## Principy objektového programování

- Jaký je rozdíl mezi třídou a objektem?
  - Třída je šablona, objekt je konkrétní instance třídy, abstraktní/konkrétní
- Jak je určena viditelnost členských proměnných a metod?
  - 3 přístupové specifikátory:
  - Public volány a upravovány zevnitř i zvenku
  - Private jen zevnitř
  - Protected zevnitř a zevnitř tříd odvozených
- Jaký je rozdíl mezi členskými metodami (a proměnnými) třídy a objektu?
  - Deklarované v rámci třídy, každý objekt má vlastní sadu
- Jaký je rozdíl mezi klíčovými slovy protected a private?
  - Členské proměnné a metody deklarované jako private jsou přístupné jen zevnitř třídy, slovo protected zaručuje přístup zevnitř třídy i třídy odvozené
- Co je polymorfismus a jak je ho dosaženo?
  - Jedna funkce může provádět různé operace v závislosti na typu objektu, který funkci volá
  - Virtuální funkce deklarace v bázové třídě pomocí slova "virtual", potomci mají vlastní implementaci funkce
- Co je to bázová třída a čím se vyznačuje?
  - Třída, která slouží jako základ pro další třídy, definuje společné vlastnosti potomků
  - Odvozené třídy dědí vlastnosti bázové třídy, potomci dále rozšiřují a specializují chování
- Uveďte příklad deklarace plně virtuální metody?
  - Class Shape{ virtual void draw() = 0; }
  - Nemá definici pro bázovou třídu
- Co je to implicitní konstuktor?
  - Konstruktor automaticky generován kompilátorem, defaultní konstruktor
- Co je to kopírovací konstruktor?
  - Jediný parametr: konstantní reference na objekt stejné třídy
  - Vytvoří kopii objektu, pokud není definován uživatelem, je automaticky generován kompilátorem
- Jaké jsou možnosti dědění? Uveďte příklady.

- Public veřejné a chráněné členy jsou veřejné přístupní v odvozené třídě
- Protected jsou chráněné v odvozené třídě
- Private jsou přístupné pouze v odvozené třídě
- Class Derived : private Base {};
- Co je to virtuální dědění a k čemu je dobré?
  - Umožňuje vyhnutí se "problému diamantu", nejednoznačnosti při dědění, třída zdědí stejnou bázovou třídu přímo i nepřímo
  - Odvozená třída má jen jednu instanci
  - Class Animal {}; class Mammal : public virtual Animal {}; class Bird : public virutal Animal {}; class Platypus : public Mammal, public Bird {};
- Které vlastnosti rodiče potomek nedědí?
  - Konstruktory a destruktory
  - Přetížené operátory
  - Přátelské funkce
  - Static členy
  - Const metody
- Co je to přetížení? Uveďte příklad (několik řádek kódu, ze kterých bude zřejmé, o co jde).
  - Umožňuje definovat více funkcí nebo operátorů se stejným názvem, ale s odlišnými parametry (typ, počet parametrů)
  - Void printnumber(int number) {}void printnumber(float number) {}
- Jaký je rozdíl mezi přepsáním (override) a přetížením (overload)?
  - Přepsání odvozená třída přepisuje implementaci rodičovské třídy, odvozená třída stejný název funkce jako rodičovská třída
  - Přetížení definovat více funkcí se stejným názvem, jiné parametry
- Co je vazba typu kompozice? Uveďte příklad (několik řádek kódu, ze kterých bude zřejmé, o co jde).
  - Jedna třída obsahuje objekt jiné třídy
  - Život objektu v kompozici závisí na životnosti vlastního objektu
  - Class Engine {} class Car {Engine engine;}
- Co je vazba typu asociace? Uveďte přiklad (několik řádek kódu, ze kterých bude zřejmé, o co jde).
  - Jedna třída používá objekt jiné třídy, slabší než vazba kompozice
  - Umožňuje komunikaci třídy s objektem jiné třídy, objekty existují samovolně
  - Vytvářená ukazatelem, referencí
  - Class Teacher {} class Student {Teacher \*teacher;}

- Co reprezentuje klíčové slovo this?
  - Ukazatel na aktuální objekt, ve kterém je kód právě prováděn
- Jaký je rozdíl mezi referencí a ukazatelem?
  - Reference musí být inicializovaná při deklaraci, nemůže ukazovat na jiný objekt
  - Ukazatel obsahuje adresu objektu nebo proměnné, může se měnit adresa, může být null, je potřeba použít \* k přístupu k hodnotě
  - Reference elegantní syntaxe, ukazatel změna cíle, dynamická alokace
- Jaký je rozdíl mezi mělkou a hlubokou kopií?
  - Mělká kopie jsou zkopírovány pouze hodnoty ukazatelů, referencí, objekty sdílejí paměť, může dojít k chybě při úpravě dat
  - Hluboká kopie vytváří nové kopie dat, nový objekt má oddělenou paměťovou oblast

## Programování v C++

- Napište příklad načtení hodnoty typu int ze standarního vstupu pomocí proudového operátoru. Jak lze ošetřit správnost vstupu?
  - Int number; std::cin >> number; if(std::cin.bad()) {cout << "chyba";}
- Jak se v C++ provádí přetížení operátoru? Uveďte příklad.
  - Void print(int number) {cout << number;} void print(double number) {cout << number;}</li>
  - Class Complex { double real; double imaginary; Complex operator+(const Comlex &other){} Complex result; result.real = real + other.real; result.imaginary = imaginary + other.imaginary; return result};
- Jaké operátory nelze v C++ přetížit?
  - •
  - Sizeof()
  - ..
  - .\*
  - ?:
- Jak a kdy se volá konstruktor a destruktor?
  - Konstruktor při vytváření objektu, při vytváření na haldě, vytváření objektu jako člena třídy
  - Destruktor při ukončení platnosti objektu, při ručním uvolnění pomocí "delete", při zániku objektu jako člena třídy

- Jaký je rozdíl mezi objektem alokovaným v zásobníku a objektem dynamicky alokovaným na haldě?
  - Jinak se k nim přistupuje, dynamicky alokovaný je uložen v ukazateli
  - Objekt na haldě se nezničí automaticky po ukončení rámce
- Jaký je rozdíl mezi operátory delete a delete[]?
  - Delete se používá na k uvolnění paměti alokované jedním prvkem
  - Delete[] k uvolnění paměti alokované polem
- Jaká je návratová hodnota destruktoru?
  - Destruktor nemá návratovou hodnotu
- Co je třeba zajistit při dynamické alokaci? Uveďte příklad.
  - Uvolnění alokované pamětí pomocí delete, delete[]
    - Int \*ptr = new int; delete ptr;
  - Výjimky, chyby při alokaci
    - If(ptr == nullptr) { cout << "chyba"; }</li>
- Co je to výjimka a k čemu je dobrá? Uveďte příklad.
  - Umožňuje reagovat na neočekávané události, umožňuje zastavení toku programu a přesun k zvláštnímu kódu

```
std::ifstream file("neexistujici_soubor.txt");
```

if (!file) {throw std::runtime\_error("Chyba: Soubor se nepodařilo otevřít."); }

- Kolik výjimek je možné v programu ošetřit?
  - Neomezené množství
- Co je to generická funkce?
  - "šablona funkce", umožňuje vytvářet funkce, která pracuje s různými datovými typy

```
template < typename T>
T maximum(T a, T b) {
  return (a > b) ? a : b;
}
```

- Jak lze vytvořit explicitní instanci generické funkce? K čemu je to dobré?
  - Definice funkce pro určitý specifický datový typ

template<>

```
std::string maximum(std::string a, std::string b) {
  return (a.length() > b.length()) ? a : b;}
```

- Jak jsou obvykle parametrizovány šablony tříd?
  - template <typename T> class Template ...
  - Template < int > obj;

- V čem spočívá problém vícenásobného dědění? Jak je možné tomuto problému zabránit?
  - Odvozená třída dědí od více bázových tříd, které dědí od stejného předka, zdědí některé prvky vícekrát
  - Virtuální dědění
  - Class B : virtual public A { };
- Co je to zapouzdření a čemu je dobré?
  - Není veřejný přístup k proměnné třídy, k proměnným se dá přistupovat jen skrz veřejné rozhraní, skrytí vnitřní reprezentace dat
  - Ochrana proti chybám, rozdělení do nezávislých částí
- Jak lze využít implicitní hodnoty argumentu funkce? Uveďte příklad.
  - Volání funkce s vynecháním argumentu

void Greet(const std::string& name, const std::string& greeting = "Hello") {}
int main() { Greet("John"); }

- Jaký je rozdíl mezi deklarací třídy pomocí class a struct.
  - Ve viditelnosti
  - Class výchozí viditelnost je private
  - Struct výchozí viditelnost je public
- Co je to konverzní konstruktor?
  - Konstruktor, který změní typ objektu
- Jaký je rozdíl mezi statickou a dynamickou vazbou?
  - Statická rychlejší, neumožňuje polymorfismus, rozhodnutí o volání při překladu
  - Dynamická pomalejší, určuje volanou metodu až v průběhu na základě typu objektu
- Jaký je význam klíčového slova auto? Lze ho použít vždy?
  - Deklarace proměnné s automatickým určením datového typu
  - Nejde musí se inicializovat, ne deklarace parametru funkce, ne členy třídy
- Co přesouvací (move) kontruktor?
  - Přesunutí objektu do objektu stejného typu, &&, std::move
  - Dobré pro velké datové struktury, nepřenáší se data, ale vlastnictví zdrojů

```
Vector(Vector&& other) {
    size = other.size;
    data = other.data;
    other.data = nullptr;
    other.size = 0;}
```

Vector v1(1000);

Vector v2 = std::move(v1);

- Co je pravidlo tří (rule of three)?
  - Kopírovací konstruktor, přetížení operátoru přiřazení, destruktor
  - Pokud se definuje jeden, je dobré definovat i ostatní, pro správné fungování kódu
- Jaký je tvar definice lambda funkce v C++? Uveďte příklad a popište, co funkce dělá.
  - [capture] (parameters) -> return\_type {tělo fce}
  - Pohodlný způsob jak definovat jednoduché funkce
  - auto sum = [](int a, int b) { return a + b; };

## **STL** kontejnery

- Co je to kontejner?
  - Datová struktura, uschování, organizace a manipulace hodnot
  - Přidávání, vyhledávání, odbírání hodnot
- Jaký je rozdíl mezi sekvenčními a asociativními kontejnery? Uveďte příklad.
  - Sekvenční prvky uloženy ve specifickém pořadí, přístup k prvkům na základě indexu, pozice (vektor, seznam)
  - Asociativní prvky uloženy pomocí hodnot a klíčů (mapa, množina)
- Jaké jsou klíčové vlastnosti kontejneru std::vector?
  - Jeho velikost se automaticky mění s přidáváním/odbíráním prvků
  - Lineární uspořádání prvků, přístup pomocí indexu
- Jaké jsou klíčové vlastnosti kontejneru std::map?
  - Uspořádání na základě klíčů, páry klíč/hodnota, každý klíč jedna hodnota
  - Dynamicky rozšiřitelný velikost se automaticky upravuje
- Jakého datového typu může být klíč kontejneru std::map?
  - Libovolný datový typ, který se dá porovnávat operátory <, >, <=, >=
- Co je třeba zajistit v případě, že klíčem v kontejneru std::map je strukturovaný datový typ (např. naše vlastní třída)?
  - Přetížení operátorů <, >, <=, >=
  - Stabilita porovnání datový typ stabilní, časem se nemění vzájemné porovnání
- Jaký je význam datového typu std::pair?

- Umožňuje uložení a spojení dvou hodnot jiného datového typu, sdružení dvojice do jednoho páru
- Std::pair<int, std::string> myPair; myPair.first = 1; myPair.second = "a";
- Co je to iterátor? Uveďte příklad.
  - Objekt, umožňuje procházet prvky kontejneru
  - std::vector<int> numbers = {1, 2, 3, 4, 5};
  - std::vector<int>::iterator it = numbers.begin();
- Jaký je rozdíl mezi konstatním a nekonstantním iterátorem?
  - Konstantní nejde upravovat prvky, na které ukazuje
  - Nekonstantní dají se upravovat
- Co všechno je třeba zajistit pro vytvoření vlastního iterátoru?
  - Operátor inkrementace a porovnání
  - Správné nastavení rozsahu
  - Přístup k prvkům, \*
- Co je range-for cyklus? Uveďte příklad.
  - Jednoduchý průchod kontejnerem, automatické vytvoření iterátoru
  - for (const auto& cislo : cisla)
- Uveďte příklad použití algoritmu sort (deklarujte vektor libovolného datového typu, vložte do něj data a vektor seřaďte).
  - Std:.vector<int> cisla = {2, 4, 1};
  - std::sort(cisla.begin(), cisla.end());

## Praktické programování (Qt a související)

- Co je to event-driven programování?
  - Zaměřuje se na zpracování událostí a odezvu na události
  - Aplikace nemusí stále sledovat stav, ale jen reaguje na eventy
  - Systém signálu a slotů
- Jaký je rozdíl mezi událostí a signálem?
  - Event interakce s uživatelem, klik myší, dění v aplikaci
  - Signál slouží k oznámení události, může být emitovaná objektem, reagují na ně sloty, oznámení dalším objektům
- Uveďtě příklady událostí v Qt a jejich využití.
  - Kliknutí myší (QMouseEvent), Stisknutí klávesy (QKeyEvent), Uzavření okna (QCloseEvent):
- Popište systém signálů a slotů, uveďte příklad propojení.
  - Umožnuje komunikaci mezi objekty, signál je vyslaný objektem, sloty jsou metody, které se spouští jako odpověď na signálu

- QPushButton button("Click me!");
- QObject::connect(&button, &QPushButton::clicked, [&]() {});
- Jakým způsobem lze vytvořit v Qt okno a v něm vykreslit grafickou komponentu?
  - QPushButton b("Hello"); b.show();
  - QWidget \*window = new QWidget;
  - Window->addWidget(new QPushButton("Hello"));
  - Window->show();
- Jak lze měnit grafickou podobu jednotlivých Qt komponent?
  - setyStyleSheet("background-color: red; color: white;")
  - .setText .setFont
- Co je časovač (QTimer), k čemu je dobrý a jak se používá?
  - Umožní vytvářet události s časovou prodlevou
  - Spustí se, po ukončení vydá signál timeout(), je potřeba propojit se slotem
  - QTimer\* timer = new QTimer(this);
  - connect(timer, SIGNAL(timeout()), this, SLOT(handleTimeout()));
  - timer->start(1000);
- Co si představujete pod pojmem MVC?
  - Architektura pro návrh aplikace s UI, rozdělení na nezávislé části
  - Model datový model, se kterým pracuje aplikace
  - View uživatelské rozhraní, vizualizace dat
  - Controller řídící logika, změny dat na základě uživatelských inputů
- Jaký je rozdíl mezi UDP a TCP transportní vrstvou?
  - TCP spolehlivý, kontroluje ztráty, řízení spojení
  - UDP jednoduchý, nekontroluje doručení, pakety odesílány nezávisle
- Co je to REST rozhraní?
  - Architektura, umožňuje komunikace s webovými službami