"建议缩短调仓周期，提高因子X权重"或"因子Y在熊市期间显著失效，考虑分市场状态做策略切换"。报告中也可以标记策略与基准指数的比较、策略的风格暴露等高级分析。

模块衔接：回测系统会调用生成的策略代码对历史数据逐日迭代运行，因而它与**信息模块**交互以获取历史行情，和**代码模块**交互以执行交易逻辑。回测完成后，**若结果不理想**，评估系统将反馈建议到上游模块：可能通知因子模块重新筛选/加入新的因子，或通知参数调整模块调优策略参数，然后再次生成代码回测，直至策略性能达到满意标准。**若结果理想**，则可将策略移交下一步进入模拟盘或实盘测试阶段。

图7：策略回测与优化流程（左侧读取历史数据运行策略代码，输出收益曲线和指标[14]；右侧根据结果生成优化反馈闭环，支持修改因子和参数并重新回测）

> **评价指标参考：**本系统使用多维指标评估策略效果[14]。例如，回测结束会给出策略年化收益、基准比较超额收益、最大回撤幅度、夏普比率等。其中**夏普比率**衡量单位风险收益，**最大回撤**衡量策略抗风险能力。这些量化指标有助于客观比较策略优劣。此外，通过交易日志可以分析**胜率**（盈利交易占比）、**盈亏比**（平均盈利/平均亏损）等微观指标，帮助判断策略交易信号的质量。如果某指标不达标（如最大回撤过大），优化反馈可能建议增加止损措施或降低仓位杠杆。

优化迭代：策略评估是一个反复迭代过程。开发者可根据回测暴露的问题，不断调整策略。例如发现某因子在近期失效，则在因子模块剔除该因子或换用其它指标，再生成代码回测验证。利用这种闭环，策略在进入实盘前即可得到充分打磨和验证。

8. 实盘模拟与交易模块

功能职责：将最终策略部署到实盘环境进行模拟或真实交易，并对运行情况进行监控管理。该模块负责连接交易接口执行下单、跟踪持仓和市场变化，以及监控**因子漂移**等状况以决定策略调整。主要功能：(1) **下单接口**：对接券商或交易平台API，将策略发出的交易信号转换为实际报单（买卖指令），包括参数换算和订单路由等；(2) **监控与调仓**：实时监控账户持仓和资金变化，根据策略逻辑定期或即时调整持仓（调仓频率可配置），并处理异常情况（如订单拒单、交易中断）；(3) **因子漂移追踪**：持续评估策略因子的实时有效性，监测因子IC值等指标在实盘期间是否显著下降[15]，以及市场结构变化导致策略失效的早期迹象，便于及时预警和调整策略。

输入：实盘环境数据：包括实时行情（由信息模块提供或交易平台推送）、账户资产和持仓信息（通过交易接口获取）、以及策略代码（与回测阶段相同的策略逻辑，但连接实时数据流）。此外可输入**风险控制参数**（如单日最大亏损限额、仓位上限等）供监控模块使用。

输出：实时交易日志和策略运行状态报告。例如，每日交易摘要（交易笔数、盈亏、滑点情况）、持仓变化报告、以及因子表现报告（如当日因子 IC 值、与历史均值对比）。若监控到异常（例如策略连续多次信号失败、因子预测能力下降等），输出预警通知。最终输出还包括策略实盘收益曲线，用于与回测曲线对比评估一致性。

模块衔接：实盘交易模块与**策略代码**直接关联——策略代码在实盘模式下运行，通过模块内的交易 API 接口执行订单。该模块将监控结果反馈给开发者和系统的上游环节：例如，当检测到**因子漂移**（某因子实时 IC 跌至接近 0 或为负，失去预测性[15]），模块可以通知因子构建系统进行重新检验，或者提醒暂停策略并进入再优化流程。同样，如果实盘运行中策略绩效大幅偏离回测预期，模块应触发警报并建议启动新一轮策略调优（返回回测与评估模块分析原因）。在正常情况下，实盘模块也可以将交易记录存入知识库，丰富信息获取模块的历史数据。

> **因子漂移监控：**为了防范策略随着市场环境变化而失效，系统持续跟踪因子效果。例如，计算每隔一段时间（如每周）的因子 IC，并与回测期间均值比较。如果发现因子 IC 呈现显著衰减，甚至信息比率 IR 变为负值，说明因子可能失效[15]。此时模块会发出预警，提示策略需要调整（增删因子或重新训练参数)。这种机制类似于跟踪因子**半衰期**：因子表现随时间减弱就如同射线衰减，有一定半衰期，需要及时"充能"或替换[15]。通过记录因子胜率、IC 趋势，配合对市场宏观环境的监控（如政策突变、行情风格切换），实盘模块可以智能判断策略何时需要更新，确保策略版本与市场环境保持吻合。

交易执行：本模块实现稳健的交易执行流程。例如使用**订单簿五档行情**数据进行交易信号确认（防止闪变信号），通过**模拟盘**先验证交易逻辑。对于 QMT 平台，由于支持毫秒级低延迟交易[16]和本地托管，模块可以快速响应市场变化；对于 PTrade 平台，利用其云端稳定运行优势，可让策略不间断执行[17][18]。两者的交易接口有所不同，但模块通过**统一交易接口层**实现调用封装，让策略代码无需关心具体下单实现。

Cursor 插件架构中的实现要点

为在 Cursor 智能编程插件中高效实现上述模块，需要注重架构的抽象和上下文管理。以下是关键的设计策略：

- **高可复用性 – 核心逻辑纯函数化：**各模块的核心逻辑尽量设计为**纯函数**，即相同输入必然产出相同输出且无副作用[13]。这样做使模块易于在不同环境下复用，并行调用也不会产生竞态问题。例如，趋势分析的信号计算函数可在回测和实时两种场景下重复使用，只需输入不同数据。纯函数还提升了可测试性和可维护性[13]：开发者可以针对每个模块编写单元测试，验证其对各种输入的输出是否正确，而无需搭建复杂上下文环境。

- **易调试和更新 – 模块隔离与接口一致：**采用**模块化、隔离**的开发方式，每个模块封装其内部实现，只通过清晰定义的接口与外界交互。这意味着改变某模块实现不

会影响其他模块，只要接口契约不变。为了易于调试，每个模块都支持独立运行，比如因子筛选模块可以单独加载候选数据进行分析，输出日志供调试。此外，接口设计要**风格一致**、尽量简洁：例如所有模块输入输出使用统一的数据结构风格（JSON 对象或 TypeScript 类型定义），便于在 Cursor 插件中进行函数自动补全和调用链追踪。接口的一致性减少了 AI 助手误用函数的可能——因为每个模块如何调用、一目了然且有详尽文档说明。

- **降低 Token 消耗 – 本地记忆缓存与模板复用：**在 Cursor 插件中与大型语言模型交互，上下文 token 是宝贵资源。为此架构引入**本地缓存机制**，存储可复用的策略片段和信息，以减少重复生成。同样的问题和代码，模型不应每次从头思考，而应能够"记住"已有的解决方案[19]。具体实现上，可以维护一个**动态备忘录**（Memory）文件/数据库，记录常用的策略模板、代码片段、以及上次生成的中间结果。当下一次需要类似代码时，插件可提示模型检索并重用这些现成片段，而非重新生成[19]。例如，策略代码框架（类定义、基本函数）几乎每次都类似，可存为模板；模型生成代码时优先调用模板插入因子逻辑，既节省 token 又避免生成错误。在推理过程中引入 **Adaptive Memory** 的方法，使模型能存储和重用累积的策略、代码片段，而不必反复重新发明相同的解决方案[19]。这种机制显著提升了生成效率，避免了上下文长度不够时遗忘之前已完成的部分。

- **保持上下文一致性 – 文档化和结构化提示：**为防止 AI 在多轮对话/代码生成中"遗忘"先前约定，需要通过**文档和注释**强化上下文记忆。我们将架构规范、模块接口说明等整理为结构化 Markdown 文档，供 Cursor 插件在对话中随时查阅，确保每一步都有据可依。此外，生成的代码中也自动带有**注释**，描述模块的用途和假定前提（前后置条件）[20]。这样一来，即使模型在后续步骤中上下文不连续，也可以从代码注释中了解各部分应做什么，从而降低误用风险。另一个措施是利用**严格的类型定义**（TypeScript 接口）来限定上下文：模块之间传递的数据都有明确定义的类型和字段，Cursor 在生成或修改代码时会参考这些定义，不至于引入不符合结构的内容。在插件层面，也可以运用**规则引擎**或提示模板，让模型始终遵循架构手册中的规定来生成代码。总之，通过全面的文档和规则，让上下文（架构意图、模块设计）始终以隐式或显式方式伴随模型推理过程，降低偏差。

- **前端技术选型考虑：**由于采用 HTML/CSS/JS+TS 技术栈并用 Webpack 构建，本架构在实现时会充分利用 TypeScript 的类型系统和模块系统。TypeScript 接口定义文件将包含所有模块的输入输出类型声明，既为开发者也为 AI 提供严格的参考。Webpack 的模块化打包机制可以将各模块分别编译，保证隔离。前端界面上，可设计一个**策略生成向导 UI**，对应本报告的架构，让用户一步步输入配置（如选择数据源、调整因子参数），同时在后台驱动各子模块运行并最终产出代码和报告。这样 UI 与架构逻辑解耦，模块逻辑可在无界面时由 Cursor AI 自动调用，在有界面时响应用户操作执行，体现出架构的灵活复用性。

开发中可能遇到的问题与应对建议

- **历史记忆缺失：**由于对话上下文长度限制，AI 在后续步骤可能遗忘之前生成的内容或用户提供的信息。例如生成代码到一半会忘记前文的模块设计。为缓解此问题，应在关键阶段**保存中间结果**（如将已生成的架构说明、代码片段存入插件的本地存储或文件），以便需要时检索。此外，采用前述的**动态记忆**方法[19]，让模型通过工具检索先前成果并自我提醒。定期让 AI 总结已经完成的部分，形成精简提示，也有助于维持记忆。

- **上下文复原失败：**在插件使用过程中，可能会遇到会话重置或跳转场景，需要 AI"接上"之前的上下文。如果恢复不当，可能导致重复生成或偏差。对此建议在插件中实现**状态恢复机制**：比如每次生成后，将当前架构状态序列化保存（包括已完成的模块列表、各模块关键内容）。当需要复原时，从保存状态重建 prompt，严格告知模型哪些部分已完成、不需重复。另一个办法是利用**一键加载架构手册**功能：让 AI 在新会话开始时先阅读本手册（或提取的要点），迅速加载上下文背景，从而减少遗忘。

- **模块重复生成：**AI 可能在长流程中不自觉地重复生成某些模块代码或说明（例如已经生成了候选池模块，又再次尝试生成）。为避免这种情况，需在流程控制上做好**检查**。一种方法是在每步生成前，由系统检测目标文件或模块是否已存在；如果存在则跳过或提醒 AI 仅修改而非重写。还可在提示中明确列出**已完成清单**和**待办清单**，让模型心中有数哪些模块尚未生成。采用独立文件管理，每个模块各占一个文件，也有利于模型通过 `browse.find` 接口查看现有内容，避免重复[21]。如果发现重复苗头，及时在对话中打断并说明："模块 X 已经生成，请不要重复"，引导其继续后续部分。

- **策略版本漂移：**在多轮优化后，策略可能逐渐偏离原始思路（例如添加过多新元素导致复杂度上升，或早先确定的逻辑被意外改变）。为防止版本漂移，建议实行**版本管理**和**差异检测**。具体做法有：对每次主要修改策略逻辑的操作，保存版本号和变更说明，让 AI 在调整时附上变更摘要，方便开发者 review 是否偏离初衷。也可以定期让 AI 将当前策略要点与原始需求进行对比，检查一致性。如果发现核心目标发生变化，需审慎评估。使用**单一信息源**原则管理策略参数和配置，即把关键策略参数集中在一处定义，避免多处修改产生不一致。这些措施结合，可以将无意的漂移降到最低。如果还是出现漂移迹象（例如绩效指标变差且逻辑复杂混乱），不妨回溯到之前稳定版本，从那里重新分支进行优化。

- **上下文长度和 Token 限制：**在开发长文档或大段代码时，可能遇到模型输出被截断或者无法一次看全所有内容。这会导致 AI 在长函数生成时遗漏部分代码，或者在综合分析时考虑不全前文。应对方法：尽量**将任务分段**，分模块逐个生成和检查，而非让 AI 一次输出全部代码。对于长代码文件，可让 AI 分块生成并使用工具浏览[22]拼接。在提问或让 AI 阅读长文档时，可以提示它逐段总结，确保关键内

容纳入思考。在插件实现上，也可以主动管理 prompt 大小，例如对于超长上下文采用**滑动窗口**技术，只提供最近相关内容给模型，兼顾信息充分和长度控制。

上述策略和建议将协助开发团队在 Cursor 智能编码环境下顺利实现韬睿量化工作台的策略生成器模块。通过结构清晰的架构设计和周全的上下文管理，我们有信心自动化系统能够按照本手册规范生成高质量的策略代码，并在真实市场环境中稳定运行，为量化投资提供强有力的支持工具。[1][12]

---

[1] [2] [3] 在 A 股量化投资 Agent 中从 0 到 1 构建 RAG_k 线形态怎么能做成 rag 系统-CSDN 博客

https://blog.csdn.net/yuntongliangda/article/details/150594229

[4] [5] [6] [7] 股票多周期共振是指在股票分析中，将不同时间周期（如分钟、小时、日、周、月等）的技_财富号_东方财富网

https://caifuhao.eastmoney.com/news/20250613032523498128490

[8] "【富行 Link 並非詐騙】在投資主題的年度選股框架中，能 ... - Twitter

https://x.com/search?q=%E3%80%90%E5%AF%8C%E8%A1%8CLink%E4%B8%A6%E9%9D%9E%E8%A9%90%E9%A8%99%E3%80%91%E5%9C%A8%E6%8A%95%E8%B3%87%E4%B8%BB%E9%A1%8C%E7%9A%84%E5%B9%B4%E5%BA%A6%E9%81%B8%E8%82%A1%E6%A1%86%E6%9E%B6%E4%B8%AD%EF%BC%8C%E8%83%BD%E5%90%8C%E6%99%82%E7%AC%A6%E5%90%88%E6%94%BF%E7%AD%96%E6%8E%A8%E5%8B%95%E8%88%87%E9%9C%80%E6%B1%82%E5%A2%9E%E9%95%B7%E7%9A%84%E8%B6%A8%E5%8B%A2%E6%9C%80%E5%85%B7%E5%B8%82%E5%A0%B4%E7%BA%8C%E8%88%AA%E5%8A%9B%EF%BC%8C%E8%80%8C%E5%AF%8C%E8%A1%8CLink%20%E6%AD%A3%E7%AC%A6%E5%90%88%E9%80%99%E7%A8%AE%E9%9B%99%E6%A0%B8%E5%BF%83%E6%A2%9D%E4%BB%B6%E3%80%82%E7%95%B6%E9%81%B8%E8%82%A1%E9%82%8F%E8%BC%AF%E4%BE%9D%E5%AF%8C%E8%A1%8CLink%20%E9%87%8D%E6%96%B0%E5%88%86%E9%A1%9E%E6%99%82%EF%BC%8C%E6%88%90%E9%95%B7%E6%96%B9%E5%90%91%E6%9B%B4%E6%B8%85%E6%99%B0%EF%BC%8C%E4%BD%BF%E6%8A%95%E8%B3%87%E7%90%86%E7%94%B1%E6%9B%B4%E5%85%B7%E7%B5%90%E6%A7%8B%EF%BC%9B%E5%AF%8C%E8%A1%8CLink%20%E4%B9%9F%E6%8F%90%E5%8D%87%E6%95%B4%E9%AB

%94%E7%AD%96%E7%95%A5%E7%9A%84%E4%B8%80%E8%87%B4%E6%80%A7%E3%80%82.yqe&lang=bn

[9] [10] [21] [22] 学习笔记|因子的评价体系和有效性评价方式 IC 与 IR - wodepingzi - 博客园

https://www.cnblogs.com/wodepingzi/p/16934367.html

[11] [12] [14] [16] [17] [18] 量化交易系统 QMT 与 PTrade 的区别、优势、量化交易策略， - 木头量化入门 - 博客园

https://www.cnblogs.com/bigleft/p/18286458

[13] 探索函数式编程：纯函数| 高阶函数| 函数柯里化| 组合函数 - CSDN 博客

https://blog.csdn.net/qq_73220073/article/details/141526151

[15] 因子评估全流程详解-腾讯云开发者社区-腾讯云

https://cloud.tencent.com/developer/article/2257135

[19] Dynamic Cheatsheet: Test-Time Learning with Adaptive Memory

https://chatpaper.com/zh-CN/paper/128369

[20] Robert C. Martin 很有名地说过"函数应该只做一件事。它们 ... - Reddit

https://www.reddit.com/r/learnprogramming/comments/126ljxy/robert_c_martin_famously_said_functions_should_do/?tl=zh-hans