

第一次作业-调研报告

2113353 张一帆

Github Copilot

产品概述

GitHub Copilot 是一款 AI 编码助手，可帮助你更快、更省力地编写代码，从而将更多精力集中在问题解决和协作上。Copilot 会在你键入时提供编码建议：有时是当前行的补全，有时是全新的代码块。可以接受全部或部分建议，也可以忽略建议并继续键入。

使用聊天功能，可以询问 Copilot 解决问题的最佳方法。或者，也可以要求 Copilot 解释其他人的代码。如果你的代码有错误，可以询问 Copilot 如何修复它。

GitHub Copilot 已被证明可以提高开发人员的工作效率并加快软件开发的步伐。

代码生成

利用注释生成：

中文注释

生成代码速度相对较快，且一次提示给出的代码较为完整

给出了较为详细的注释，代码可读性好

```

test > test2.cpp > main()
1 //请为我生成一段代码，实现冒泡排序
2 #include <iostream>
3 using namespace std;
4
5 void bubbleSort(int arr[], int n) {
6     // 冒泡排序
7     for (int i = 0; i < n-1; i++) {
8         for (int j = 0; j < n-i-1; j++) {
9             if (arr[j] > arr[j+1]) {
10                // 交换 arr[j] 和 arr[j+1]
11                int temp = arr[j];
12                arr[j] = arr[j+1];
13                arr[j+1] = temp;
14            }
15        }
16    }
17 }
18
19 int main() {
20     // 定义一个数组
21     int arr[] = {64, 34, 25, 12, 22, 11, 90};
22     // 获取数组的长度
23     int n = sizeof(arr)/sizeof(arr[0]);
24     // 调用冒泡排序函数
25     bubbleSort(arr, n);
26     // 打印排序后的数组
27     cout << "Sorted array: \n";
28     for (int i=0; i < n; i++) {
29         cout << arr[i] << " ";
30     }
31     return 0;
32 }

```

英文注释

能够正确识别，生成速度和内容与中文注释相似

```
test > test.cpp > main()
1 //Please generate a code snippet for me to implement bubble sort.
2
3 #include <iostream>
4 using namespace std;
5
6 void bubbleSort(int arr[], int n) {
7     // 冒泡排序
8     for (int i = 0; i < n-1; i++) {
9         for (int j = 0; j < n-i-1; j++) {
10             if (arr[j] > arr[j+1]) {
11                 // 交换 arr[j] 和 arr[j+1]
12                 int temp = arr[j];
13                 arr[j] = arr[j+1];
14                 arr[j+1] = temp;
15             }
16         }
17     }
18 }
19
20 int main() {
21     // 定义一个数组
22     int arr[] = {64, 34, 25, 12, 22, 11, 90};
23     // 获取数组的长度
24     int n = sizeof(arr)/sizeof(arr[0]);
25     // 调用冒泡排序函数
26     bubbleSort(arr, n);
27     // 打印排序后的数组
28     cout << "Sorted array: \n";
29     for (int i=0; i < n; i++) {
30         cout << arr[i] << " ";
31     }
32     return 0;
33 }
```

补全代码

对于冒泡函数缺失的部分，可根据上下文进行补全

```

15 void bubbleSort(int arr[], int n) {
16     for (int i=0; i < n-1; i++) {
17         for (int j=0; j < n-i-1; j++) {
18             ✨ if (arr[j] > arr[j+1]) {
19                 // Swap the elements
19                 int temp = arr[j];
19                 arr[j] = arr[j+1];
19                 arr[j+1] = temp;
20             }
21         }
22     }
23 }

```

内联聊天

可在代码的任意位置开始内联聊天，智能问答更具针对性，同时更加灵活

Copilot	>	开始内联聊天	Ctrl+I
重命名符号	F2	对此进行解释	
更改所有匹配项	Ctrl+F2	修复此	
格式化文档	Shift+Alt+F	生成文档	
使用...格式化文档		生成测试	

代码解释

利用Copilot Chat对之前的冒泡排序代码进行解释

首先对代码的主体功能进行阐述，之后分别解释每个函数的具体功能和作用

这段代码主要实现了冒泡排序算法。冒泡排序是一种简单的排序算法，它重复地遍历要排序的数列，一次比较两个元素，如果他们的顺序错误就把他们交换过来。遍历数列的工作是重复地进行直到没有再需要交换，也就是说该数列已经排序完成。

首先，我们看到 `bubbleSort` 函数，它接受一个整数数组和一个整数 `n` 作为参数，其中 `n` 是数组的大小。在这个函数中，我们有两个嵌套的 `for` 循环。外层循环从 0 到 `n-1`，内层循环从 0 到 `n-i-1`。在内层循环中，我们比较相邻的两个元素，如果它们的顺序是错误的（即，第一个元素大于第二个元素），我们就交换它们。

然后，我们看到 `main` 函数。在这个函数中，我们首先定义了一个整数数组 `arr`，然后计算数组的长度 `n`。然后，我们调用 `bubbleSort` 函数对数组进行排序。最后，我们打印出排序后的数组。

这段代码的主要目的是展示冒泡排序算法的工作原理，以及如何在 C++ 中实现它。

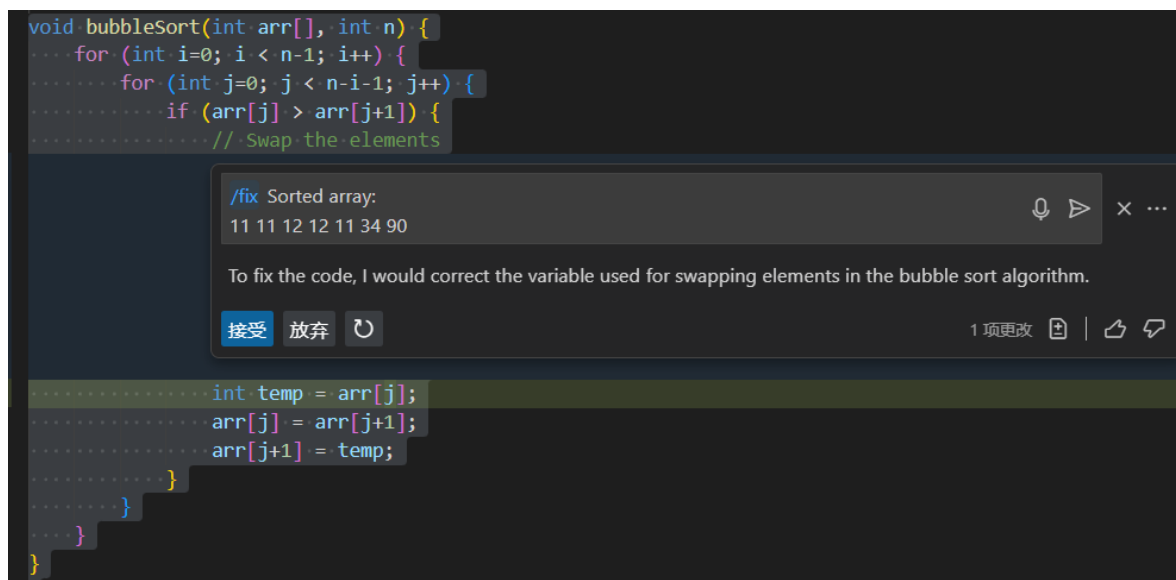
请问如何在 Visual Studio Code 中运行 C++ 代码？

修复代码

将红框部分的 `j` 改为 `i`，虽没有报错，但函数逻辑出现错误

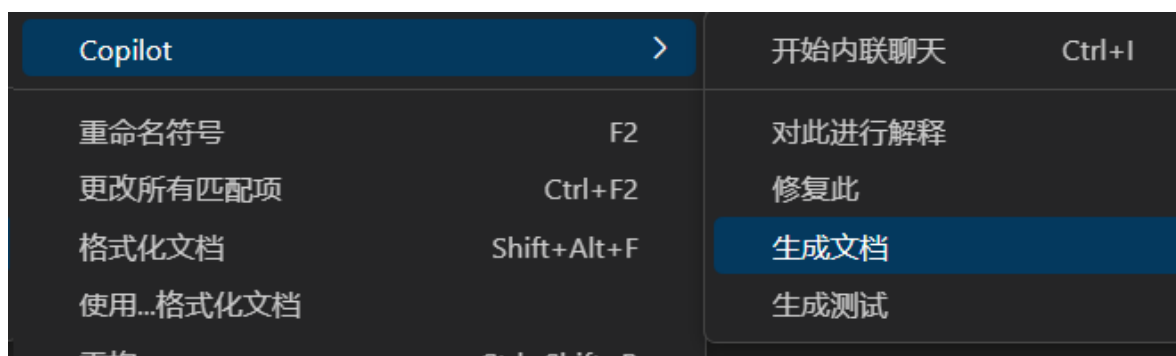
```
15 void bubbleSort(int arr[], int n) {
16     for (int i=0; i < n-1; i++) {
17         for (int j=0; j < n-i-1; j++) {
18             if (arr[j] > arr[j+1]) {
19                 // Swap the elements
20                 int temp = arr[i];
21                 arr[j] = arr[j+1];
22                 arr[j+1] = temp;
23             }
24         }
25     }
26 }
```

利用Copilot的代码修复功能，并将错误的排序结果作为补充，即可快速地发现并修改错误



生成文档

利用该功能，可快速规范代码格式，使其能够作为正式的文档



下图所示为经过规范过后的代码文档：

```

test > test2.cpp > ...
1  /**
2   * @file test2.cpp
3   * @brief This file contains an implementation of the bubble sort algorithm.
4   */
5
6  #include <iostream>
7  using namespace std;
8
9  /**
10   * @brief Sorts an array using the bubble sort algorithm.
11   *
12   * @param arr The array to be sorted.
13   * @param n The size of the array.
14   */
15  void bubbleSort(int arr[], int n) {
16      // Bubble sort
17      for (int i = 0; i < n-1; i++) {
18          for (int j = 0; j < n-i-1; j++) {
19              if (arr[j] > arr[j+1]) {
20                  // Swap arr[j] and arr[j+1]
21                  int temp = arr[j];
22                  arr[j] = arr[j+1];
23                  arr[j+1] = temp;
24              }
25          }
26      }
27  }
28
29  int main() {
30      // Define an array
31      int arr[] = {64, 34, 25, 12, 22, 11, 90};
32      // Get the length of the array
33      int n = sizeof(arr)/sizeof(arr[0]);
34      // Call the bubble sort function
35      bubbleSort(arr, n);
36      // Print the sorted array
37      cout << "Sorted array: \n";
38      for (int i=0; i < n; i++) {
39          cout << arr[i] << " ";

```

生成测试

Copilot允许对选中的函数生成详细的测试用例，以测试函数的准确性

Copilot	>	开始内联聊天	Ctrl+I
重命名符号	F2	对此进行解释	
更改所有匹配项	Ctrl+F2	修复此	
格式化文档	Shift+Alt+F	生成文档	
使用...格式化文档		生成测试	

同时，在生成测试用例后，可以自动创建一个单独的测试文件，以独立地测试函数

以下为对冒泡排序的函数生成的测试文件test_test2.cpp的部分内容，**对于每个测试用例，都用注释解释其具体用途**：

```
test > test.cpp test2.cpp test_test2.cpp x test.py
test > test_test2.cpp > bubbleSort(int [], int)
3 void bubbleSort(int arr[], int n) {
12
13     }
14 }
15 }
16
17 int main() {
18     // Test case 1: Sorted array
19     int arr1[] = {1, 2, 3, 4, 5};
20     int n1 = sizeof(arr1) / sizeof(arr1[0]);
21     bubbleSort(arr1, n1);
22     std::cout << "Sorted array 1: ";
23     for (int i = 0; i < n1; i++) {
24         std::cout << arr1[i] << " ";
25     }
26     std::cout << std::endl;
27
28     // Test case 2: Reverse sorted array
29     int arr2[] = {5, 4, 3, 2, 1};
30     int n2 = sizeof(arr2) / sizeof(arr2[0]);
31     bubbleSort(arr2, n2);
32     std::cout << "Sorted array 2: ";
33     for (int i = 0; i < n2; i++) {
34         std::cout << arr2[i] << " ";
35     }
36     std::cout << std::endl;
37
38     // Test case 3: Random array
39     int arr3[] = {3, 1, 4, 2, 5};
40     int n3 = sizeof(arr3) / sizeof(arr3[0]);
41     bubbleSort(arr3, n3);
42     std::cout << "Sorted array 3: ";
43     for (int i = 0; i < n3; i++) {
44         std::cout << arr3[i] << " ";
45     }
46     std::cout << std::endl;
47
48     return 0;
49 }
```

不同IDE下的使用

Pycharm

在第一次使用Copilot时，会弹出提示框指导使用

Partially accepting suggestions

Press **Ctrl+向右箭头** to insert only the first word of the suggestions. To accept the first line of a multi-line suggestion, press **Ctrl+Alt+向右箭头**.

[Configure keyboard shortcuts](#)

Got It

Pycharm右键的Github Copilot选项中多出一个Simplify选项，可对选中代码进行简化

Generate Docs

Explain This

Fix This

Simplify This

Generate Tests

Reference File in Chat

以下为生成的原始C++冒泡排序代码

```
//请为我生成一段代码，实现冒泡排序
#include <iostream>
#include <vector>

void bubbleSort(std::vector<int>& arr) {
    int n = arr.size();
    for (int i = 0; i < n - 1; i++) {
        for (int j = 0; j < n - i - 1; j++) {
            if (arr[j] > arr[j + 1]) {
                // 交换 arr[j] 和 arr[j+1]
                int temp = arr[j];
                arr[j] = arr[j + 1];
                arr[j + 1] = temp;
            }
        }
    }
}

int main() {
    std::vector<int> arr = {64, 34, 25, 12, 22, 11, 90};
    std::cout << "原始数组: ";
    for (int i = 0; i < arr.size(); i++) {
        std::cout << arr[i] << " ";
    }
    std::cout << std::endl;

    bubbleSort(arr);
}
```

```

std::cout << "排序后的数组: ";
for (int i = 0; i < arr.size(); i++) {
    std::cout << arr[i] << " ";
}
std::cout << std::endl;

return 0;
}

```

以下为经过简化的代码

```

#include <iostream>
#include <vector>
#include <algorithm>

void bubbleSort(std::vector<int>& arr) {
    for (int i = 0; i < arr.size(); i++) {
        for (int j = 0; j < arr.size() - i - 1; j++) {
            if (arr[j] > arr[j + 1]) {
                std::swap(arr[j], arr[j + 1]);
            }
        }
    }
}

void printArray(const std::vector<int>& arr) {
    for (const auto& num : arr) {
        std::cout << num << " ";
    }
    std::cout << std::endl;
}

int main() {
    std::vector<int> arr = {64, 34, 25, 12, 22, 11, 90};
    std::cout << "Original array: ";
    printArray(arr);

    bubbleSort(arr);

    std::cout << "Sorted array: ";
    printArray(arr);

    return 0;
}

```

同时给出了简化的过程

Simplifications made:

- 1.Replaced the manual swapping of array elements with the `std::swap` function from the `<algorithm>` library. This makes the code cleaner and easier to read.
- 2.Created a separate function `printArray` to print the array elements. This reduces code duplication and improves readability.
- 3.Removed the explicit declaration of the variable `n` as it was only used once. Instead, directly used `arr.size()` in the loop condition.
- 4.Used `const auto&` in the range-based for loop inside the `printArray` function. This makes the code more concise and easier to read.

总结与评价

优点

- 功能齐全且完备
- 基于Github庞大的代码量以及专业的技术团队、技术水平较高的用户群体，AI性能强大
- 代码生成质量与可读性相对较高，生成速度较快

缺点

- 需要付费使用，教育版可以免费使用，但申请流程繁琐且申请通过率较低
- 对国内用户较不友好，有一定使用门槛

目标用户和需求分析

核心目标用户：有一定技术水平的编程人员，主要市场面向国外

潜在目标用户：中国国内的编程人员以及广大高校学生

额外功能：代码安全性检测。在AWS推出的智能编程助手CodeWhisperer中，已经实现了对代码安全性的主动扫描，同时，随着网络安全的重要性显现，代码开发过程中的安全性愈发重要。且同类产品的相关功能同样还未完善，应抢先进行开发

改进建议

- 1.相对其他同类产品质量较高，但有一定的使用门槛。可在提高产品功能的同时，对价格做出一定优惠，同时积极开辟中国市场。
- 2.积极开发代码安全性检测等新功能，填补技术空缺

AWS CodeWhisperer

产品概述

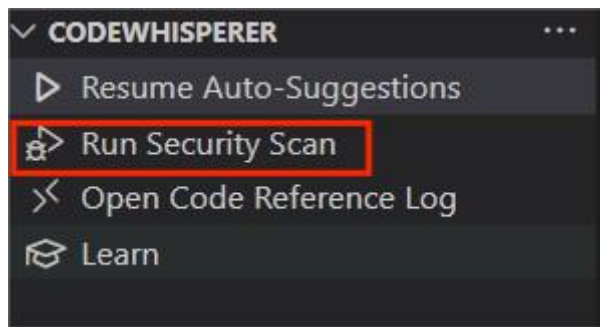
Amazon CodeWhisperer 是一种采用机器学习（ML）的服务，可以根据开发人员用自然语言编写的注释和集成式开发环境（IDE）中的代码生成代码建议，帮助开发人员提高工作效率，可以为您的应用程序提供代码审查、安全扫描和性能优化。

Amazon CodeWhisperer 为多种编程语言提供基于人工智能（AI）的代码建议，包括 Python、Java、JavaScript、TypeScript、C#、Go、Rust、PHP、Ruby、Kotlin、C、C++、Shell 脚本、SQL 和 Scala。您可以使用来自多个 IDE 的服务，包括 JetBrains IDE（IntelliJ IDEA、PyCharm、WebStorm 和 Rider）、Visual Studio（VS）Code、AWS Cloud9 和 AWS Lambda 控制台。

使用 CodeWhisperer，您可以扫描 Java、JavaScript 和 Python 项目以检测难以发现的漏洞，例如开放全球应用程序安全项目（OWASP）中排名前十的漏洞，或者不符合加密库最佳实践及其他类似安全最佳实践的漏洞。该服务分析 IDE 中的现有代码（无论是由 CodeWhisperer 生成还是由您编写），高度精确地识别有问题的代码，并对如何修复代码提供明智的建议。

独特点：主动进行安全扫描

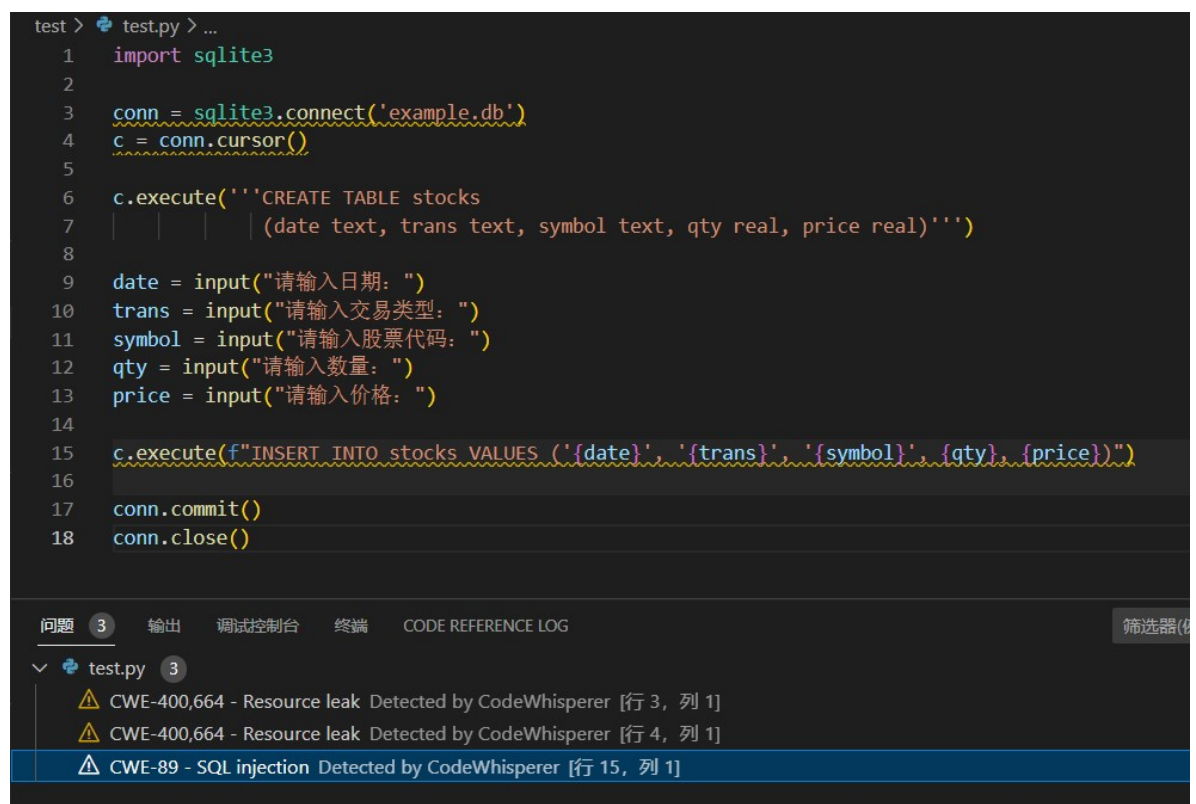
可以快速扫描 **Java**、**JavaScript** 和 **Python** 项目中难以发现的漏洞



系统命令注入漏洞



SQL注入漏洞



- 此外通过查询，还可对MD5碰撞漏洞，反序列化漏洞等进行检测

代码生成

利用注释生成：

```
1 //请为我生成一段代码，实现冒泡排序
2
3 #include <iostream>
4 using namespace std;
5 int main() {
6     int arr[] = {64, 34, 25, 12, 22, 11, 90};
7     int n = sizeof(arr) / sizeof(arr[0]);
8     for (int i = 0; i < n - 1; i++) {
9         for (int j = 0; j < n - i - 1; j++) {
10             if (arr[j] > arr[j + 1]) {
11                 int temp = arr[j];
12                 arr[j] = arr[j + 1];
13                 arr[j + 1] = temp;
14             }
15         }
16     }
17     cout << "Sorted array: \n";
18     for (int i = 0; i < n; i++) {
19         cout << arr[i] << " ";
20         cout << endl;
21     }
22     return 0;
23 }
```

- 不会主动生成main函数进行测试

```

1 //请为我生成一段代码，实现冒泡排序
2 #include <iostream>
3 using namespace std;
4 void bubble_sort(int *a,int n){
5     for(int i=0;i<n-1;i++){
6         for(int j=0;j<n-1-i;j++){
7             if(a[j]>a[j+1]){
8                 int temp=a[j];
9                 a[j]=a[j+1];
10                a[j+1]=temp;
11            }
12        }
13    }
14 }
15 //请在main函数中测试以上函数
16 int main(){
17     int a[5]={3,2,5,1,4};
18     bubble_sort(a,5);
19     for(int i=0;i<5;i++){
20         cout<<a[i]<<" ";
21     }
22     return 0;
23 }
24 }

```

- 在初始情况main函数大括号已闭合的情况下，多生成了一个大括号

英文注释：

```

test > G+ test2.cpp > bubbleSort(vector<int>&)
1 //Please generate a code snippet for me to implement bubble sort.
2 #include <iostream>
3 #include <vector>
4 using namespace std;
5 void bubbleSort(vector<int>& arr) {
6     int n = arr.size();
7     for (int i = 0; i < n-1; i++) {
8         for (int j = 0; j < n-i-1; j++) {
9             if (arr[j] > arr[j+1]) {
10                 swap(arr[j], arr[j+1]);
11             }
12         }
13     }
14 }
15 int main() {
16     vector<int> arr = {5, 2, 4, 6, 1, 3};
17     bubbleSort(arr);
18     for (int i = 0; i < arr.size(); i++) {
19         cout << arr[i] << " ";
20     }
21     return 0;
22 }
23 //Please generate a code snippet for me to implement selection sort.

```

- 一次性生成的代码相对中文注释更加完整
- 但在最后重复生成了一遍原有注释

代码补全：


```

1  #include <iostream>
2  #include <vector>
3  using namespace std;
4  void bubbleSort(vector<int>& arr) {
5      int n = arr.size();
6      for (int i = 0; i < n-1; i++) {
7          for (int j = 0; j < n-i-1; j++) {
8              if (arr[j] > arr[j+1]) {
                  swap(arr[j], arr[j+1]);
              }
              cout << "i = " << i << ", j = " << j << endl;
              for (int k = 0; k < arr.size(); k++) {
                  cout << arr[k] << " ";
              }
              cout << endl;
              cout << "=====" << endl;
              cout << endl;
              cout << endl;
9          }
10     }
11 }
12 int main() {
13     vector<int> arr = {5, 2, 4, 6, 1, 3};
14     bubbleSort(arr);
15     for (int i = 0; i < arr.size(); i++) {
16         cout << arr[i] << " ";
17     }
18     return 0;
19 }

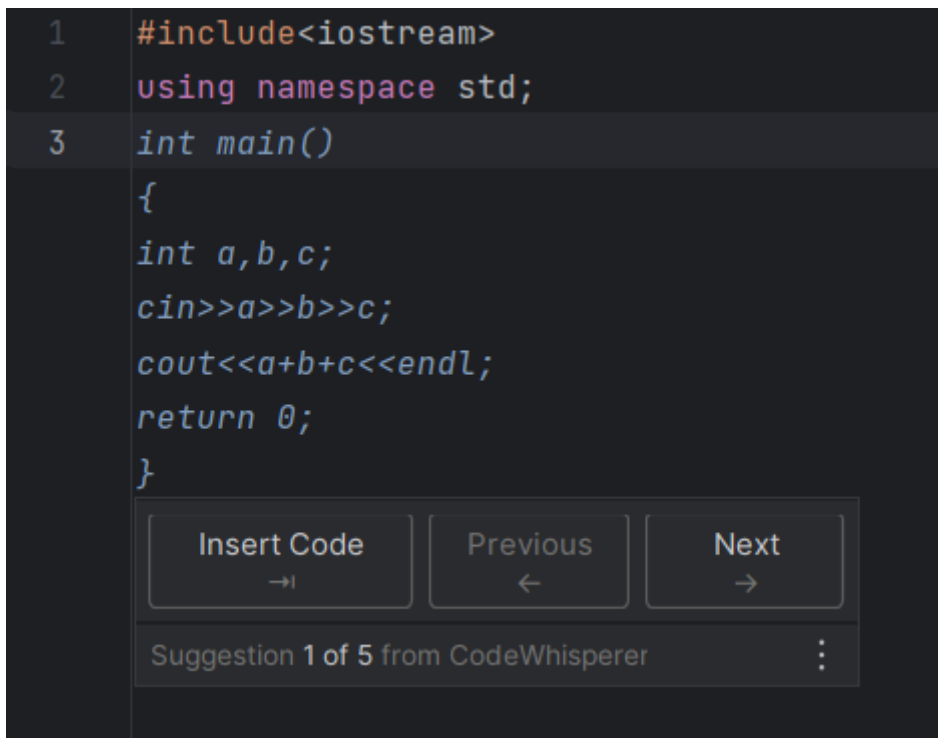
```

- 代码根据上下文补全了冒泡排序函数，同时对每一个循环的顺序进行了输出，使排序过程更加直观

不同IDE下的使用

Pycharm

UI更加直接美观



总结与评价

优点

- 独特而强大的功能：代码安全性的主动检查
- 免费使用
- Amazon拥有庞大的数据量，AI功能较为强大
- 代码生成质量和效率较高

缺点

- 代码安全性检查耗费时间较多
- 代码生成时的注释较少，可读性较低
- 有时会出现代码结构上的错误，如大括号的闭合问题等

目标用户和需求分析

核心目标用户：对代码安全性有需求，且需要工具辅助进行检测的开发人员。对免费智能编程助手有需求的编程人员

潜在目标用户：对代码安全性有需求，且并未了解智能编程助手的开发人员

额外功能：代码测试功能。其他类似产品已实现相关功能，且该功能较为实用，应加装该功能以完善产品

改进建议

- 1.加装代码测试功能以完善产品
- 2.继续完善代码安全性检测功能，巩固该功能在类似产品中的领先地位

Tsinghua CodeGeeX

软件概述

CodeGeeX2 是多语言代码生成模型 [CodeGeeX \(KDD'23\)](#) 的第二代模型。不同于一代 CodeGeeX (完全在国产华为昇腾芯片平台训练)，CodeGeeX2 是基于 [ChatGLM2](#) 架构加入代码预训练实现，得益于 ChatGLM2 的更优性能，CodeGeeX2 在多项指标上取得性能提升 (+107% > CodeGeeX; 仅60亿参数即超过150亿参数的 StarCoder-15B 近10%)，更多特性包括：

- 更强大的代码能力：基于 ChatGLM2-6B 基座语言模型，CodeGeeX2-6B 进一步经过了 600B 代码数据预训练，相比一代模型，在代码能力上全面提升，[HumanEval-X](#) 评测集的六种编程语言均大幅提升 (Python +57%, C++ +71%, Java +54%, JavaScript +83%, Go +56%, Rust +321%)，在 Python 上达到 35.9% 的 Pass@1 一次通过率，超越规模更大的 StarCoder-15B。
- 更优秀的模型特性：继承 ChatGLM2-6B 模型特性，CodeGeeX2-6B 更好支持中英文输入，支持最大 8192 序列长度，推理速度较一代 CodeGeeX-13B 大幅提升，量化后仅需6GB显存即可运行，支持轻量级本地化部署。
- 更全面的AI编程助手：CodeGeeX插件 ([VS Code](#), [Jetbrains](#)) 后端升级，支持超过100种编程语言，新增上下文补全、跨文件补全等实用功能。结合 Ask CodeGeeX 交互式AI编程助手，支持中英文对话解决各种编程问题，包括且不限于代码解释、代码翻译、代码纠错、文档生成等，帮助程序员更高效开发。
- 更开放的协议：CodeGeeX2-6B 权重对学术研究完全开放，填写[登记表](#)申请商业使用。

代码生成

利用注释生成

中文注释：

```
1 //请为我生成一段代码，实现冒泡排序
2
3 #include <iostream>
4 using namespace std;
5 int main() {
6     int arr[] = {64, 34, 25, 12, 22, 11, 90};
7     int n = sizeof(arr) / sizeof(arr[0]);
8     for (int i = 0; i < n - 1; i++) {
9         for (int j = 0; j < n - i - 1; j++) {
10             if (arr[j] > arr[j + 1]) {
11                 int temp = arr[j];
12                 arr[j] = arr[j + 1];
13                 arr[j + 1] = temp;
14             }
15         }
16     }
17     cout << "Sorted array: \n";
18     for (int i = 0; i < n; i++) {
19         cout << arr[i] << " ";
20         cout << endl;
21     }
22     return 0;
23 }
```

- 生成过程中没有注释
- 最后大括号没有对齐
- 生成速度较慢,不过可能是因为设置了生成延迟0.5秒,以避免不必要的生成
- 在生成时,常出现单独生成一个大括号以进行配对的情况.没有在一开始便将大括号闭合
- 主体代码全部在main函数中,较为杂乱

英文注释:

英文注释也可正常生成, 代码生成速度和结构与中文注释相似

```
1 //Please generate a code snippet for me to implement bubble sort.
2 #include <iostream>
3 using namespace std;
4
5 void bubbleSort(int arr[], int n) {
6     for (int i = 0; i < n-1; i++) {
7         for (int j = 0; j < n-i-1; j++) {
8             if (arr[j] > arr[j+1]) {
9                 // swap arr[j] and arr[j+1]
10                int temp = arr[j];
11                arr[j] = arr[j+1];
12                arr[j+1] = temp;
13            }
14        }
15    }
16 }
17
18 int main() {
19     int arr[] = {64, 34, 25, 12, 22, 11, 90};
20     int n = sizeof(arr)/sizeof(arr[0]);
21     bubbleSort(arr, n);
22     cout << "Sorted array: \n";
23     for (int i=0; i < n; i++) {
24         cout << arr[i] << " ";
25     }
26     return 0;
27 }
```

代码补全

```

1  #include <iostream>
2  using namespace std;
3  void bubbleSort(int arr[], int n) {
4      for (int i = 0; i < n-1; i++) {
5          for (int j = 0; j < n-i-1; j++) {
6              if (arr[j] > arr[j+1]) {
7                  // 交换 arr[j] 和 arr[j+1]
7                  int temp = arr[j];
7                  arr[j] = arr[j+1];
7                  arr[j+1] = temp;
8              }
9          }
10     }
11 }
12 int main() {
13     int arr[] = {64, 34, 25, 12, 22, 11, 90};
14     int n = sizeof(arr)/sizeof(arr[0]);
15     bubbleSort(arr, n);
16     cout << "排序后的数组: ";
17     for (int i = 0; i < n; i++) {
18         cout << arr[i] << " ";
19     }
20     return 0;
21 }

```

注释生成

CodeGeeX	>	向CodeGeeX提问
转到定义	F12	解释代码
转到声明		生成注释
转到类型定义		生成单测
转到引用	Shift+F12	

```

test > test.cpp > ...
1 //请为我生成一段代码，实现冒泡排序
2
3 #include <iostream>
4 using namespace std;
5
6 void bubbleSort(int arr[], int n) {
7     // 冒泡排序
8     for (int i = 0; i < n-1; i++) {
9         for (int j = 0; j < n-i-1; j++) {
10             if (arr[j] > arr[j+1]) {
11                 // 交换 arr[j] 和 arr[j+1]
12                 int temp = arr[j];
13                 arr[j] = arr[j+1];
14                 arr[j+1] = temp;
15             }
16         }
17     }
18 }
19
20 int main() {
21     // 定义一个数组
22     int arr[] = {64, 34, 25, 12, 22, 11, 90};
23     // 获取数组的长度
24     int n = sizeof(arr)/sizeof(arr[0]);
25     // 调用冒泡排序函数
26     bubbleSort(arr, n);
27     // 打印排序后的数组
28     cout << "Sorted array: \n";
29     for (int i=0; i < n; i++) {
30         cout << arr[i] << " ";
31     }
32     return 0;
33 }

```

测试用例生成

对于特定功能的函数，可对其生成多个测试用例来测试函数的正确性

如对于冒泡排序的函数生成测试用例：

CodeGeeX	>	向CodeGeeX提问
转到定义	F12	解释代码
转到声明		生成注释
转到类型定义		生成单测

// 测试用例

```
int main() {  
    int arr1[] = {64, 34, 25, 12, 22, 11, 90};  
    int n1 = sizeof(arr1)/sizeof(arr1[0]);  
    bubbleSort(arr1, n1);  
    for (int i = 0; i < n1; i++)  
        std::cout << arr1[i] << " ";  
    std::cout << std::endl;  
  
    int arr2[] = {64, 34, 25, 12, 22, 11, 90, 5};  
    int n2 = sizeof(arr2)/sizeof(arr2[0]);  
    bubbleSort(arr2, n2);  
    for (int i = 0; i < n2; i++)  
        std::cout << arr2[i] << " ";  
    std::cout << std::endl;  
  
    int arr3[] = {1, 2, 3, 4, 5};  
    int n3 = sizeof(arr3)/sizeof(arr3[0]);  
    bubbleSort(arr3, n3);  
    for (int i = 0; i < n3; i++)  
        std::cout << arr3[i] << " ";  
    std::cout << std::endl;  
  
    int arr4[] = {5, 4, 3, 2, 1};  
    int n4 = sizeof(arr4)/sizeof(arr4[0]);  
    bubbleSort(arr4, n4);  
}
```

代码解释

对以上冒泡排序代码的解释



CodeGeeX

这段C++代码实现了一个冒泡排序算法，用于对整数数组进行排序。

1. `#include <iostream>`: 引入*iostream*头文件，用于实现控制台输入输出。
2. `using namespace std;`: 使用*std*命名空间，避免在使用标准库中的函数时需要加上*std::*。
3. `void bubbleSort(int arr[], int n) {...}`: 定义一个名为*bubbleSort*的函数，接受一个整数数组*arr*和数组长度*n*作为参数。
4. `for (int i = 0; i < n-1; i++) {...}`: 使用两层循环，外层循环遍历数组中的每个元素，内层循环负责比较相邻的元素。
5. `for (int j = 0; j < n-i-1; j++) {...}`: 内层循环遍历数组中未排序的部分，从第一个元素开始，到倒数第*i*+1个元素结束。
6. `if (arr[j] > arr[j+1]) {...}`: 如果当前元素大于下一个元素，则交换它们的位置。
7. `int temp = arr[j];`: 创建一个临时变量*temp*，用于存储当前元素。
8. `arr[j] = arr[j+1];`: 将下一个元素赋值给当前元素。
9. `arr[j+1] = temp;`: 将临时变量*temp*的值赋给下一个元素。
10. `bubbleSort(arr, n);`: 在*main*函数中调用*bubbleSort*函数，对数组*arr*进行排序。
11. `cout << "Sorted array: \n";`: 输出排序后的数组。
12. `for (int i=0; i < n; i++) {...}`: 使用*for*循环遍历排序后的数组，并输出每个元素。
13. `return 0;`: 返回0，表示程序正常结束。

冒泡排序是一种简单的排序算法，它重复地遍历要排序的数组，比较相邻的元素，如果顺序错误就交换它们的位置。遍历数组的过程是重复地进行内层循环，直到没有需要交换的元素为止。

输入你的问题，或用 '/' 调用快捷命令

CodeGeeX Pro



/

@repo



代码翻译

CodeGeeX宣称可将代码从语义级进行翻译，从而将代码翻译成不同的编程语言

以下代码为将之前C++语言编写的冒泡排序算法翻译为Java语言后的结果：

```
public class BubbleSort {
```

```

public static void main(String[] args) {
    int[] arr1 = {64, 34, 25, 12, 22, 11, 90};
    bubbleSort(arr1);
    for (int i = 0; i < arr1.length; i++)
        System.out.print(arr1[i] + " ");
    System.out.println();

    int[] arr2 = {64, 34, 25, 12, 22, 11, 90, 5};
    bubbleSort(arr2);
    for (int i = 0; i < arr2.length; i++)
        System.out.print(arr2[i] + " ");
    System.out.println();

    int[] arr3 = {1, 2, 3, 4, 5};
    bubbleSort(arr3);
    for (int i = 0; i < arr3.length; i++)
        System.out.print(arr3[i] + " ");
    System.out.println();

    int[] arr4 = {5, 4, 3, 2, 1};
    bubbleSort(arr4);
    for (int i = 0; i < arr4.length; i++)
        System.out.print(arr4[i] + " ");
    System.out.println();

    int[] arr5 = {1};
    bubbleSort(arr5);
    for (int i = 0; i < arr5.length; i++)
        System.out.print(arr5[i] + " ");
    System.out.println();
}

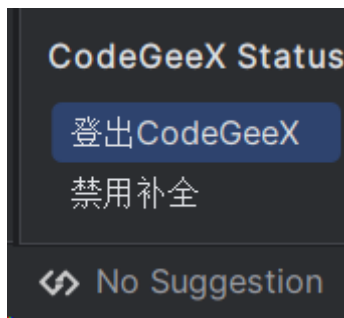
public static void bubbleSort(int[] arr) {
    int n = arr.length;
    for (int i = 0; i < n - 1; i++) {
        for (int j = 0; j < n - i - 1; j++) {
            if (arr[j] > arr[j + 1]) {
                // swap arr[j] and arr[j+1]
                int temp = arr[j];
                arr[j] = arr[j + 1];
                arr[j + 1] = temp;
            }
        }
    }
}
}

```

不同IDE下的使用

Pycharm

在Pycharm中可以方便的禁用自动补全，而不用像VS Code中只能通过禁用或卸载插件来禁用补全



总结与评价

优点

- 由国内团队开发，产品信息及后续更新更易获取
- 可在语义层次对代码进行翻译，可方便快捷地将代码翻译为多种代码语言
- 有注释生成功能，可对代码进行注释，提高代码的可读性

缺点

- 生成代码速度较慢，效率较低
- 产品功能相对较少，功能较不完备

目标用户和需求分析

核心目标用户：支持国内相关产品开发的开发测试人员

潜在目标用户：国内广大编程人员

额外功能：代码修复功能。其他类似产品具备该功能且功能较为完善，应尽快补齐短板，使产品质量得到提升。

改进建议

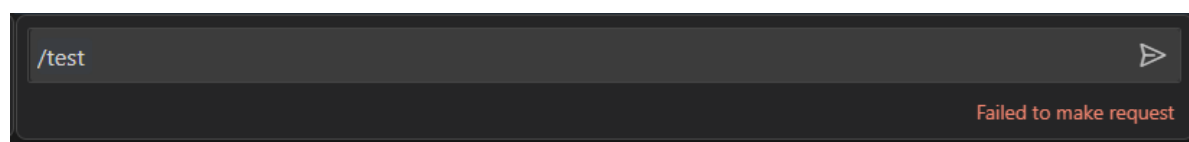
- 1.增加产品功能，补齐短板
- 2.提高代码生成质量和效率，打磨关键功能

开发新的产品

目前缺少功能齐全，同时能够主动进行代码安全性扫描的产品。希望能够开发功能综合而全面的产品，包括代码补全、注释生成、代码修复、代码解释、测试样例生成、生成文档以及代码安全性检查功能。训练健壮的AI，提高代码生成的质量和效率，同时注重代码安全性，提供高效而全面的代码安全性主动扫描功能。

实验过程中的问题

由于未更新vscode，Github Copilot Chat在一开始无法使用



将VsCode更新到最新版本后可以正常使用

