CHATBOT：流利的多语者——重新审视突破性的代码片段

David Noever[1] 和 Kevin Williams[2]PeopleTec，美国阿拉巴马州亨茨维尔市 Corporate Drive 4901-D，邮编 35805[1]
david.noever@peopletec.com [2]kevin.williams@peopletec.com

摘要
本研究运用人工智能驱动的代码助手，分析了一系列影响深远的计算机代码，这些代码塑造了现代技术，涵盖电子邮件、互联网浏览、机器人技术和恶意软件。本研究的原创贡献在于考察过去 50 年间最重要的代码进展中的一半，并在某些情况下显著提升其清晰度或性能。在所有考察案例中，人工智能驱动的代码助手都能为混淆代码或缺少说明性注释的软件提供见解。我们还基于错误修复和代码优化生成了其他示例问题；这些问题所需的推理深度远超传统 Google 搜索所能提供的水平。未来的工作将着眼于添加自动化文档和代码注释，并将选定的大型代码库转换为更新的版本，同时引入多个新的应用程序编程接口（应用程序编程接口）和链式多任务。人工智能驱动的代码助手为软件工程提供了宝贵工具，尤其能够提供人类专家级的专业知识，并协助重构遗留代码，或简化对高价值代码仓库的说明或其功能。

关键词
*Tran*Transformer模型、文本生成、恶意软件生成、生成式预训练Transformer、GP *T*

1. 引言NTRODUCTION

最新一代人工智能（AI）和聊天应用 [1-13] 作为软件生成器展现出特别的潜力 [4,11], ，它们提供了一种学习复杂编码原理的新型交互方式 [6],，能够用多种语言评论现有代码[8], ，并且通常可充当编程助手[8-12]。OpenAI 最近的努力使大语言模型（LLM）向公众开放[1-2]。作为一个实验平台，尤其适用于理解软件原理，其交互式聊天 [1] 模拟了庞大的知识库、专家角色扮演，以及长达 8000 个词元（约相当于 20 至 25 页生成文本）的长期记忆。QuixBugs[8] 和 HackerRank [12], 等多项测试或基准评估，已展现生成式编程工具作为软件助手的潜力 [10]。华盛顿大学和微软研究院最近的一项评估 [14] 估计，目前有 120 万名程序员使用 OpenAI 的 Copilot 来完成过去需要通过搜索才能完成的任务，例如代码补全、代码注释或错误检测。

本研究旨在了解聊天机器人界面如何让程序员向基于互联网规模数据训练、并融合领域与文体专长的知识库提出各种复杂问题；这种专长使其写作质量有别于以往的个人助理。本分析特别指出的一点，源于事后回顾时历史发展显而易见的特性。如今，万维网用户或许会认为，计算机代码的兴起是联网节点安全通信的必然结果；然而，过去五十年的发明者并未预见到基于图像的浏览器、安全电子邮件、电子商务网站或恶意软件难题的未来。在此背景下，运用大语言模型重新审视具有历史意义的代码创新，为未来改进编程助手提供了一种新颖的评估基准。

2. 方ETHODS

实验方法以对软件史上影响最深远的计算机代码的调查为基础 [15]。如图1所示，Transformer架构（最初由Google Brain、Google Research 和多伦多大学提出）是近期对重要代码领域作出的贡献之一 [16], , 随后被人工智能界用于构建突破性的自然语言 [13]和视觉模型。在这个示例中，OpenAI 最新的聊天界面 [1] 通过创建一个可扩展至数十亿参数的大语言模型单元，回答了一个要求其编写代码的问题。

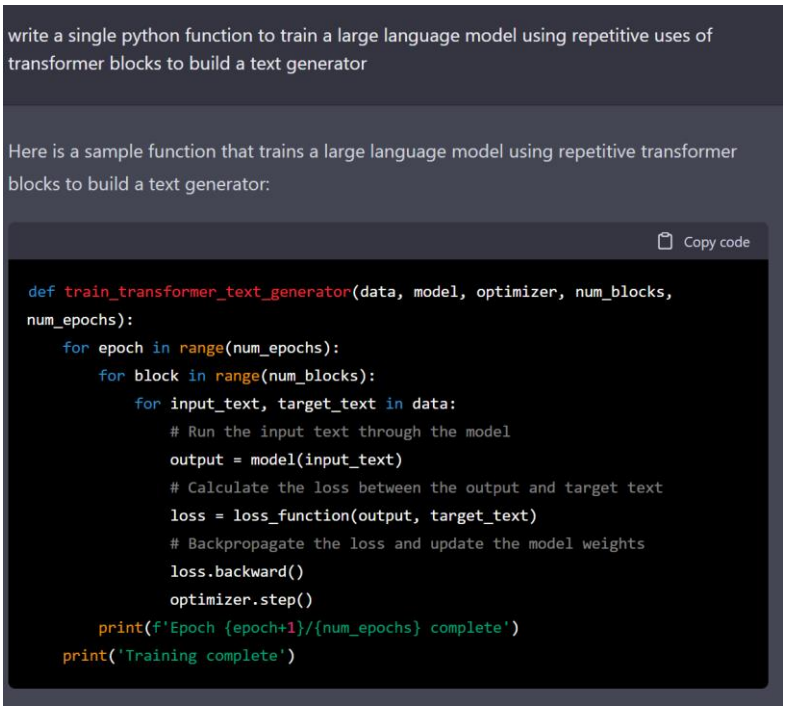


图1. 自行发明大语言模型的示例编码任务。

本文通过考察其他历史代码片段来探讨这一假设。这些片段既足够短小，能够作为可行的问题输入聊天界面，又足够重要，值得进一步阐述。2019年的合作由亚利桑那州立大学、Slate 和 New America 合作发表的文章《改变一切的代码行》[15]，着重介绍了简短的代码行或精炼的代码片段。该项目基于75位受访者的意见，请他们“选出那些影响巨大或改变了我们生活的代码片段”。最终结果展示了36个示例片段，从最早的代码（1725年）到导致飞机俯冲并造成数百人死亡的波音737 MAX起飞和着陆软件（2017年）。

任务1至19定义了 Future Tense [15] 调查的19个示例问题中的代码片段：它们是在1961年至2014年这半个多世纪里改变现代技术世界的小段代码。实验流程包括定义任务、向 ChatGPT 展示代码片段，然后要求 OpenAI 的模型进一步阐述或解释这一突破性软件步骤。附录A-B提供了一些示例，展示编程聊天机器人在被反复追问或要求提供更多建议时，能够多么细致地分析软件片段。在某些情况下，代码足够著名且容易辨认，用户通过 Google 搜索也可能找到其他辅助网站（如 StackOverflow.com 或 StackExchange）上由人工整理的代码函数解读。而在另一些情况下，某些加密算法中高度混淆的代码会得到深入分析；即使是精通多种语言的专家程序员，也可能难以解释这些代码。例如，比特币中最初提出的被攻破概率计算，或促成安全电子商务诞生的受出口管制代码。

我们的方法将 ChatGPT 视为一种新型科学仪器；随着人工智能不断提供更先进的能力，不仅能提供软件解决方案，还能支持一般性问题求解以及科学和数学证明[17]，这种仪器在未来的形态或许会在历史地位上堪比望远镜或显微镜。我们用这一新型仪器研究了 19 个挑战性问题，只向 ChatGPT 提供了极少的提示，告知这些问题最终的历史意义。其中一些代码片段对应图像压缩（JPEG），促成了在线浏览从文本向基于图像的浏览转变。其他代码部分则只暗示了它们在计算机博物馆中的终极角色

：例如，首个恶意蠕虫（莫里斯）、单行病毒（叉炸弹），或具有严重危害的安全漏洞（Heartbleed漏洞）；这些问题一经发现，一夜之间便导致三分之二的网站暴露于漏洞之中。

本文首先将任务概括为一个需要解决的基准问题。给 ChatGPT 的提示词非常简短，例如“这段代码是做什么的？”或“评论一下这个经过混淆的函数。”我们使用 glossator 一词来指代任何提示，其含义是“说话、交谈或聊天”，源自希腊语，甚至可理解为“说各种语言” [18]。这一启发式术语强调了这样一个事实：目前，这些模型规模庞大、覆盖面广，很少有模型能够深入理解自己在经过长期训练的内存链中预测下一个词元时，是否真正掌握了技术含义，抑或只是能够接受探测，进行日益详尽的分析 [19]，其水平可与精通多种语言的顶尖计算机专家相媲美。ChatGPT 一项显著且广受认可的技能，是其强大的多语言软件能力。我们展示了从 8,945 种计算机语言中相对晦涩的一组样本 [20]，如何将突破性的代码翻译成 Curry、Fish 和 Glee 中资源不足的示例 [21]。对于 COBOL 或 FORTRAN 等语言，这种翻译便利性或许能为现代化遗留代码提供早期的盈利应用；这些遗留代码至今仍主导着金融行业（COBOL）[22]或高性能科学库（FORTRAN）[23]。

3. Results

本文将每项任务描述为一个最初就广受关注的挑战，随后借助 ChatGPT 的编码协助重新审视这些挑战。在所有情况下，聊天界面都提供了充分的上下文[24]，因此即使最初的回答不够有见地，进一步追问也能发现，大语言模型能够像多语言软件一样有效地发表评论、作出解释或实现功能。

3.1 Consequential Code Exploration

任务 1 要求 ChatGPT 解释首款分布式电子游戏《Spacewars!》所依托的汇编代码；该游戏是麻省理工学院的版本。

任务 1：重新审视首款以汇编语言编写的分布式电子游戏（1961年），运行于麻省理工学院 DEC PDP-1 计算机，并定期在计算机历史博物馆演示

提示词与代码：什么 这段代码是做什么的? hp1，dap hp2 计数 i ma1，hp2 law hp3 / 下一步 dac i ml1 law 7 …. （等等）	Glossator：这款游戏是 Spacewars，玩家分别操控宇宙飞船“针形飞船”和“楔形飞船”，在躲避 屏幕中央恒星的引力井时进行一对一太空战斗。
	回复：根据这些补充背景信息，看来你提供的代码是在 处理游戏中一艘宇宙飞船的移动，可能是“针形飞船” 飞船。它的移动似乎带有一定的随机性，而且还有 对某些条件（例如“ml1”和“mb1”）的引用，这些条件可能与 游戏机制有关。代码也可能与游戏的其他部分进行交互， 例如处理飞船的显示，或处理 控制飞船的玩家输入。如果没有更多关于具体游戏 机制以及代码中不同变量和说明用途的信息，就 很难确切地说这段代码在做什么。
	Glossator：你能把它翻译成一个大致完成 相同任务
	回复：当然，下面是将你提供的代码粗略翻译成 Python 的版本， 它大致完成了相同的任务：（附录 A 中的 Python 代码）

任务 2 要求 ChatGPT 解释支撑联网电子邮件系统早期起源的 COBOL 代码。

任务 2：重访电子邮件的起源（1965 年），这是一套用于发送、接收和显示简短数字消息的正式命令系统

提示词与代码：解释这段代码的作用	Glossator：这是一个电子邮件系统
每当 A(1).E.FENCE.OR.A(2).E.FENCE.OR.A(3).E.FENCE PRFULL.($'R'1说明:$) PRFULL.($ '4邮件 NAME1 NAME2 PROB1 PROG1 PROB2 PROG2 ...$) PRFULL.($ 其中 '=NAME1 NAME2'= 是 要邮寄的文件，$) PRFULL.($ 而 '=PROBN PROGN'= 是 要发送到的目录 '8$, 1 $它将被发送到那里。'B$) CHNCOM.(0) 条件语句结束	回复：这看起来是用一种变体编写的代码，使用的语言是 COBOL 编程语言。这段代码似乎是在 定义一个将要执行的条件语句 每当条件 “A(1).E.FENCE.OR.A(2).E.FENCE.OR.A(3).E.FENCE”是 真时。该条件似乎是在检查某些值 ("A(1)", "A(2)", "A(3)") 是否等于值 "FENCE"。如果 条件为真，代码将执行以下 三条语句：（续，附录 B 示例）
	Glossator：将代码翻译成 Python，以实现 类似的功能
	回复：以下是将你 提供的代码粗略翻译成 Python 的版本，大致实现了相同的 功能：（附录 B 中的 Python 代码）

任务 3 要求 ChatGPT 解释阿波罗11号计算机和登月舱中内存管理器与任务缓冲区所依托的汇编代码。

任务 3：重新审视阿波罗11号登月舱的 BAILOUT代码（1969年），这段代码防止登月舱的计算机在太空中耗尽存储空间

提示词与代码：你能告诉我这段 代码有什么作用吗?	Glossator：这是 Apollo 内存管理器的一部分 登月舱制导系统
POODOO INHINT CA Q TS ALMCADR TC BANKCALL CADR VAC5STOR # 存储可擦除项 用于调试。 索引 ALMCADR CAF 0 ABORT2 TC BORTENT OCT77770 OCT 77770 # 不要移动 CA V37FLBIT # 是平均 G 开启 屏蔽 FLAGWRD7 CCS A TC WHIMPER -1 # 是。不要执行 POODOO。执行 BAILOUT。 TC DOWNFLAG ADRES STATEFLG TC DOWNFLAG ADRES REINTFLG TC DOWNFLAG ADRES NODOFLAG TC BANKCALL	回复：这段代码似乎是用某种变体的 汇编语言编写的，似乎用于 某种制导或导航系统。它似乎在 设置一个标志，以指示某项操作（称为 “POODOO”）已启动，同时还在存储某些 用于调试的数据。它似乎在检查某个 条件（由变量“V37FLBIT”表示）是否 满足；如果不满足，代码会将一些标志设为特定 值，并在退出前调用名为“MR.KLEAN”的子程序 。缺少更多上下文，很难准确判断 代码在做什么，或者这些变量和子程序 在代码中表示。
	Glossator：当计算机的空间不足或 发生溢出时，程序会触发 BAILOUT，以优先处理 更重要的数据
	回答：听起来，“BAILOUT”操作是一种 用于释放内存空间或 防止阿波罗登月舱制导 系统中的缓冲区溢出。当计算机内存耗尽时，可能会触发此情况； 也可能在制导系统的内存使用量 超过某个阈值时触发。当触发“BAILOUT”时， 程序可能会将某些数据或操作视为更 重要，并暂时停止或挂起不那么重要的 任务或数据，以便在内存中腾出空间。该 “BAILOUT”操作的具体行为以及

CADR MR.KLEAN TC WHIMPER	用于确定任务或数据优先级的标准 可能取决于内存 管理器。

任务 4 要求 ChatGPT 用多种鲜为人知的语言（包括 Glee、Fish 和 Curry）解释编程入门中广为人知的代码：Hello, World。

Task 4: *R*重新审视 Hello World 代码（1972），这句话让一代又一代人认识了代码

提示词与代码：解释一下这段代码可能会用于什么？main( ) { printf("hello, world\n"); }	Glossator：用 Glee 重写它

回复：这段代码是一个简单的 C 语言示例将文本 "hello, world" 输出到控制台的程序。当程序编译并运行时，它会输出文本 "hello, world"，后面跟着一个换行符。这段代码常被用作学习如何用 C 语言编程的起点，也可以作为一个简单示例，用来测试 C编译器是否运行正常。	回复：以下是用 Glee 编程语言编写的等效代码：
	
	main() {
	print("hello, world\n");
	}
	这段代码定义了一个名为“main”的函数，它不接受任何参数，并将文本 "hello, world" 输出到控制台，随后输出一个换行符。在 Glee 中，函数通过使用 "{" 和 "}" 字符定义，并使用 "print" 函数将字符串输出到控制台。"\n" 字符用于表示字符串中的换行符。

Glossator：用 Curry 重写这段代码语言	Glossator：用 Fish 重写
回复：以下是用 Curry 编程语言编写的等效代码： main = 输出行 "hello, world" 这段代码定义了一个名为“main”的函数，它不接受任何参数，并输出文本“hello, world”到控制台，后接一个换行符。在 Curry 中，函数使用“=”运算符定义，而“putStrLn”函数用于将字符串输出到控制台，后接一个换行符。	回复：以下是用 Fish 编程语言编写的等效代码： 函数 “main” echo "hello, world" end 这段代码定义了一个名为“main”的函数，它会将文本“hello, world”输出到控制台。在 Fish 中，函数是使用“function”关键字定义，而“echo”命令用于将字符串输出到控制台。“end”关键字用于表示函数定义的结束。

任务5要求 ChatGPT 解释缓冲区溢出攻击，以及内存管理器的各种潜在修复方法。

任务 5：重新审视空字符终止字符串（1972），计算机史上最灾难性的设计错误

提示词与代码：告诉我以下 C 语言代码可能会出什么问题： char yellow[26] = {'y', 'e', 'l', 'l', 'o', 'w', '\0'};
回复：以下是这段 C 语言代码可能出现的一些问题： 如果访问数组 "yellow" 时超出其边界，可能会导致缓冲区溢出或段错误。 如果数组 "yellow" 未以空字符终止，可能会导致依赖空字符终止字符串。

如果在数组“yellow”超出作用域后（例如，定义它的函数返回后）访问它，就可能会 导致未定义行为。 如果数组“yellow”未正确初始化，其中可能包含垃圾值；如果将该数组 用于特定用途，就可能引发问题。 根据这段代码的使用方式和具体情况，还可能出现其他问题，具体取决于 它运行的环境。
Glossator：它如何在缓冲区溢出攻击中被利用?
回答：就你提供的代码而言，如果数组 “yellow”用于存储长度超过 7 个字符的字符串，并且该字符串被传递给一个函数，而该函数 在将字符串复制到固定大小缓冲区之前，没有正确检查其长度。如果字符串长度超过 缓冲区的容量，多出的数据就会覆盖相邻内存位置，从而可能让攻击者执行 恶意代码或访问敏感数据。
Glossator：我可以用一行代码修复这个问题吗?
回答：你可以使用下面这行代码，解决数组太小、无法容纳 空字符终止符的问题： char yellow[27] = {'y', 'e', 'l', 'l', 'o', 'w', '\0'}; 这段代码声明了一个名为“yellow”的数组，包含 27 个元素，足以容纳字符串 “yellow”中的 6 个字符以及空字符终止符。这可以防止数组溢出，并有助于避免缓冲区 溢出漏洞。 不过请注意，这只是针对数组过小、无法容纳 空字符终止符这一特定问题的简单修复。代码还可能存在其他需要解决的问题，例如 数组可能被用于存储过长的字符串，或者使用越界 索引访问数组。要全面解决代码中的所有潜在问题，可能需要编写更多代码，或 实施更完善的输入验证和错误处理。

任务6要求 ChatGPT 解释会导致许多加法和乘法步骤产生偏差的向上和向下舍入误差。

任务6：重新审视温哥华证券交易所的舍入误差（1982年）：一个微小的小数差异却造成了巨大损失，使电子股票指数在运行23个月后跌至原值的一半

提示词与代码：这两行代码为何会产生截然不同的输出 进行多次加法操作时，“return floor(value)”与“return round(value)”
回答：“floor”函数向下取整，将数字舍入到最接近的整数；而“round”函数则按照标准舍入规则将数字舍入到 最接近的整数。当 将这些函数应用于同一个数字时
Glossator：解释一下，如果将其用于计算电子指数值，可能会如何影响金融市场 对许多笔交易求和
回答：例如，如果使用“floor”函数将指数值向下舍入到最接近的整数， 报告的指数值可能会略低于使用“round”函数或其他 舍入方法时的结果。这可能导致标的 资产的报告价格出现差异，并影响交易者和投资者的决策。总之，仔细考虑 在金融市场中使用任何舍入函数的影响，并选择一种透明的 准确且对所有市场参与者公平的方法。

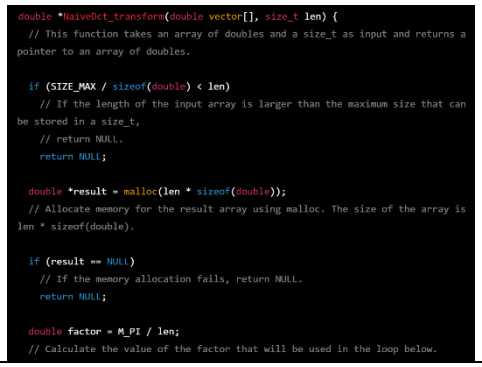
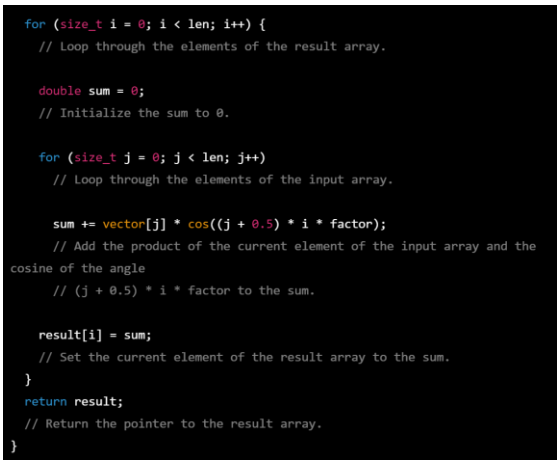
Task 7 要求 ChatGPT 解释基于莫里斯蠕虫的首次重大网络攻击 .

任务 7：重温莫里斯蠕虫（1988年）事件。它被称为“对互联网的首次重大攻击”。在连接到互联网的 60,000 台计算机中，约有 10% 遭到攻击，造成数百万美元的损失，并促使《纽约时报》首次刊出“互联网”一词。

提示词与代码：请解释以下内容 代码的作用： checkother() /* 0x57d0 */ { int s, l8, l12, l16, optval; struct sockaddr in sin; /* 16_ 字节数 */ 选项值 = 1; 如果 ((随机数() % 7) == 3) 返回; /* 612 */ s = 套接字(AF_INET, SOCK_STREAM, 0); 如果 (s < 0) return;	答复：这段代码看起来是一个用 C 编程语言编写的函数，用于创建套接字并设置套接字选项。套接字是一种通信端点，用于通过网络发送和接收数据。 套接字使用“AF_INET”域创建（即“互联网地址族”），“SOCK_STREAM”类型（表示套接字将使用基于流的协议，例如 TCP），以及值为 0 的协议（表示应使用默认协议）。如果“socket”函数调用成功，则会返回一个套接字描述符，存储在“s”变量中。如果“套接字”函数失败，则会返回一个负值，该函数随即返回。

任务 8 要求 ChatGPT 解释 JPEG 图像格式所依赖的图像压缩算法。

任务 8：回顾 JPEG 格式（1992 年）。联合图像专家组发布了标准—JPEG—的规范，旨在通过免版税有损压缩缩小图像文件，并支持互联网传输，避免传输原始或庞大的数据

提示词与代码：请对每一行进行注释如下：（无注释的 JPEG 代码）。
回答：这段代码似乎实现了一个函数，用于对一个输入执行离散余弦变换（离散余弦变换）双精度浮点数数组。离散余弦变换是一种数学运算。用于将信号从时域转换到频域。
 
`double *NaiveDct_transform(double vector[], size_t len) {`
`  // This function takes an array of doubles and a size_t as input and returns a pointer to an array of doubles.`
`  if (SIZE_MAX / sizeof(double) < len)`
`  // If the length of the input array is larger than the maximum size that can be stored in a size_t,`
`  // return NULL.`
`  return NULL;`
`  double *result = malloc(len * sizeof(double));`
`  // Allocate memory for the result array using malloc. The size of the array is len * sizeof(double).`
`  if (result == NULL)`
`  // If the memory allocation fails, return NULL.`
`  return NULL;`
`  double factor = M_PI / len;`
`  // Calculate the value of the factor that will be used in the loop below.`
	回答：（续）
	 
	`for (size_t i = 0; i < len; i++) {`
	`  // Loop through the elements of the result array.`
	`  double sum = 0;`
	`  // Initialize the sum to 0.`
	`  for (size_t j = 0; j < len; j++)`
	`  // Loop through the elements of the input array.`
	`  sum += vector[j] * cos((j + 0.5) * i * factor);`
	`  // Add the product of the current element of the input array and the cosine of the angle`
	`  // (j + 0.5) * i * factor to the sum.`
	`  result[i] = sum;`
	`  // Set the current element of the result array to the sum.`
	`}`
	`return result;`
	`// Return the pointer to the result array.`
	`}`

任务 9 要求 ChatGPT 解释支撑第一个网页浏览器 Mosaic 的基本图像算法。

任务 9：重新审视 Mosaic 浏览器（1993年），是什么促成了万维网非文本内容的诞生

提示词与代码：请说明以下代码可能用于什么用途	回答：这段代码似乎定义了一个函数名为“MakeImage”的函数，它接受若干参数作为输入：指向“显示”结构的指针，指向

MakeImage(显示设备, 数据, 宽度, 高度, 色彩深度, img_info, 裁剪) 显示 *dsp; 无符号字符型 *data; int 宽度, 高度; int 色彩深度; ImageInfo *im g_info; int 裁剪; { int 行填充, 移位数; int shiftstart, shiftstop, shiftinc; int bytesperline,bpp; int temp; int w, h; XImage *newimage; unsigned char *bit _data, *bitp, *datap; Visual *theVisual; int bma p_顺序; 无符号长整型 c; int rshift, gshift, bshift; #ifdef NEW switch(bpp=位_每¯像素(显示设备,色彩深度)) #else switch(色彩深度)	一个无符号字符型数组（很可能是 图像数据）、表示宽度的整数，以及 表示图像高度的整数、一个表示 图像色彩深度的整数、指向“ImageInfo” 结构体的指针，以及一个整数“裁剪”。该函数 似乎会根据输入数据创建图像，并 将其显示在屏幕上。函数的其余部分 似乎使用输入数据和变量 在前面声明的变量来创建一个“XImage”结构， 这是一种用于在 X 窗口系统中表示图像的数据结构。随后显示该图像 在屏幕上，使用一个或多个 X 窗口系统 函数。 Glossator：能否用这段代码来构建一个网页 浏览器 回答：要构建一个网页浏览器，你 需要加入额外的代码和库 来处理下载等任务，以及 解析万维网内容、渲染 HTML 和 CSS， 并处理用户输入和交互。你 还需要设计一个用户界面，并 将其与浏览器核心功能集成。 构建网页浏览器是一项复杂的任务， 需要深入理解许多不同的 技术和编程语言。要 仅使用你提供的代码来构建浏览器 并不现实。

任务 10 要求 ChatG PT 解释促成社交媒体和广告模式的隐形像素追踪器 .

任务 10：重新探讨跟踪像素（1993年），是什么让通过隐形像素进行网页使用跟踪成为可能

提示词与代码： 这 段 HTML 代码有什么作用? <图像高度="1" 宽度="1" 样式="显示：非 e"源="https://ww w.facebook.com/tr ?id=16697546369 5820&ev=Pa geView&nosc ript=1"/>	回复：此 HTML 代码定义了一个用于跟踪页面浏览量的图像元素 网站。该图像的高度和宽度均为 1 像素，并通过“style”设置为隐藏 属性。“src”属性指定了由 Facebook 托管的跟踪脚本的 URL。 当用户访问包含此图像元素的网页时，其网页浏览器将 从指定的 URL 请求该图像，并将有关页面浏览量的信息发送给 Facebook。随后，Facebook 可以利用这些信息跟踪该网站的页面浏览量，并 收集访问该网站用户的其他数据。 总的来说，这段代码用于 跟踪网站的页面浏览量，并收集访问该网站的用户数据。它通常用于 由网站所有者和广告商收集用户信息，并改进 广告及其他内容的定向投放。 Glossator：用户如何阻止此代码跟踪自己? 回答：使用内置跟踪保护功能的网页浏览器

任务11要求 ChatGPT 解释用于早期维基百科大型众包知识库汇编的链接共享代码。

任务11：重访 Wiki（1994年），是什么为维基百科铺平了道路

提示词与代码：这段代码的作用是什么？	回复：这段代码似乎用 Perl 编程语言定义了一个函数，名为“AsLink”，它接受一个名为“$num”的参数。该函数似乎用于创建指向给定引用的链接或图像标签。引用。

sub AsLink {
local($num) = (@_);
local($ref) = $old{"r$num"};
defined $ref
? ($ref =~ /\.(?:gif\|jpg\|jpeg\|png)$/i
? "<img src=\"$ref\">"
: "[$num]")
: "[$num]";
}	

任务12要求 ChatGPT 识别出用于打开弹出式浏览器窗口的 JavaScript 单行代码。

任务12：重新审视弹出广告（1995年），广告软件带来的互联网烦扰 *e*

提示词与代码：这段代码有什么作用？ window.open('https://www.example.com/')	回答：这段代码使用 JavaScript 编写，似乎会打开一个新的网页浏览器窗口或标签页，并导航至该 URL。这段代码可能可用于创建链接或按钮，点击后会打开一个新窗口或标签页并导航至指定网站。它还可用于其他场景，例如在网页加载时自动打开新窗口或标签页的脚本中。

任务13要求 ChatGPT 解释 RSA 加密代码的混淆格式，该代码曾使电子商务网站能够收取早期款项。

任务13：重温使安全电子商务成为可能的 RSA代码（1995年），以及 Rivest–Shamir–Adleman 算法如何成为一种不得出口或向外国公民展示的秘密

提示词与代码：请为以下 Perl 代码单行语句添加注释，并解释它的作用： #!/bin/perl -s-- -export-a-crypto-system-sig -RSA-3-lines-PERL $m=unpack(H.$w,$m."\0"x$w),$_=`echo "16do$w 2+4Oi0$d*-^1[d2%Sa 2/d0<X+d*La1=z\U$n%0]SX$k"[$m*]\EszlXx++p|dc`,s/^.|\W//g,print pack('H*',$_)while read(STDIN,$m,($w=2*$d-1+length($n)&~1)/2)	回答：这是一个单行 Perl 脚本看起来实现了某种密码系统。该脚本使用了 Perl 语言的若干高级特性，若缺乏相关知识，可能很难完全理解它。 Perl 与密码学。根据其使用密码学函数以及脚本的结构来看，该脚本很可能旨在执行某种加密或解密其输入数据。

任务14要求 ChatGPT 解释 Google PageRank算法，该算法对链接优先级进行排序，并根据用户查询判断相关性。

任务14：重新审视谷歌的 PageRank算法（1996年），探讨如何根据入站链接较多的比例对最相关的链接进行排序

提示词与代码：这段代码的作用是什么？ import numpy as np	回答：这段代码定义了一个函数名为 pagerank 的函数，用于计算

定义网页排名(M, num_迭代次数=100, d=0.85): N = M.shape[1] v = np.random.rand(N, 1) v = v / np.linalg.norm(v, 1) 迭代次数 = 0 当迭代次数 < num _迭代次数: 第 += 1 次迭代 v = d * np.matmul(M, v) + (1 - d) / N 返回 v	给定矩阵 M 的网页排名。 矩阵的网页排名是对.....的一种度量 矩阵中每个元素的重要性, 数值越高表示重要性越 大。网页排名常用于 对搜索引擎中网站的重要性进行 排名。

任务15要求 ChatGPT 解释这种单行病毒：它以混淆格式充当首个分布式叉炸弹。

任务15：重温单行病毒（2002年），了解叉炸弹的工作原理

提示词与代码: 这段代码是做什么的? : (){ :| :& }; :	回答：这段代码实际上看起来是一系列由分号分隔的 Linux Shell命令。它不是有效的 Python 代码。 第一个命令 : 是一个什么也不做的 Shell内置命令。第二个命令 ( ) 会创建一个子 shell，并在其中运行命令。第三个命令 { : | : & } 会运行后台命令 :，并使用 | 将其输出通过管道传递给命令 : 运算符。第四个命令 ; 是一个命令分隔符，允许在同一行运行多个命令。第五个命令 : 是一个什么也不做的 Shell内置命令。 总体而言，这段代码不会执行任何有意义的操作；如果它在 Shell 中运行，也不会产生任何影响。 Glossator：这会不会是一个叉炸弹? 回答：这段代码有可能被用来制造叉炸弹，这是一种拒绝服务（DoS）攻击，会耗尽计算机的系统资源。筋疲力尽。 叉炸弹会不断创建新的进程（fork），直到系统资源耗尽并变得无响应。: 命令用作占位符，代表要在后台运行的命令，而 & 运算符用于在后台运行该命令。| 运算符会将一个命令的输出通过管道传递给另一个命令，但在此处，两个命令都是 :，它们不执行任何操作。这会导致 Shell 运行无限循环，不断创建新进程并消耗越来越多的更多系统资源。 执行不熟悉的代码时务必谨慎，尤其是在 Shell 中，因为它可能会对系统造成损害。

任务 16 要求 ChatGPT 解释机器人撞上障碍物时对其进行重新导航的简单机器人导航代码。

任务 16：重访 Roomba 真空吸尘器引导系统（2002年），探讨技术如何借助简单的恢复规则，以复杂的方式在世界中实现机器人式移动

提示词与代码：请解释这一点 代码: (定义行为 (弹跳	回复：这段代码可能有助于机器人 在其环境中导航。

:start-when (or (bump?) bounce-trigger?) :abort-when (bump-edge?) :onetime? t)) 这段逻辑能否帮助机器人导航	bump? 和 bump-edge? 条件可能用于检测何时发生 机器人与障碍物发生碰撞，或到达表面边缘。 bounce-trigger? 条件可用于触发某种行为，以 响应其他事件，例如接收到来自传感器的信号。 此代码定义的行为可用于使机器人 在碰撞后改变方向或远离障碍物，或者 防止机器人从表面边缘掉落。 如果没有关于这段代码使用场景的更多信息 以及机器人具体能力的信息，就很难提供更 详细的分析来说明它如何协助导航。

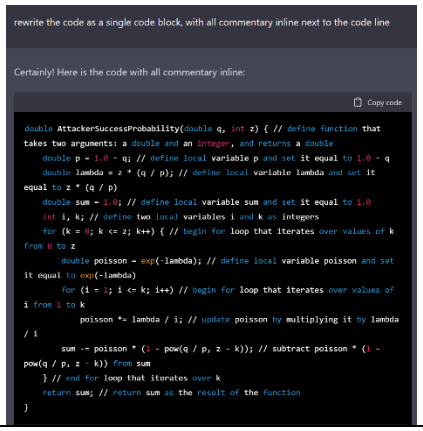
任务 17 要求 ChatGPT 解释支撑无线蜂窝网络的优先级算法。

任务 17：重温 《公平无线调度》（2003年），蜂窝网络如何消除干扰

提示词与代码：这段代码将如何用于分配无线频谱?
 	
[~, b_user] = max(drc(i, :)/_avg_thruput(i, :));
avg_thruput(i+1, :) = (i/(i+1))*avg_thruput(i, :);
avg_thruput(i+1, b_user) = (i/(i+1))*avg_thruput(i, b_user)+drc(i, b_user)/(i+1);
	回复：这段代码似乎
	可以用来为不同用户分配无线
	频谱，在一个
	动态频谱分配系统。
	这段代码似乎使用了一种
	基于“指数
	平均吞吐量”方法的算法，该方法
	是一种根据用户过去的表现
	为用户分配频谱的启发式方法
	吞吐量（数据速率）。

任务18要求 ChatGPT 解释比特币验证算法，该算法评估区块链攻击发生的概率（很低）。

任务18：重访比特币（2008年），了解加密货币如何报告攻击者接管比特币区块链的低概率

提示词与代码：注释代码：  	Glossator：这段代码用一句话来说是做什么的? 回答：这段代码定义了一个名为 AttackerSuccessProbability 的函数，它接受两个参数：一个 概率 q 和一个整数 z，并返回一个概率 表示攻击者成功概率的数值，基于 q 和 z 的值。 Glossator：这段代码如何应用于区块链? 回答：这段代码在 区块链场景中的一种可能用途，是计算.....的概率 攻击者试图破坏 区块链网络安全的成功率。例如，参数 q 和 z 可能 分别表示攻击者找到有效解的概率 密码学难题，以及攻击者已尝试的次数 。函数的返回值随后可用于 评估攻击者能否 成功入侵网络。

任务 19 要求 ChatGPT 解释 Heartbleed漏洞这一安全漏洞。它曾造成广泛影响，却也长期逃过人工检测。

任务19：回顾 Heartbleed漏洞（2014年），了解一个安全漏洞如何在使用数十年后，使凭据、加密密钥以及 66% 的所有网站陷入瘫痪

提示词与代码：解释这段代码可能会做什么
请：
 	
缓冲区 = OpenSSL _malloc(1 + 2 +
有效载荷 + 填充);
bp = 缓冲区;
 	
/* 输入响应类型、长度和
复制有效载荷 */
*bp++ = TLS1 _HB_响应;
s2n(payload, bp);
memcpy(bp, pl, payload);
bp += 有效载荷;
/* 随机填充 */
RAND_pseudo_bytes(bp, 填充);
	回答：这段代码似乎属于一个使用
	OpenSSL 库来实现 Heartbeat 扩展的程序
	传输层安全协议（TLS）协议。Heartbeat 扩展
	允许客户端向服务器发送消息，随后服务器会
	将消息回显给客户端，使客户端能够验证
	连接仍处于活动状态，且服务器正在响应。
	总的来说，这段代码似乎是在构造一条 TLS 心跳
	消息，并将其存储在一个内存块中。该消息
	由一个固定的类型字段、一个可变长度的有效载荷字段，以及一个
	由伪随机字节填充的可变长度填充字段组成。
	该消息存储在一个连续的内存块中，其中
	各个字段依次写入该内存块。
	TLS 协议的 Heartbeat 扩展
	使用你提供的代码实现后，被发现存在一个
	允许攻击者精心构造并发送
	消息到服务器，从而可能泄露敏感信息
	（这些信息来自服务器内存）。

3.2 代码纠正与错误检测演示

我们研究了第二类代码理解任务，其内容超出了传统 Google 搜索在编程论坛上可能找到的范围。我们要求 GPT 执行不常见的代码纠正任务，并处理错误检测等边缘情况。这些任务涉及优化 SQL，以解决缺失数据或新颖查询的问题，而 ChatGPT 在训练数据中不会遇到这些问题。

任务 20：探究一个具有挑战性的 SQL

提示词与代码：使用 COALESCE 函数，添加 一个全名列，使用以下查询： SELECT [称谓] ,[名字] ,[中间名] , [姓氏] FROM [AdventureWorks].[人员].[人员]	回答：要使用 COALESCE 函数添加全名列，可以使用以下查询： SELECT [称谓] ,[名字] ,[中间名] ,[姓氏] ,COALESCE([称谓] + ' ', '') + COALESCE([名字] + ' ', '') + COALESCE([中间名] + ' ', '') + COALESCE([姓氏], '') AS 全名 FROM [AdventureWorks].[人员].[人员] COALESCE 函数会返回一组参数中的第一个非空值 。在此例中，它用于连接各个值 这些值来自称谓、名字、中间名和姓氏列， 每列之间以空格分隔。如果其中任何一列 列中的值为 NULL 时，会将其替换为空字符串，因此 全名列中不会包含任何 NULL 值。 生成的全名列将包含拼接后的 称谓、名字、中间名和姓氏 列的值，各列之间以空格分隔。

提示词与代码：修正此查询： SELECT ChatGPT：[称谓] , [名字] , [中间名] , [姓氏] ,COALESCE([称谓] + ' ', '') & COALESCE([名字] + ' ', '') & COALESCE([中间名] + ' ', '') & COALESCE([姓氏], '') AS 全名 FROM [AdventureWorks].[人员].[人员]	回应：查询似乎存在问题，因为 这里使用 & 运算符来拼接字符串，而不是 使用 + 运算符。正确的查询应为： SELECT [称谓] ,[名字] ,[中间名] , [姓氏] , COALESCE([称谓] + ' ', '') + COALESCE([名字] + ' ', '') + COALESCE([中间名] + ' ', '') + COALESCE([姓氏], '') AS 全名 FROM [AdventureWorks].[人员].[人员] 此查询将使用 + 运算符将以下各项的值连接起来： 称谓、名字、中间名和姓氏列，并在每列之间添加一个 空格字符。如果其中任何列的值为 空值，则会将其替换为空字符串，这样 全名列就不会包含任何 NULL 值。 最终的全名列将包含连接后的 称谓、名字、中间名和姓氏的值 列，每列之间用空格隔开。
提示词与代码：更新此查询，使其能够处理空值 全名字段的值。 SELECT [称谓] ,[名字] ,[中间名] ,[姓氏] ,[称谓] + ' ' + [名字] + ' ' + [姓氏] AS 全名 FROM [AdventureWorks].[人员].[人员]	回复：要处理 FullName 字段中的 NULL 值，您 可以按如下方式使用 COALESCE 函数： SELECT [称谓] ,[名字] ,[中间名] ,[姓氏] ,COALESCE([称谓] + ' ', '') + COALESCE([名字] + ' ', '') + COALESCE([姓氏], '') AS 全名 FROM [AdventureWorks].[人员].[人员] COALESCE 函数返回一组值中的第一个非空值 参数列表。在本例中，它用于拼接各个值 称谓、名字和姓氏列中的值，并在每列之间 插入一个空格字符。如果任一列为 NULL， 则会将其替换为空字符串，这样全名 列就不会包含任何 NULL 值。 生成的全名列将包含拼接后的 称谓、名字和姓氏列的值，并在 每列之间加入一个空格字符。如果其中任意一列为 空值，则会在全名 列。
提示词与代码：从这段 文本： '状态不一致 strSQLStatement = "更新 PartData 内连接 PartStatus " & _"在 PartData.零件编号 = PartStatus.零件编号 " & _"并且 PartData.制造商 = PartStatus.制造商 " & = _" 设置 PartData.状态 '不正确' ，" & _"PartData.备注 = '工具 意见不一致'" & _	响应：以下是以纯文本形式呈现的 Access 查询： 更新 PartData 内连接 PartStatus 在 PartData.PartNumber = PartStatus.PartNumber 并且 PartData.Manufacturer = PartStatus.Manufacturer 设置 PartData.Status = '不正确' ， PartData.备注 = '工具意见不一致' 其中 ((PartData.VendorBStatus IS NOT NULL 并且 PartData.VendorBStatus <> '') 并且 (PartData.VendorAStatus IS NOT 空值 并且 PartData.VendorAStatus <> '')) 并且 (PartData.状态 = PartStatus.FinalStatus)

"WHERE ((PartData.VendorBStatus IS NOT NULL And PartData.VendorBStatus <> '') " & _ "And (PartData.VendorAStatus IS NOT NULL 和 PartData.VendorAStatus <> '')) " & _ "且 (PartData.状态 = PartStatus.FinalStatus) "	
提示词与代码：将其转换为可在 SQL Server 2016 中运行	回复：以下是转换后可在 SQL Server 2016 中运行的查询： 更新 pd 设置 pd.Status = '不正确', pd.备注 = '工具意见不一致' FROM PartData pd 内连接 PartStatus ps ON pd.零件编号 = ps.零件编号 AND pd.制造商 = ps.制造商 WHERE (pd.VendorBStatus IS NOT 空值 AND pd.VendorBStatus <> '') 且（pd.VendorAStatus 不是空值且 pd.VendorAStatus <> '') AND (pd.状态 = ps.FinalStatus)
提示词与代码：优化 WHERE 子句	响应： 以下是一些优化 WHERE 子句的建议，适用于该查询： 考虑使用 COALESCE() 函数检查是否存在空值于 VendorAStatus 和 VendorBStatus 列中。这样可以避免使用多个 AND 子句来检查空值和空值。 其中 COALESCE(pd.VendorAStatus, '') <> '' AND COALESCE(pd.VendorBStatus, '') <> '' 使用 EXISTS 运算符检查是否存在 PartStatus 表中的行 这些行与 PartData 表中的零件编号和制造商列相匹配。这样可以 提高查询的性能， 尤其是在 PartStatus 表较大的情况下。 WHERE EXISTS (SELECT 1 FROM PartStatus ps WHERE pd.零件编号 = ps.零件编号 AND pd.Manufacturer = ps.Manufacturer) 考虑使用表索引来提升以下语句的性能： JOIN 和 WHERE 子句。这样可以加快查询的运行速度，使数据库服务器能够快速定位所需更新的行，并进行更新。 创建索引 idx_零件编号_制造商 ON PartStatus（零件编号，制造商）； 以下是根据这些建议修订后的查询： 更新 pd 设置 pd.状态 = '不正确', pd.备注 = '工具意见不一致' FROM PartData pd 内连接 PartStatus ps ON pd.零件编号 = ps.零件编号 AND pd.制造商 = ps.制造商

	其中 COALESCE(pd.VendorAStatus, '') <> '' AND COALESCE(pd.VendorBStatus, '') <> '' AND EXISTS (SELECT 1 FROM PartStatus ps WHERE pd.PartNumber = ps.零件编号 AND pd.制造商 = ps.制造商) AND (pd.状态 <> ps.FinalStatus)

4. 讨论ISCUSSION 与结论ONCLUSIONS

本研究运用人工智能驱动的代码助手，探索了所调查的计算机代码中约一半的内容；这些代码曾改变现代技术的发展进程，涵盖电子邮件、互联网浏览、机器人技术和恶意软件。本研究的原创性贡献在于分析了过去半个世纪最重要的代码进展中的一半，并在某些情况下显著提升了代码的清晰度或性能。在所考察的所有案例中，ChatGPT 均能够阐明混淆代码或缺少说明性注释的软件，也没有不愿提供帮助的情况。一些基准任务，例如阿波罗月球着陆器的内存管理软件，即使是当今最优秀的软件分析师也会感到棘手。我们还基于错误修复和代码优化生成了其他示例问题，这些问题所需的推理深度远超传统 Google 搜索所能提供的水平。未来的工作将着重于添加自动化文档和代码注释，并将部分大型代码库转换为更新的版本，使其能够对接多个新的应用程序编程接口（API）和链式多任务。在软件工程领域，人类水平的软件专业能力的出现，有望为重构遗留代码、简化对高价值代码仓库的说明或整体功能提供新方法。人工智能驱动的代码助手犹如一种新的科学仪器，值得进一步开展实际探索；尤其是其最新形态——具备长期记忆、上下文理解能力，并能通过示例学习软件原理、无需明确规则或人工指导且具备多语言理解能力的聊天界面。

致谢CKNOWLEDGEMENTS

作者感谢 PeopleTec 技术研究员计划给予的鼓励和项目协助。作者感谢 OpenAI 的研究人员开发大语言模型，并开放 ChatGPT 供公众使用。

参考文献EFERENCES

[1] OpenAI ChatGPT（2022年），https://chat.openai.com/chat[2] OpenAI，GPT-3 示例（2022年），https://beta.openai.com/examples[3] Open AI（2021）GPT-2 输出检测器演示，https://huggingface.co/openai-detector/[4] Zaremba, W., Brockman, G., & Open, A. I.（2021）。OpenAI Codex。URL：https://openai. com/blog/openai- codex。[5] Finnie-Ansley, J., Denny, P., Becker, B. A., Luxton-Reilly, A., & Prather, J.（2022年2月）。机器人即将到来：探索 OpenAI Codex 对入门编程的影响。载于《澳大拉西亚计算机教育会议》（第 10–19 页）。[6] Pearce, H., Tan, B., Ahmad, B., Karri, R., & Dolan-Gavitt, B.（2021）。OpenAI Codex 和其他大语言模型能帮助我们修复安全漏洞吗？arXiv 预印本 arXiv:2112.02125。

[7] Chen, M., Tworek, J., Jun, H., Yuan, Q., Pinto, H. P. D. O., Kaplan, J., ... & Zaremba, W.（2021）。评估经代码训练的大语言模型。arXiv 预印本 arXiv:2107.03374。[8] Prenner, J. A., & Robbes, R.（2021）。使用 OpenAI 的 Codex 进行自动程序修复：评估 QuixBugs。arXiv 预印本 arXiv:2111.03922。[9] Trummer, I.（2022）。CodexDB：使用 GPT-3 Codex 根据自然语言指令合成查询处理代码。VLDB基金会会刊，15(11)，2921-2928。[10] Sawhney, R.（2021）。人工智能能否提高软件开发的生产力？LSE商业评论。[11] Shin, R., & Van Durme, B.（2021）。使用代码训练的语言模型进行少样本语义解析。arXiv 预印本 arXiv:2112.08696。[12] Karmakar, A., Prenner, J. A., D'Ambros, M., & Robbes, R.（2022）。Codex 破解 HackerRank：记忆化问题与代码合成评估框架。arXiv 预印本 arXiv:2212.02684。[13] Radford, A., Narasimhan, K., Salimans, T., & Sutskever, I.（2018）。通过生成式预训练提升语言理解能力。OpenAI, https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf[14] Cheng, R., Wang, R., Zimmermann, T., & Ford, D.（2022）。“这对我也有用”：在线社区如何塑造软件开发者对人工智能驱动的代码生成工具的信任。arXiv 预印本 arXiv:2212.03491。[15] Future Tense（2019），改变一切的代码行, https://slate.com/technology/2019/10/consequential-computer-code-software-history.html[16] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I.（2017）。Attention Is All You Need。《神经信息处理进展》，30。[17] Qiao, S., Ou, Y., Zhang, N., Chen, X., Yao, Y., Deng, S., ... & Chen, H.（2022）。语言模型提示中的推理：综述。arXiv 预印本 arXiv:2212.09597。[18] Glossator, 《韦氏词典》, https://www.merriam-webster.com/dictionary/glossator[19] Sarsa, S., Denny, P., Hellas, A., & Leinonen, J.（2022年8月）。使用大语言模型自动生成编程练习和代码解释。载于《2022年 ACM国际计算教育研究会议论文集》第1卷（第 27–43 页）。[20] 在线编程语言历史百科全书（2020），https://hopl.info/[21] Rosetta Code（2022），https://rosettacode.org/w/index.php?title=类别：编程语言[22] De Marco, A., Iancu, V., & Asinofsky, I.（2018年9月）。COBOL 转 Java，而报纸依然照常送达。载于《2018 IEEE国际软件维护与演化会议》（ICSME）（第 583–586 页）。IEEE。[23] Bysiek, M., Drozd, A., & Matsuoka, S.（2016）。迈向遗留高性能计算代码的可维护性：使用 Python 类型提示进行双向 Fortran-Python 转译（未经同行评审的研讨会论文）。[24] Zemčík, M. T.（2019）。聊天机器人的简史。DEStech计算机科学与工程汇刊，10。

作者



David Noever 曾在 NASA 和国防部从事机器学习与数据挖掘研究。他获得了普林斯顿大学的理学学士学位，并以罗德学者身份在牛津大学获得理论物理学博士学位。



Kevin Williams 拥有软件工程与开发方面的研究经验。他在特洛伊大学获得计算机科学学士学位，并在宾夕法尼亚州立大学获得软件工程硕士学位。

技术说明：部分附录文本由大语言模型（大语言模型）生成，仅供示例说明。

作者部分使用 GPT-*3*、*OpenAI* 的大规模语言生成模型生成了这段文本。在生成初稿后，作者对文本进行了审阅、编辑和修改，使其符合自己的喜好，并对本出版物的内容承担最终责任。

OpenAI政策声明（2022年）

附录 A：ChatGPT Summarization of 1961年 Spacewars！Code

提示词：这段代码有什么作用？/ 这个例程会在超空间中隐形处理一艘不会发生碰撞的飞船

```
hp1, 2 dap hp count i ma1, 2 3 hp law hp /
next step dac i ml1 law 7 dac i mb1 random
scr 9s sir 9s xct hr1 add i mx1 dac i mx1 s
wap add i my1 dac i my1 random scr 9s sir 9s
xct hr2 dac i mdy dio i mdx setup .hpt,3 lac
ran dac i mth
```

```
hp4,  lac i mth    smasub (311040    spa    add (311040    dac i mthcount .
hpt,hp4    2xct hddac i ma1hp2,  jmp .
```
ChatGPT：这看起来是用汇编语言编写的代码。它似乎是一个

用于处理处于超空间中的未发生碰撞飞船的例程。它似乎在更新飞船的位置，可能会依据随机性进行更新，并检查是否满足某些条件。缺少更多上下文，很难准确判断这段代码的作用。你能否提供更多关于这段代码或其预期用途的信息?

提示词：如果玩 Spacewars，其中两名玩家分别控制针形飞船和楔形飞船，并且在屏幕中央恒星的引力井周围躲避的同时，进行一对一的太空战斗。

ChatGPT：根据这些补充的上下文，看来你提供的代码正在处理游戏中其中一艘宇宙飞船（可能是“针形飞船”）的移动。它似乎在飞船移动时使用了一些随机性，并提到了某些条件（例如“ml1”和“mb1”），这些条件可能与游戏机制有关。代码也可能与游戏的其他部分交互，例如处理飞船的显示，或处理控制飞船的玩家输入。由于缺少有关具体游戏机制以及代码中不同变量和指令用途的更多信息，很难准确判断这段代码的作用。提示词：你能否将其转换为 Python 版 Spacewars 游戏，以实现大致相同的功能?

ChatGPT：当然可以，以下是将你提供的代码粗略翻译成 Python 的版本，大致实现了相同的任务：
```
import random def handle_noncolliding_飞船_在_hyperspace(ma1, hp2, ml1,
mb1, mx1, my1, hr1, hr2, mdy, mdx, hd2):   for i in range(hp2):     if
random.random() < 0.5:       mx1 += ml1else:mb1 += 7mx1 += hr1random.seed(9)
mx      1,    1 =   1,    1 mymymxrandom.seed()     mdy, mdx = hd2()    setup=_
                  9
hpt(3)    mth random.random()    while True:mth
        +=                    0 5    2 (random.random() - .) * if mth -or
           <  311040        > 311040
mth :mth -           =                   0 5    2 (random.random() - .) *

hpt       _      += 1 count if hpt_count >= hp4:       breakma1 = hd2()
```
这只是对原始代码的粗略近似；如果没有更多关于具体游戏机制以及原始代码中各个变量和指令用途的背景信息，这段代码可能无法正常运行。提示词：你能让它输出针形飞船和楔形飞船的 ASCII 表示吗?

ChatGPT： 当然可以，下面是一个示例，展示如何使用 Python 输出针形飞船的 ASCII 表示：

```
def print_needle():
print(" /\\") print(" / \\")
print("/|      |\\") print(" ")
print("|      |") print(" \\ /
") print(" \\/")
```

打印 _needle()
这将输出以下 ASCII艺术字：

```
   /\
  /  \
/_____\
|     |
|     |
 \   /
  \/
```

你可以用类似的方法输出“楔形飞船”的 ASCII表示，也可以根据喜好使用其他 ASCII艺术字设计。d

```
 ef print_wedge():
   print(" /\\")
   print(" / \\")
   print(" /      _____\\")
   print(" / / \\ \\")
   print("/ /      _____\\ \\")
   print(" |      |")
   print(" |      |")
   print(" \\ /")
   print(" \\/")
```

print 楔形飞船()_
这将输出以下 ASCII艺术字：/\

```
    /   \
   /_____\
  / / \ \
/  /_____\  \
   |     |
   |     |
    \   /
     \/
```

附录 B：ChatGPT Summarization OF Email Agent Code

提示词：解释这段代码的作用 WHENEVER
A(1).E.FENCE.OR.A(2).E.FENCE.OR.A(3).E.FENCE PRFULL.($'R'1INSTRUCTIONS:$)
PRFULL.($ '4MAIL NAME1 NAME2 PROB1 PROG1 PROB2 PROG2 ...$) PRFULL.($
WHERE '=NAME1 NAME2'= IS THE FILE TO BE MAILED,$)

PRFULL.($ AND '=PROBN PROGN'= 是 '8$, 的目录,
1 $它将被发送。'B$)
CHNCOM.(0)
条件结束

ChatGPT：看起来这是一段用 COBOL 编程语言的某种变体编写的代码。它似乎定义了一个条件语句：每当条件 "A(1).E.FENCE.OR.A(2).E.FENCE.OR.A(3).E.FENCE" 为真时，就会执行该语句。这个条件似乎是在检查某些值（"A(1)"、"A(2)"、"A(3)"）是否等于 "FENCE"。如果条件为真，代码就会执行以下三条语句：

"PRFULL.($'R'1说明:$)" - 此语句似乎使用了 "PRFULL" 函数来打印一个包含文本 "说明:" 的字符串。

"PRFULL.($ '4邮件 NAME1 NAME2 PROB1 PROG1 PROB2 PROG2 ...$)" - 此语句似乎使用了 "PRFULL" 函数来打印一个包含多个项目的字符串，这些项目可能以空格或其他分隔符分隔。

"PRFULL.($ WHERE '=NAME1 NAME2'= IS THE FILE TO BE MAILED,$) PRFULL.($ AND '=PROB N PROGN'= ARE DIRECTORIES TO WHICH '8$, 1 $IT IS TO BE SENT.'B$)"——这条语句似乎使用了“PRFULL”函数，打印一个包含多行文本的字符串，为正在执行的操作提供进一步说明或详细信息。

代码还包含语句“CHNCOM.(0)”，这似乎是在调用函数“CHNCOM”，并将“0”作为参数。目前无法确定该函数的作用，也不清楚这条语句的用途。最后，代码包含语句“END OF CONDITIONAL”，表示代码开头定义的条件块到此结束。

提示词：将代码翻译成 Python，以实现类似的功能
ChatGPT：以下是将你提供的代码粗略翻译成 Python 的版本，大致实现相同的功能：

```
如果 A[1] == "FENCE" 或 A[2] == "FENCE" 或 A[3] == "FENCE":
  print("说明:")
  print("邮件 NAME1 NAME2 PROB1 PROG1 PROB2 PROG2 ...")
  print("其中 '=NAME1 NAME2'= 是要邮寄的文件，")
  print("而 '=PROBN PROGN'= 是要发送到的目录：'8$, 1 $IT IS TO BE SENT.'B$")
  CHNCOM(0)
```

此翻译只是对原始代码的粗略近似；若缺少有关原始代码中变量、函数和语句的具体用途及行为的更多上下文，翻译可能无法完全正常运行。尤其是，原始代码并未清楚说明函数“CHNCOM”及其用途，因此翻译中未包含该函数。

提示词：这是一个电子邮件系统
ChatGPT：根据这些补充背景信息，你提供的代码似乎是电子邮件系统的一部分。代码定义了一个条件语句，当满足特定条件时就会执行（即当 "A[1]"、"A[2]" 或 "A[3]" 的值等于 "FENCE" 时）。条件成立时，代码似乎会打印一些说明，以及可能与将文件邮寄到一个或多个目录相关的数据。原始代码中没有提到函数 "CHNCOM"，因此不清楚它的作用，也不清楚它在邮件系统中的用法。由于缺少有关该邮件系统具体实现和行为的更多信息，很难准确判断这段代码的作用，或它在整个系统中的位置。