Обектно-ориентирано програмиране (записки)

- **Марина Господинова**
- **Илиан Запрянов**

Дефиниции

Тема 01

def| Namespace

- инструмент за избягване на конфликти на имена
- scope (област на действие), в който има дефинирани символи (папка от символи)

def| Енумерация - enum

- тип, рестриктиран до домейн от стойности, които включват специално дефинирани константи. (енумератори)

def| Структури - struct

- последователност от полета, които се пазят в определен ред

def.| Alignment requirement

- разликата на 2 последователни адреса, на които можем да разположим дадена променлива

def.| Обединения - union

- последователност от полета, които заемат (споделят) една и съща памет

def.| Endianness

- начин на подреждане на байтовете в една дума

Тема 02

def.| Поток

- последователност от байтове “насочени” в определена посока

def.| Интерфейс (не сме го писали като def)

- набор от функции или методи (които можем да използваме, за да манипулираме даден обект/инстанция)

def.| Синхронизация (не сме го писали като def)

- правим промените в буфера и след някакъв интервал от време ще се изсипе във файла

Тема 03

def.| Двоичен файл
- готов за зареждане в паметта **(файлове с пряк достъп)**

Тема 04

def.| Член-функции
- всяка функция в тялото на структура/клас
- функции, които работят с член-данните на обекта от дадена структура

def.| Абстракция
- използваме нещо без да се интересуваме как работи
- скриване на ненужните детайли

def.| Капсулация
- ограничаване на достъпа

def.| Mutable (не сме го писали като def)
- член-данни, които могат да бъдат променяни дори от **const** функции

def.| Конструктор (не сме го писали като def)
- извиква се при създаването на обекта
- има същото име като класа/структурата
- няма тип на връщане
- можем да имаме много конструктори
- конструктор, който няма параметър се нарича default-ен конструктор

def.| Деструктор (не сме го писали като def)
- отговаря за затварянето на външни ресурси
- точно един, ако нямаме компилаторът ще създаде такъв
- също име като структурата само че **~ + име**

Тема 05

def.| Макроси (не сме го писали като def)
- замести парче код с друго парче код (стрингообработка)

def.| Копиращ конструктор (не сме го писали като def)
- приема обект от същия клас и текущия става негово копие
- ако не го разпишем, то компилаторът създава такъв

def.| Оператор= (не сме го писали като def)
- приема обект от същия тип и текущия става негово копие
- текущият обект е съществувал преди това
- ако не го разпишем, то компилаторът създава такъв

Тема 06

def.| Голямата четворка (не сме го писали като def)

- default констр. + деструктор + копиращ констр. + operator=

Тема 07

def| Оператор (унарни/бинарни/тернарен)

- функция със специален синтаксис

def| Приятелски класове/функции

- класове/функции, които имат достъп до private имплементациите ни

Тема 08

def| Static локални променливи (не сме го писали като def)

- държи се в паметта на глобалните/статичните променливи
- променяме продължителността на локалната променлива от automatic (т.е. до края на scope-а) на static (променливата се създава в началото на програмата и се унищожава в края на програмата, точно като глобалните променливи)
- инициализира се само веднъж - при първото влизане в съответния scope и запазва стойността си дори след като излезе от scope-а

def| Static функции (не сме го писали като def)

- обвързва се с една компилационна единица и не може да се използва от други файлове
- тоест не можем да използваме тази функция в друг .cpp файл

def| Static член-данна на клас (не сме го писали като def)

- не е обвързана с конкретен обект, а с целия клас
- всички обекти от класа използват една и съща инстанция
- инициализира се извън класа

def| Static член-функция на клас (не сме го писали като def)

- не е обвързана с конкретен обект, а с целия клас
- използва се за достъпване на статичните член-данни
- няма указател **this**
- не е нужен обект, за да се достъпи

def| Статичен клас (не сме го писали като def)

- клас, който има само статични член-данни

def| Exception

- сигнал, че има проблем

def| Stack unwinding (не сме го писали като def)

- при хвърляне на грешка изпълнението на функцията се прекратява
- програмата проверява дали текущата функция или някоя от извикващите функции нагоре по стека може да се справи с изключението (има **try-catch** блок)
- ако бъде намерен съответстващ блок за обработка на изключение, изпълнението се прескача от момента, в който е хвърлено изключението, до началото на съответстващия блок за обработка
- това изисква **stack unwinding** (премахване на текущата функция от стека на повикванията) толкова пъти, колкото е необходимо, за да може функцията, обработваща изключението, да бъде най-горе в call stack-а
- когато текущата функция се премахне от call stack-а, всички успешно създадени локални променливи се унищожават както обикновено, но не се връща стойност

01. ако някой обработва грешката
02. ако не е обработена -> **std::terminate**

Тема 09

def.| Колекция от обекти

- клас, който съдържа колекция от еднотипни обекти

def.| Placement new

- приема вече заделена памет и извиква конструкторите върху нея

def.| lvalue

- име на съществуваща променлива/функция

def.| prvalue

- литерали: **73, nullptr, “ABC”, false**
- извикване на функция, която връща копие

def.| xvalue (expiring value)

- обекти към края на жизнения си цикъл

def.| rvalue

- **prvalue + xvalue**

def.| rvalue reference

- **int&&**
- референция, която може да се прикачи единствено към **rvalue**

def.| Move конструктор

- конструктор за “откradване” на данни

Тема 10

def.| Наследяване

- **Der** е наследник на **Base**, ако разширява неговите данни/поведение

Тема 11

def.| Изглед

- клас, който се ползва за преглед на интервал от колекции

def.| Полиморфизъм

- едно име на функция, но много различни имплементации

def.| Compile time polymorphism

- по време на компилация се определя коя функция да се извика

def.| Статично свързване

- изборът на функция става при време на компилация
- определя се от: **типа на указателя/референцията**, от който се извиква функцията

def.| Динамично свързване

- изборът на коя функция да се извика става по време на изпълнение на програмата **(run-time polymorphism)**
- чрез виртуални функции

def.| Чисто виртуална функция (pure virtual function)

- функция, която няма имплементация
- предназначена да бъде пренаписана от наследниците
- може да има тяло

def.| Абстрактен клас

- клас, който има поне една чисто виртуална функция и е предназначен за наследяване
- ако чисто виртуалната функция не се разпише от наследник и той става абстрактен

def.| Виртуална таблица - “масив от указатели към функции”

- всеки клас, който има виртуални функции има своя виртуална таблица - в нея пише коя функция трябва да се извика
- всеки обект има виртуален указател като допълнителна член-данна (в началото на класа), която сочи към виртуалната таблица на класа и влияе на размера му (8 байта за 64-битова система)
- Невиртуалните функции не са в тези виртуални таблици
- Деструкторът, от друга страна, е в тази таблица, затова задължително при полиморфизъм деструкторът е виртуален, в противен случай - memory leak (достатъчно е да кажем само на този на класа най-отгоре на йерархията да бъде виртуален, останалите ще се направят по подразбиране)

Тема 12

def.| Множествено наследяване

- в случаите, когато производният клас наследява директно повече от един базов клас, се казва, че наследяването е множествено.

def.| Хетерогенен контейнер

- клас, който съдържа колекция от указатели към абстрактен клас и се грижи за менажирането на паметта

def.| Visitor Pattern

- когато си взаимодействат обекти от полиморфна йерархия (например член-функция на наследник, на която се подава обект от друг наследник)

Тема 13

def| Шаблон

- клас/функция с общо предназначение спрямо типа „ (необходима функция на тип)
- шаблон = параметричен полиморфизъм

def| Темплейтна специализация

- клас/функция с различно поведение спрямо типа

def| Обекти , които се държат като функции

- обекти на клас, който има предефиниран operator(), който смята резултат от функция

def| Умни указатели

- обвиващи класове на указателите, които менажират паметта на обектите, към които сочат; не се интересуваме от триенето, а го прави указателя (delete)

Тема 14

def| Type casting

- когато искаме да преобразуваме от един тип в друг

def| Design Patterns

- обобщени практики (добри)
- решения на често възникващи проблеми

(не специфичен код, а концепция за решение)