физических систем и явлений. Клеточные автоматы позволяют моделировать эмерджентное поведение, когда сложное поведение системы возникает из простых правил взаимодействия между ее составляющими частями. Это открывает новые возможности для изучения самоорганизации, формирования структур, динамики процессов и других аспектов физических явлений. Их использование способствует лучшему пониманию сложных систем и помогает создавать новые методы анализа и прогнозирования поведения природных и искусственных объектов.

Список использованной литературы

1. Mathworld [Electronic resource] // Представление клеточных автоматов – Mode of access: https://mathworld.wolfram.com/ CellularAutomaton.html – Date of access: 12.05.2024.

2. Researchgate [Electronic resource] // Понятие о устойчивых структурах – Mode of access: https://www.researchgate.net/figure/Illustration-of-the-cellular-automata-model-Original-figure-was-retrieved-from_fig3_220136467 – Date of access: 12.05.2024.

ОСОБЕННОСТИ ИСПОЛЬЗОВАНИЯ ВИРТУАЛЬНЫХ МЕТОДОВ И АБСТРАКТНЫХ КЛАССОВ В ЯЗЫКЕ ПРОГРАММИРОВАНИЯ C++

Поливач Сергей (УО МГПУ им. И.П. Шамякина, г. Мозырь)
Научный руководитель – А.А. Голуб, канд. физ.-мат. наук, доцент

Объектно-ориентированное программирование в C++ характеризуется наличием ряда специфических особенностей, отличающих его от реализации этой парадигмы в таких языках программирования, как Java или C#. Например, используются виртуальные методы и абстрактные классы, которые являются одним из центральных механизмов, позволяющих реализовать гибкую работу с полиморфными объектами.

В C++ механизм виртуальных функций базируется на использовании специальной таблицы методов (vtable). Каждый объект класса, содержащего хотя бы один виртуальный метод, внутри себя хранит скрытый указатель на эту таблицу. Именно этот механизм обеспечивает динамический выбор реализации метода в момент выполнения программы [1].

Рассмотрим простейший пример реализации виртуального метода [2].

```
class Base {
public:
    virtual void show() {
      cout << "Base";
    }
};
class Derived : public Base {
public:
    void show() override {
      cout << "Derived";
   }
};
```

Здесь вызов метода show() через указатель на объект базового класса автоматически приводит к выполнению версии, определённой в классе-наследнике. Главное отличие от Java заключается в том, что в C++ программист обязан явно помечать метод ключевым словом virtual.

Абстрактные классы в C++ реализуются посредством объявления виртуальных функций (например, virtual void f() = 0;). Объекты таких классов нельзя непосредственно создавать; абстрактные классы служат в качестве основы для подпредставления интерфейса. Ниже приведен пример абстрактного класса.

```
class Abstract {
public:
    virtual void calculate() = 0;
};
```

В отличие от C#, где для определения интерфейсов используется ключевое слово interface, абстрактные классы в C++ могут совмещать как виртуальные методы, так и обычные, реализуя часть функциональности объектов, позволяя тем самым создавать более гибкие конструкции.

При разработке программ, использующих абстрактные классы, важно также уделять внимание корректному уничтожению объектов. Если базовый класс объявлен с виртуальными методами, его деструктор обязательно должен быть виртуальным, чтобы при удалении объекта через указатель на базовый класс гарантировалось корректное освобождение ресурсов. Например:

```
class Base {
public:
    virtual ~Base() {}
};
```

Такой подход особенно критичен в C++, где отсутствует автоматическая система сборки «мусора», и неправильное управление памятью может приводить к нестабильной работе программ.

Рассмотрим решение проблемы «ромбовидного наследования» в C++. Эта ситуация возникает, когда два класса наследуют одного общего базового класса. В C++ для её устранения применяется механизм виртуального наследования:

```
class A {};
class B : virtual public A {};
class C : virtual public A {};
class D : public B, public C {};
```

Такой механизм отсутствует в языках, где множественное наследование полностью запрещено (например, в Java или C#).

Стоит отметить, что применение виртуальных методов ведёт к незначительному увеличению размера объекта (за счёт дополнительного указателя на vtable, обычно от 4 до 8 байт) и может слегка замедлять выполнение кода из-за косвенных вызовов методов. Однако эта небольшая потеря эффективности компенсируется возможностями полиморфизма,

позволяющими создавать более масштабируемые и гибкие архитектуры программ.

С учетом всего вышесказанного можно сказать, что использование абстрактных и виртуальных классов в C++ требует более тщательного и продуманного проектирования, однако предоставляет разработчику мощный инструмент для тонкой настройки работы с памятью и управления структурой программы.

Список использованной литературы

1. Virtual inheritance [Электронный ресурс] / Википедия. – Режим доступа: https://en.wikipedia.org/wiki/Virtual_inheritance. – Дата доступа: 12.03.2025.

2. Виртуальный базовый класс [Электронный ресурс] / Программирование на C и C++. – Режим доступа: https://www.c-cpp.ru/books/virtualnyy-bazovyy-klass. – Дата доступа: 12.03.2025.

РАЗРАБОТКА ДЕСКТОПНЫХ ПРИЛОЖЕНИЙ НА PYTHON

Полын Серафим (УО МГПУ им. И.П. Шамякина, г. Мозырь)
Научный руководитель – В.В. Давыдовская, канд. физ.-мат. наук, доцент

Разработка десктопных приложений остаётся актуальной задачей для многих разработчиков. Такие программы широко применяются в различных сферах – от корпоративного программного обеспечения до личных утилит. Несмотря на рост популярности веб- и мобильных приложений, десктопные решения обладают рядом преимуществ, таких как высокая производительность, офлайн-доступ и более глубокая интеграция с операционной системой [1].

Python является одним из наиболее удобных языков для создания настольных приложений благодаря своей простоте, читаемости кода и огромному количеству библиотек. Он позволяет разрабатывать как простые утилиты, так и сложные кроссплатформенные системы с графическим интерфейсом.

Существует несколько популярных инструментов для создания GUI-приложений на Python: PyQt, Tkinter, Kivy, wxPython, PySide.

PyQt – это набор привязок к фреймворку Qt, который позволяет разрабатывать кроссплатформенные приложения с современным интерфейсом. Он поддерживает Windows, macOS и Linux, а также обладает богатым функционалом для работы с виджетами, окнами, меню и событиями. Qt предоставляет широкий набор компонентов, включая кнопки, текстовые поля, таблицы, вкладки и графики, а также позволяет создавать сложные макеты и настраивать внешний вид приложения. PyQt поддерживает систему сигналов и слотов, что упрощает обработку событий и взаимодействие между элементами интерфейса. Кроме того, PyQt позволяет использовать OpenGL для работы с 3D-графикой и поддерживает мультимедийные возможности, включая воспроизведение видео и аудио [2].