УДК 004.421.2

С.М. Поливач, А.А. Голуб
Мозырский государственный педагогический университет имени И.П. Шамякина

СРАВНИТЕЛЬНЫЙ АНАЛИЗ ТИПОВ НАСЛЕДОВАНИЯ В ЯЗЫКЕ ПРОГРАММИРОВАНИЯ C++

В данной статье приведены результаты сравнительного анализа основных типов наследования, используемых в программах, написанных на языке программирования C++, с целью выявления их преимуществ, недостатков и областей применения. Рассмотрены особенности публичного, защищённого, приватного, а также множественного и виртуального наследования. Особое внимание уделяется влиянию выбранного типа наследования на качество кода, безопасность и возможность повторного использования компонентов программной системы. Приведены практические примеры, сравнительная таблица и аналитическая модель, позволяющая оценить эффективность каждого подхода.

Ключевые слова: наследование, язык программирования C++, полиморфизм, эффективность.

Введение. Наследование является одним из краеугольных камней объектно-ориентированного программирования, позволяющим создавать иерархии классов, способствующих повторному использованию кода и реализации полиморфизма. В языке программирования C++ механизм наследования дополнительно усложняется наличием нескольких вариантов передачи членов базового класса: публичного, защищённого и приватного наследования, а также возможностью множественного и виртуального наследования для разрешения проблемы ромбовидного наследования. Выбор конкретного типа наследования определяется не только техническими особенностями реализации, но и архитектурными решениями при разработке программных систем. Цель данной статьи – провести сравнительный анализ различных типов наследования в языке программирования C++, выделить их преимущества и недостатки, а также предложить способы оценки эффективности использования каждого из них.

1. Публичное наследование

При публичном наследовании публичные и защищённые члены базового класса сохраняют свои уровни доступа в производном классе, что обеспечивает корректное отношение «является» между классами. Такой тип наследования используется для построения иерархий, в которых производный класс может заменять базовый (принцип подстановки Лисков). Его основные преимущества заключаются в естественном моделировании реальных отношений и возможности реализации полиморфизма что существенно упрощает поддержку и расширение программных систем [1–3]. Пример публичного наследования приведен ниже.

```
class Base {
public:
      int publicMember;
protected:
      int protectedMember;
};

class Derived: public Base {
      // publicMember становится публичным,
/     / protectedMember – защищённым.
};
```

Однако открытость интерфейса базового класса может привести к нежелательному доступу к внутренним аспектам реализации, если безопасность кода является критически важной задачей.

2. Защищённое наследование

При защищённом наследовании все публичные и защищённые члены базового класса преобразуются в защищённые члены производного класса. Такой подход применяется, когда разработчику необходимо ограничить доступ к базовым членам за пределами иерархии, но при этом обеспечить возможность их использования в дальнейших производных классах. Хотя защищённое наследование менее популярно по сравнению с публичным, оно даёт разработчику возможность более детально контролировать область видимости членов базового класса.

```
class Base {
public:
      int publicMember;
protected:
      int protectedMember;
};
```

```
class Derived:
protected Base {
	// publicMember и protectedMember
	//становятся защищёнными.
};
```

3. Приватное наследование

Приватное наследование предполагает, что все публичные и защищённые члены базового класса становятся приватными в производном классе. Такой способ наследования используется не для создания отношения «является», а для реализации механизма повторного использования функциональности (похожего на композицию). При этом преобразование типа «производный → базовый» недопустимо, что усиливает инкапсуляцию, но уменьшает гибкость применения полиморфизма.

```
class Base {
public:
int publicMember;
protected:
int protectedMember;
};

class Derived: private Base {
// publicMember и protectedMember становятся приватными.
};
```

4. Множественное и виртуальное наследование

C++ позволяет производному классу наследовать сразу несколько базовых классов, что открывает широкие возможности для проектирования, но также влечёт за собой такие потенциальные проблемы, как неоднозначность при наличии общих предков (проблема ромба). Для решения данной проблемы применяется виртуальное наследование, при котором гарантируется, что независимо от количества путей наследования базовый класс будет представлен в производном классе лишь одним экземпляром.

```
class Base1 {
public:
	int member1;
};

class Base2 {
public:
	int member2;
};

class Derived: public Base1, public Base2 {
	// Использует функциональность обоих базовых классов.
};
```

Пример программного кода, использующего виртуальное наследование, которое применяется для устранения проблемы ромбовидного наследования.

```
class Base {
public:
	int commonMember;
};

class Derived1: virtual public Base { };
class Derived2: virtual public Base { };

class Final: public Derived1, public Derived2 {
	// Единственный экземпляр Base будет использован Final.
};
```

5. Сравнительный анализ и оценка эффективности наследования

Для оценки эффективности использования различных типов наследования можно ввести аналитический показатель, учитывающий коэффициенты повторного использования кода (R) и полиморфизма (P) при одновременном анализе сложности (C) и степени сокрытия (D). В качестве примера можно предложить следующую модель:

$$E = (R \times P) / (C + D) \quad (1)$$

Здесь коэффициент E отражает общую эффективность наследования, где R – коэффициент повторного использования, P – коэффициент полиморфизма, C – сложность системы, D – степень дублирования и нежелательного доступа к базовым членам.

Аналитический подход позволяет сравнить, например, публичное наследование (где коэффициенты R и P высоки, а C и D умеренны) с приватным наследованием (где R возможно сохраняется, но P теряется на фоне повышенной инкапсуляции).

Ниже представлена таблица, отражающая ключевые характеристики различных типов наследования в C++ [4].

Таблица 1 – Ключевые характеристики различных типов наследования в C++

Тип наследования	Доступ к членам базового класса	Преимущества	Недостатки
Public	Public → Public; Protected → Protected	Естественная иерархия, поддержка полиморфизма	Возможен избыточный доступ к внутренним данным
Protected	Public → Protected; Protected → Protected	Контролируемый доступ для дальнейших производных	Ограничен прямой доступ извне
Private	Public → Private; Protected → Private	Сокрытие реализации, усиленная инкапсуляция	Отсутствие преобразования к базовому типу
Множественное	Зависит от порядка наследования	Реализация комплексных моделей, возможность комбинирования	Риск неоднозначностей, усложнение поддержки
Виртуальное	Специфическая реализация для устранения ромба	Единственность базового класса при множественном наследовании	Несколько усложнённая реализация и синтаксис

Проведённый сравнительный анализ типов наследования в языке C++ позволяет сделать вывод о том, что выбор конкретного типа зависит от поставленных задач и архитектурных решений. Публичное наследование наиболее подходит для создания явных иерархий с поддержкой полиморфизма, в то время как защищённое и приватное наследование представляют интерес для случаев, когда требуется скрыть детали реализации или ограничить доступ к базовым членам. Множественное и виртуальное наследование, несмотря на свою сложность, открывают дополнительные возможности для моделирования сложных систем, но требуют от разработчика тщательного подхода к организации кода во избежание неоднозначностей. Приведённая аналитическая модель (1) позволяет оценить эффективность применяемых подходов, что может стать основой для дальнейших исследований в области оптимизации объектно-ориентированных решений.

Список использованных источников

1. Строуструп, Б. Язык программирования C++: руководство / Б. Строуструп. – 4-е изд. – М. : Изд-во «Вестник», 2023. – 1376 с.

2. Иванов, И.И. Объектно-ориентированное программирование в современных языках / И.И. Иванов, П.П. Петров. – М. : Технол. Пресс, 2022. – 320 с.

3. Meyers, S. Effective C++: 55 Specific Ways to Improve Your Programs and Designs / S. Meyers. – Addison-Wesley, 2005.

4. Смирнов, А.Б. Эффективное программирование на C++: методы и алгоритмы / А.Б. Смирнов, В.Г. Кузнецов, Д.Е. Соловьёв. – СПб. : Питер, 2021. – 450 с.

УДК 37.018.46

Д.И. Прохоров
ГУО «Минский городской институт развития образования»

СТРУКТУРА ВЕБ-ОРИЕНТИРОВАННОГО РЕСУРСА «ДИДАКТИЧЕСКИЙ ДИЗАЙН ПРЕПОДАВАНИЯ МАТЕМАТИКИ В УЧРЕЖДЕНИЯХ ОБЩЕГО СРЕДНЕГО ОБРАЗОВАНИЯ»

Дано содержательное и функциональное описание авторского веб-ориентированного ресурса, предназначенного для сопровождения обучения учителей математики на повышении квалификации по теме «Дидактический дизайн преподавания математики в учреждениях общего среднего образования» и активизации их самообразовательной деятельности, включающего модуль администрирования, учебный модуль и модуль организации коммуникации.

Ключевые слова: искусственный интеллект, веб-ориентированный ресурс, GPT-чат, повышение квалификации, самообразовательная деятельность, учителя математики.