Введение в объектно-ориентированное программирование

лекция №6

Программы

Работающие в консольном режиме (под управлением операционных систем, работающих в командном режиме – DOS)

- последовательные
- используют ввод/вывод на устройство CON для общения с пользователем
- технология разработки: структурное программирование

Работающие под управлением Windows (или любые оконные приложения)

- событийно-управляемые
- используют стандартный интерфейс пользователя (GUI – Graphical User Interface)
- технология разработки: объектно-ориентированное программирование

объектно-ориентированное программирование необходимо тогда, когда количество параметров подпрограмм велико (т реальный объект сложен, описывается большим количество параметров)

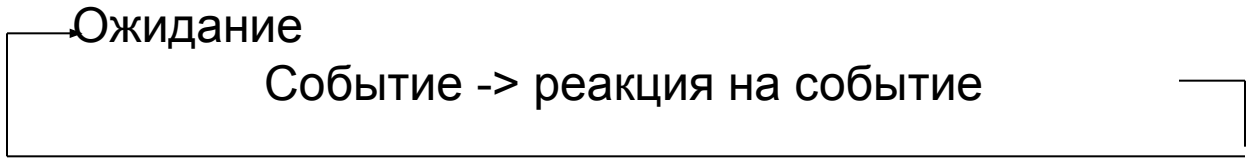
Последовательные программы

Схема работы последовательной программы:

Запуск программы -> выполнение инструкций -> завершение

Событийно-управляемые программы

Схема работы событийно-управляемой программы:



Примеры событий:

- щелчок (одинарный или двойной) кнопкой мыши (правой или левой) в произвольном месте экрана;
- выбор пункта меню (горизонтального, контекстного, и т. д.);
- нажатие клавиши (или комбинации клавиш);
- приход сигналов устройств компьютера;
- достижение встроенным таймером заданного момента времени.

Обработчик события - программный код, описывающий реакцию на событие.

Обработчик представляет собой подпрограмму, тело которой, как правило, пишется по законам последовательного программирования.

Программист может использовать стандартные обработчики или создавать собственные обработчики, отражающие специфику разрабатываемого приложения.

О структурном и объектно-ориентированном программировании

Структурное программирование подразумевает:

- точно обозначенные управляющие (базовые) структуры алгоритмов;
- соответствующее логике программы разбиение ее на программные блоки;
- автономные подпрограммы, в которых преимущественно используются локальные переменные;
- отсутствие (или, по крайней мере, ограниченное использование) операторов безусловного перехода – *goto, break* и др.

при создании средних и малых программ структурный подход дает хорошие результаты

Объектно-ориентированное программирование (ООП)

- **включает в себя лучшие идеи структурного программирования и дополняет их новыми мощными концепциями.**
- **ООП, так же как и структурное программирование, позволяет разложить задачу на подзадачи, но при этом каждая подзадача становится самостоятельным объектом, содержащим свои коды и данные.**

Практика показывает, что применение объектно-ориентированного подхода существенно уменьшает трудоемкость написания программ.

Понятие класса и объекта

- Класс = поля +методы
- Поля =данные
- Методы = подпрограммы, работающие с этими данными
- элементы класса = поля и методы
- Класс – тип
- Объект (или экземпляр класса) – переменная типа класс

Описание класса на языке Си++

Простейшее описание

```
class имя_класса {
        private:
            описание личных элементов класса
        public:
            описание общих элементов класса
    };
```

Личные элементы класса (private) - такие элементы, которые могут использоваться только методами своего класса.

Общие элементы класса (public) - к которым доступ разрешен в любом месте программы.

Принцип инкапсуляции: поля – личные элементы класса, методы – общие элементы класса (в основном). *Или:* поля класса могут использоваться только методами своего класса.

Описание полей – как обычное описание переменных.

Описание методов: внутри класса – обычно заголовок подпрограммы (в Си++ - функции), после описания класса – полное описание метода.

Поля класса – глобальные переменные по отношению к методам.

Определение методов класса

Если внутри класса метод описан заголовком:

***тип имя_метода* (*список_формальных_параметров*)**

то при полном описании метод имеет заголовок:

***тип имя_класса*::*имя_метода* (*список_форм_параметров*)**

т. е. в заголовке перед именем метода указывается имя класса (через двойное двоеточие).

Конструкторы и деструкторы

- Конструкторы и деструкторы – это специальные методы класса.
- Назначение конструктора: создание экземпляра класса и инициализация его полей.
- Назначение деструктора: уничтожение экземпляра класса. Деструктор может вызываться в программе явно или (что происходит обычно) его вызов обеспечивается компилятором в момент уничтожения экземпляра класса.
- Описываются конструкторы (деструкторы) как обычные функции, но для них используются стандартные имена. Имя конструктора совпадает с именем класса, имя деструктора:*~имя_класса.* Кроме того, у конструкторов и деструкторов не объявляется тип возвращаемого значения, у деструктора не может быть параметров.
- Наличие конструктора и деструктора для любого класса обязательно; при их отсутствии компилятор автоматически создает стандартные варианты конструктора и деструктора.

Объявление объекта (экземпляра класса)

Экземпляры класса могут создаваться **автоматически** и **динамически** (вспомните, что такое автоматические и динамические данные).

Автоматическое создание экземпляра класса осуществляется с помощью объявления:

***имя_класса имя_экземпляра*(*параметры конструктора*);**

Уничтожение автоматически созданных экземпляров классов происходит также автоматически при завершении выполнения блока функции, в котором они были определены.

Создание динамического экземпляра класса

1. Объявление указателя на экземпляр класса:
   имя_класса* указатель_на_экземпляр;
2. Создание динамического экземпляра класса с помощью операции **new**:

*указатель_на_экземпляр***= new** *имя_класса* (*фактические параметры конструктора*)**;**

3. Если экземпляр класса не нужен, то он уничтожается операцией **delete:**
   delete *указатель_на_экземпляр;//*при этом вызывается //деструктор класса

Вызов метода класса

Метод класса (по аналогии с полем структуры) вызывается одним из следующих способов:

1. *имя_экземпляра.имя_метода* (прямой выбор)
2. имя_экземпляра->имя_метода (косвенный выбор)

Схема работы с динамическим экземпляром класса

- Описание экземпляры класса:
  *имя_класса** **указатель_на_экземпляр*;***
- Создание экземпляра класса - с помощью оператора:
  указатель_на_экземпляр **= new** *имя_класса* (
  фактические параметры конструктора)**;**
- Использование экземпляра класса:
  имя_экземпляра.имя_метода
  или *имя_экземпляра->имя_метода*
- Уничтожение экземпляра класса:
  delete *указатель_на_экземпляр*

Пример.

Требуется разработать класс, основным методом которого является вычисление минимальных элементов строк матрицы, а затем применить его для обработки нескольких матриц.

Методы класса:

1) конструктор;
2) вычисление минимальных элементов строк (основной вычислительный алгоритм);
3) ввод матрицы (консольный);
4) вывод результатов (консольный).

Расчетное задание

1. Задача 3.6.N+1 с классами.
2. Пока в режиме консольного приложения, затем – оконное приложение.
3. Минимальный набор методов класса – см. предыдущий слайд.