1. procedūrinio programavimo kalbos – asemlerio abstrakcijos (C, Fortran), objektinio – realu pasauli.
2. Abstraktus duomenu tipas apibrezia: duomenis, operaciju aibe, kurias galima atlikti su to tipo duomenimis. C++ abstraktus duomenu tipas yra modeliuojamas klase. Klase yra tipas. Objektas yra to tipo kintamasis. Dar sakoma, kad objektas yra klases egzempliorius.
3. Programavimo paradigma yra programu kurimo modelis. Smalltalk, Alain Kay:

-Sprendimų erdvė yra sudaryta iš objektų

- Objektas – savotiškas kintamasis: duomenys + operacijos

– Programa yra objektų aibė, kurie siunčia vienas kitam pranešimus, nurodančius atlikti operacijas

– Nauji objekta i yra komplektuojami bei kuriami panaudojant jau egzistuojančius objektus

– Kiekvienas objektas yra tam tikro tipo = kiekvienas objektas yra tam tikros klasės egzempliorius

– Visi tam tikros klasės objektai moka atlikti tas pačias operacijas

4. C++ kompiliavimo procesas

1) Preprocesorius

- Pakeičia vienus išeities teksto “šablonus” kitais

– #include : Išorinių išeities tekstų įtraukimas

– #define : *macros*

– #if : sąlyginis teksto įtraukimas

2) Kompiliatorius

- Analizuojamas preprocesoriaus apdorotas išeities tekstas bei sukuriama tarpinė forma

– Tikrinama sintaksė

– Statinis tipų tikrinimas

- Sintezės metu iš tarpinės formos generuojama programa , kuri užrašyta mašininėmis instrukcijomis arba asembleriu (tokiu atveju ją reikia asembliuoti)
- Rezultate gaunamas *objektinis* failas

– .o arba .obj

3) Linkeris

- Linkeris sujungia nurodytus objektinius failus į programą , kurią gali įvykdyti operacinė sistema

– Išsprendžiamos nuorodos į kituose objektiniuose moduliuose (tame tarpe ir bibliotekose) esančias funkcijas bei kintamuosius

– Tikrinama ar deklaruoti kintamieji bei funcijos iš tiesų yra apibrėžti (sukurti)

– Į programos pradžią įterpiamas specialus objektinis modulis, reikalingas programos inicializavimui *(startup activities)*

5. Išskaidytos programos ir jų kompiliavimas

- Kuriant didelę programą yra svarbu ją suskaidyti į mažesnes nepriklausomas dalis, vadinamas poprogramėmis
- C, C++: funkcijos
- Skirtingos funkcijos gali būti sudėtos į atskirus failus , kurie gali būti atskirai testuojami bei kompiliuojami
- Funkcija yra atominis išeities teksto vienetas
- Skirtinguose failuose esančios funkcijos turi viena apie kitą žinoti failų kompiliavimo metu

6. Funkcijų ir kintamųjų deklaravimas ir apibrėžimas

- C, C++ : funkcijos ir kintamieji būtinai turi būti deklaruoti
- Deklaravimu kompiliatoriui yra pasakoma, kad f. ar kint. kažkur yra apibrėžti
- Apibrėžimu f. ar kint. yra sukuriami
  - Atminties išskyrimas
- Programoje f. ar kint. gali būti deklaruoti daugelį kartų (skirtinguose failuose), bet jie apibrėžiami vieną kartą
- Apibrėžimas tarnauti ir kaip deklaravimas

Funkcijos deklaravimas

- Deklaravimas
  - int func1(int,int) ;
  - int func1(int length, int width) ;
- Iškvietimas
  - a = func1(2,3);
- Kompiliavimo metu tikrinami tipai
- int func2();
  - C : bet koks skaičius bet kokio tipo argumentų
- Deaktyvuoja tipų tikrinimą
  - C++ : funkcija be argumentų
  - *Funkcijos apibrėžimas*
- Funkcijos apibrėžimas pateikia funkcijos kūną
  - int func1(int length, int width) { }

Kintamojo deklaravimas ir apibrėžimas

- Deklaravimui naudojamas raktinis žodis extern
  - extern int a;
  - extern int func1(int length, int width);
- Kintamojo apibrėžimas
  - int a;

7. Header failai

- Bibliotekų funkcijų bei kintamųjų *extern* deklaracijos yra sudedamos į *header* failus
  - .h
- Norint naudoti biblioteką, reikia į išeities tekstą įtraukti jos *header* failą
- #include preprocesoriaus direktyva
- #include <iostream>
- #include <cstdio>
- #include ”mano.h”

Bibliotekų naudojimas

1) Į išeities tekstą įtraukiamas bibliotekos *header* failas

2) Naudojamos bibliotekos funkcijos bei kintamieji

3) Bibliotekos objektinis failas sulinkuojamas į vykdomą programą

- Standartinė C++ (ir joje esanti standartine C) bibliotekos

8. The solution in C++ is to combine all the actions necessary to create an object into a single operator called **new** . When you create an object with **new** (using a *new-expression*), it allocates enough storage on the heap to hold the object and calls the constructor for that storage.

The complement to the new-expression is the *delete-expression* , which first calls the destructor and then releases the memory (often with a call to **free( )**). Just as a new-expression returns a pointer to the object, a delete-expression requires the address of an object.

9.

- (Konstantos)
- Struktūra, kurioje yra nariai kintamieji

Duomenys:

- Funkcijos, kurios dirba su struktūra

The this keyword is a pointer to the current object. All member functions of a class have a this pointer.

C struktūros trukumas – negalima idet I ja funkciiju. C++ strukturoje nebereikia typedef.

10. Inkapsuliacija :

duomenų (kintamųjų,konstantų) ir operacijų (funkcijų) apjungimas sukuria naują duomenų tipą.

11. **Kodel reikia matomumo kontroles?**

- Paslėpti nuo programuotojo kliento vidinės realizacijos dalykus
- padeda išvengti klaidų (bugs), kylančių dėl to, jog programuotojas klientas įsikiša į vidinės realizacijos dalykus
- padeda susigaudyti, kas yra svarbu, o į ką nekreipti

dėmėsio

Bibliotekos kūrėjas gali keisti (optimizuoti) vidinės realizacijos dalykus, nesirūpindamas, kaip tai paveiks programuotojo kliento sukurtas programas (nes nepaveiks)

Visi struktūros nariai pagal nutylėjimą yra public

private narius gali pasiekti/matyti tiktai tos struktūros narės funkcijos

protected narius gali matyti tiktai tos struktūros narės funkcijos arba iš tos struktūros paveldėjusios narės funkcijos

12. Draugai - friends

- Jeigu norime, kad *kas nors* iš išorės matytų struktūros private narius , reikia tą *kas nors* struktūroje aprašyti kaip draugą , panaudojant raktinį žodį friend
- *Kas nors* gali būti
- funkcija globalioje srityje (global scope function)
- kitos struktūros funkcija
- kita struktūra

13. C++ struktūra

- inkapsuliacija
- matomumo kontrolė
- C++ struktūra duomenų tipas
- aprašo objektų klasę (tipą)
- C++ struktūra ir klasė yra tas pats, tik:
- visi struktūros nariai pagal nutylėjimą yra public
- visi klasės nariai pagal nutylėjimą yra private

14. Inicializacija

- Daug programavimo klaidų kyla užmiršus tinkamai inicializuoti kintamuosius ir objektus
- Programuotojas klientas turi nepamiršti iškviesti inicializacijos funkciją

15. C++ konstruktorius

- Klasės narė funkcija, kuri vadinasi taip pat, kaip ir klasė
- Ši funkcija neturi grąžinamos reikšmės
- Šios funkcijos iškvietimą kompiliatorius įterpia po kiekvieno objekto apibrėžimo

konstruktorius pagal nutylejima

- Konstruktorius pagal nutylėjimą yra toks konstruktorius, kuris neturi argumentų
- Jeigu koks nors konstruktorius yra apibrėžtas, kompiliatorius jį būtinai iškvies kuriant objektą
- Jeigu klasėje nėra apibrėžto konstruktoriaus, kompiliatorius jį sugeneruos pats
- Toks konstruktorius nieko nedaro

16. destructorius

Kiekvienoje klasėje yra speciali funkcija, kuri yra iškviečiama sunaikinant objektą destruktorius

- Ši funkcija, kaip ir konstruktorius neturi grąžinamos reikšmes
- Destruktorius niekada neturi argumentų, nes jam jų negali prireikti – jis nenustato jokių reikšmių
- Destruktorių iškviečia kompiliatorius, kai programos vykdymas išeina iš objekto srities *(scope)*

17. Scope tai kas yra tarp { }

18. Objektu kurimas ir salyginis programos vykdymas

- *Objektas negali būti apibrėžtas programos* srityje *(scope)*, kuri yra vykdoma sąlygiškai, nes
- Kiekvienam objektui būtinai turi būti iškviestas konstruktorius
- Apibrėžiant objektą yra iškviečiamas konstruktorius
- Jeigu objektas yra apibrėžiamas sąlygiškai vykdomoje programos dalyje, tada objektas yra srityje *(scope)*, bet jo konstruktorius nebūtinai bus iškviestas (t.y.,sąlygiškai iškviestas)