. DEFINICIJA I VRSTE APSTRAKCIJE. PRINCIP SKRIVANJA INFORMACIJA

DEF: APSTRAKCIJA JE IZDVAJANJE BITNIH OSOBINA I ZANEMARIVANJE NEBITNIH KOJE SE OBDACUJU U PROCESU MODELIRANJA. POJAM SE MOŽE SMATRATI APSTRAKCIJOM AKO SE MOŽE OPISATI, AKO SE RAZUME I ANALIZIRA NEZAVISNO OD OD NAČINA NA KOJI JE REALIZOVAN. BOOCH GRADY JE TVORAC OBJEKTNOG PROGRAMIRANJA KOJI JE PRE 10 GODINA OBJEDINIO OVE DVE DEFINICIJE APSTRAKCIJE: APSTRAKCIJA KAO REZULTAT APSTRAHOVANJA OZNAČAVA SKUP BITNIH OSOBINA ENTITETA KOJA GA ODVAJAJU OD OSTALIH VRSTA ENTITETA OBEZBEĐUJUĆI TAKO OŠTRE POJMOVNE GRANICE, SVE U ZAVISNOSTI OD UGLA POSMATRANJA. U OBJEKTNOM PRG APSTRAKCIJA JE PRINCIP, A NE TEHNIKA. PRINCIP APSTRAKCIJE GLASI: INTERFEJS KLASE MORA BITI STROGO RAZDVOJEN OD NJENE REALIZACIJE. **KLASIFIKACIJA** APSTRAKCIJA ENTITETA: OBJEKAT JE MODEL STVARNOG ENTITETA, APSTRAKCIJA AKCIJE: OBJEKAT PRIKUPLJA SKUP OPERACIJA MEĐUSOBNO POVEZANIH (NPR KOMPLEXNI BROJ), APSTRAKCIJA TIPA VIRTUELNE MAŠINE (NPR LOG): GDE SE SKUP OPERACIJA NIŽEG NIVOA ZAMENJUJE JEDNOM OPERACIJOM VIŠEG NIVOA KAO NPR SLOJEVI OS, OSI MODEL. MANJE BITNA JE SLUČAJNA APSTRAKCIJA NEVEZANE OPERACIJE U OBJEKTU NPR GREŠKA.

PRINCIP SKRIVANJA INFORMACIJA

PARNAS

‘

73-'74. UČINITI INFORMACIJU NEVIDLJIVOM, JER JE APSTRAKCIJA PODELILA REALIZACIJU I INTERFEJS. U PASKALU POSTOJE TRI NIVOA: PRIVATE, PUBLIC I PROTECTED, A U DELPHIJU POSTOJI I PUBLISHED.

6. POJAM INKAPSULACIJE I REALIZACIJA U C++. PRAVILO INKAPSULACIJE

DEF INKAPSULACIJE POSTUPAK OBJEDINJAVANJA STRUKTURE I PONAŠANJA (FUNKCIONALNOSTI) U SOFTVERSKU CELINU TAKO DA SE OBEZBEDI KONTROLA PRISTUPA. INKAPSULACIJA JE SADRŽANA U NAREDBI CLLAS: NPR CLASS ime {…}; KONTROLA PRISTUPA OSTVARUJE SE LABELAMA PRIVATE, PUBLIC I PROTECTED. ATRIBUTI I METODE SU PO DEFAULT-U PRIVATE ZA RAZLIKU OD PASKALA GDE SU PUBLIC. POJEDINI DELOVI KLASE SU OTVORENI, A TREBA DA SU ZATVORENI (PODACI ČLANOVI).

PRAVILO INKAPSULACIJE

:

OBJEKAT TREBA KORISTITI DISCIPLINOVANO U SKLADU SA DOKUMENTACIOJOM PROIZVOĐAČA.

7. POJAM MODULA I PRIMENA U OBJEKTNOM PROGRAMIRANJU

MODULARNOST JE OSOBINA ONIH SISTEMA KOJI SU RAZBIJENI NA CELINE. **DEF MODULA:**

MODULJE SOFTVERSKA KOMPONENTA KOJA SE REALIZUJE AUTONOMNO (PROJEKTUJE+KODIRA+TESTIRA

=

REALIZUJE). GLAVNI PROGRAM I UNITI IMAJU OSOBINU MODULA, A POTPROGRAMI NEMAJU.

IZVORNE DATOTEKE C, C++ - SA SU SVE RAVNOPRAVNE, A ULOGU GLAVNOG PROGRAMA IMA DATOTEKA SA MAIN-NOM. MODUL IMA DATOTEKE *.HPP (*.H U C-U), *.CPP (*.C), *.OBJ. MODUL SE U SVAKOM PROGRAMSKOM JEZIKU REALIZUJE DVODELNO (INTERFEJS I IMPLEMENTATION), A U C-U INTERFEJS JE FIZIČKI ODVOJEN OD REALIZACIJE. NAČIN SMEŠTANJA KLASE U MODUL PRIMER:

class Complex {…} → complex.hpp

void Complex::ModArg(...){...} → complex.cpp

void Complex::Add(...){...} → complex.cpp

*.CPP SE PREVODI I KORISTI, NEMA TEXTUALNI OBLIK

8. KLASIFIKACIJA OPERACIJA NAD OBJEKTIMA

OPERACIJE - **1. GRUPA KONSTRUKTOR** JE OPERACIJA KONSTRUISANJA (NPR1. double r =1.3;
KOSTRUISANJE PROMENLJIVE, NPR2. Complex q; KONSTRUISANJE OBJEKTA, KLASE).
KOSTRUKTORI SU NAJVA
Ž
NIJE METODE (SVAKA KLASA IMA BAR 2 KONSTRUKTORA). Complex q;
JE POZIV METODE (EXPLICITNO).

2. GRUPA DESTRUKTORI

– OPERACIJA DESTRUKCIJE NAD PROMENLJIVOM VRŠI UNIŠTENJE PROMENLJIVE. EXPLICITNO SE VRŠI UNIŠTENJE PROMENLJIVIH POKAZIVAČKOG TIPA. SVAKA KLASA SADRŽI SAMO I TAČNO JEDAN DESTRUKTOR PORED 2 KOSTRUKTORA. PO DESTRUKCIJI OBJEKAT SE NE MOŽE KORISTITI.

3. GRUPA SELEKTORI

– PRISTUP DELOVIMA SLOŽENOG OBJEKTA ILI SLOŽENE PROMENLJIVE (NPR1. INDEXIRANJE PROMENLJIVE TJ NIZ, NPR2. POLJA SLOGA, NPR3. DEREFERENCIRANJE, NPR4. Re, Im KOMPLEXNOG BROJA).

4. GRUPA INDIKATORI

– DAJU INFORMACIJU O STANJU OBJEKTA; NIJE OBAVEZAN, ALI SE SREĆE (NPR U KLASI STEK: EMPTY, FULL).

5. GRUPA MODIFIKATORI

– METODE KOJE MENJAJU STANJA OBJEKTA (PROMENLJIVAMA VREDNOSTI), IMA IH U SVAKOJ KLASI.

6. GRUPA ITERATORI

– METODE KOJE RADE PROGRAMERI SLUŽE ZA PRISTUP SVIM DELOVIMA SLOŽENIH PROMENLJIVIH, OBJEKATA (NPR1. for, NPR2. with ZA SLOG).

9. KOSTRUKTORI I DESTRUKTORI U C++

Complex z; double a; - Complex z; JE NAJUSLOVNIJEM SMISLU REČI POZIV METODE, A DRUGI NAZIV JE POZIV **KONSTRUKTOR**A. SVAKA KLASA U C++ IMA BAR JEDAN KONSTRUKTOR: PODRAZUMEVANI UGRAĐENI KONSTRUKTOR (PUK) SE NAVODI, U SUPROTNOM NEMA PARAMETRE. KADA SE KONSTRUKTOR EXPLICITNO POZIVA JAVLJA SE PODRAZUMEVANI. AKO KLASA IMA BAR JEDAN KONSTRUKTOR KOJI JE NAPISAO PROGRAMER ONDA SE PUK NE MOŽE VIŠE KORISTITI. PODELA:

KONSTRUKTORI OBJEKTA (KO) I KONSTRUKTORI KOPIJE (KK)

. PODELA NIJE DISJUNKCIJA TJ KK SADRŽI DODATNE OSOBINE OSIM OSOBINA KO (KK JE PODGRUPA KO).

KO

POSEBNA METODA: NEMA NIKAKAV TIP PA NI VOID, NE VRAĆA NIŠTA, A PO IMENU SE MORA POKLAPATI SA IMENOM KLASE. ZA RAZLIKU OD SVIH METODA KONSTRUKTORI SU NA NIVOU KLASE. PRIMER SA 4 KONSTRUKTORA:

```
Class Complex

{

    private:    double r, i;
```

```
    public:     Complex ( ) { r = 0; i = 0;}

                Complex ( double rl, double img ) { r = rl; i = img; }

                Complex ( Complex z) – ne može, nije inline KK; PARAMETAR    KONSTRUKTO
RA NE MOŽE BITI TIPA PROMENLJIVE

                Complex ( Complex *);

                Complex (Complex &);

};

Complex :: Complex (Complex *z)

{

   r = ( *z).r, i = (*z).i; - pokazivači na objekte se pišu r = z -> r; i = z ->i;

}
```

(Complex &) se slično izvodi

Complex ( ) { }ZAUZIMA MEMORIJU I NE RADI NIŠTA. PRAVILO DOBROG PROGRAMIRANJA: PRVO SE U public – U NAVEDU KONSTRUKTORI, OBIČNO SE PRVO FORMIRA PODRAZUMEVANI KONSTRUKTOR, PA MAKAR I PRAZNO TELO. U KLASI SE MOGU POJAVITI I KONSTRUKTORI ISTOG IMENA KOJI SE RAZLIKUJU PO PARAMETRIMA. Complex z; - POZIV PUK-A, Complex a (3, 4.2); -a JE NAZIV OBJEKTA, POZIV DRUGOG, Complex c (
&
a); POZIV POKAZIVAČA, Complex d (a); POZIV 4-TOG KONSTRUKTORA KOJI PRIPADA REFERENCI. SVA 4 KONSTRUKTORA VRŠE INICIJALIZACIJU Z. KONSTRUKTOR-INICIJALIZATOR SINTAXNO SE RAZLIKUJE, JER IMA POSEBAN DEO ZA INICIJALIZACIJU (NAJČEŠĆE SE KORISTI U C++); ZAMENA ZA PUK. KLASE KOJE NEMA JU KONSTRUKTORE SU IZUZETAK.

KK

SADRŽI ISTE OSOBINE KAO I KO. PARAMETAR IM JE REFERENCA NA OBJEKAT ISTE KLASE:

```
Complex ( Complex & );

    ...

Complex :: Complex ( Complex &z )

{r = z . r ;  i = z . i;}
```

GLAVNU ULOGU IGRA U PRENOSU OBJEKTA KAO STVARNOG PARAMETRA KADA JE PRENOS PO VREDNOSTI. POJAVLJUJE SE TAMO GDE SE IZ SLOBODNE FUNKCIJE (METODE) VRAĆAJU VREDNOSTI KOJE SU OBJEKTI. KONSTRUKTOR KOPIJE UVEK POSTOJI KAO I KONSTRUKTOR OBJEKTA PA OTUD NAZIV PODRAZUMEVANI UGRAĐENI KONSTRUKTOR KOPIJE. KK VRŠI PREPISIVANJE NA STEK. KK MORAMO PROGRAMIRATI KADA KLASA IMA POKAZIVAČE (DINAMIČKE PODATKE ČLANOVE). NA PITANJE DA LI SE IZBEGAVA UKLJUČIVANJE KK DOBIJAMO ODGOVOR: AKO SU U PITANJU MALI OBJEKTI MOŽE SE IZBEĆI, JEDINO SE KOD VELIKIH OBJEKATA IZBEGAVA U SLUČAJU PRENOSA PO REFERENCI, A NE PO VREDNOSTI (UNOSI SE REFERENCA TJ ADRESA: &): PRIMER double f(Const Complex &(q)) {}, q MOŽE SE DESITI DA VRATI KROZ LOKALNU PROMENLJIVU ILI LOKALNU ADRESU (KOJA NE POSTOJI) TAKO DA SE RETKO KORISTI: Complex &q(α)

{…}.

DESTRUKTORI

SLU

ŽE ZA PONIŠTAVANJE OBJEKTA. SVAKA KLASA IMA DESTRUKTOR. IMA ISTO IME KAO KLASA SAMO ŠTO POČINJE TILDOM (

~

). NPR

~

Complex ( )

{ }-

UGRAĐENI. POZIV JE RETKO IMPLICITAN. KAD OBJEKAT IZAĐE IZ OPSEGA AUTOMATSKI SE UKLJUČI DESTRUKTOR – TO JE TAJ IMPLICITAN POZIV. KONSTRUKTOR KOPIJE POSTOJI ZBOG DESTRUKTORA. NPR AKO NEKA METODA IMA POKAZIVAČE NA OBJEKAT KOJI SE NALAZI NA STEKU (TA METODA JE FUNKCIJA PO VREDNOSTI) KONSTRUKTOR KOPIJE OMOGUĆUJE DA SE DESTRUKCIJOM NE UNIŠTI ORIGINAL NEGO SE FORMIRA KOPIJA (U OVOM SLUČAJU) LISTE, TAKO DESTRUKTOR UNIŠTAVA KOPIJU. U DRUGIM SLUČAJEVIMA SE IZBEGAVA UKLJUČIVANJE DESTRUKTORA PA TAKO I KONSTRUKTORA KOPIJE.

10. KOOPERATIVNE FUNKCIJE U C++

PRIJATELJSKE **F**UNKCIJE SU OBIČNE FUNKCIJE NISU METODE. AKO JE SLOBODNA FUNKCIJA PRIJATELJSKA KLASI ONDA IMA PRISTUP SVIM NJENIM PODACIMA PA I ZAŠTIĆENIM KAO DA JE DEO KLASE.

class T

{

```
    private: double a;

    public: friend double f( T );      U PRIVATE-U OVO NE BI FUNKCIONISALO DA
}                                      PF NE OMOGUĆAVA PROTOTIP FUNKCIJE U
double f( Tx )                         JAVNOJ SEKCIJI (PUBLIC).
{
  return 2*sin( x . a );
}
```

OSNOVNA FUNKCIJA PF JE BRZINA RADA (SPORIJE JE DODATI METODU ZA OČITAVANJE a, PA POTOM NJEN POZIV). KLASA FUNKCIJU PROGLAŠAVA PRIJATELJSKOM (A MOŽE I KLASE BIRATI ZA PRIJATELJSKE KAO I DELOVI KLASE TJ NEKE METODE). OTVARANJE DRUGE KLASE JE IZUZETAK TJ NARUŠAVANJE PRAVILA PRIKRIVANJA METODA. ZA PRIMER Complex KORISTE SE METODE Z . Re( ) I Z . Im ( ). U KLASU Complex TREBA UKLJUČITI PF-JE: friend double Re(const Complex &Z) {return Z . r}. REALIZACIJA SE MOŽE IZVRŠITI INLINE. ISTO SE IZVODI I Im.

11. POJAM I VRSTE POLIMORFIZMA

POSTOJI POLIMORFIZAM FUNKCIJA (POTPROGRAMA) I POLIMORFIZAM PROMENLJIVIH I VREDNOSTI. char c; - c MOŽE BITI CELOBROJNA. ZNAKOVNI TIP PRIPADA CELOBROJNOM I TO JE RAZLOG POLIMORFIZMA PROMENLJIVIH ZNAKOVNOG TIPA. KLASIFIKACIJA POLIMORFIZMA – **UNIVERZALNI I AD HOC (U TU SVRHU)**. KOD UNIVERZALNOG: ISTA KATEGORIJA SE RAZLIČITO PONAŠA ZAVISNO OD KONTEXTA. AD HOC: VEĆI BROJ KATEGORIJA ISTOG IMENA (KONKATANACIJA TJ SPAJANJE STRINGOVA, SABIRANJE, UNIJA SKUPOV U PASKALU) I PREDSTAVLJA SKUP POLIMORFIZAMA.
UNIVERZALNI SE DELI NA PARAMETARSKI I INKLUZIVNI.
KOD PARAMETRIZOVANIH ILI GENERIČKIH (TEMPLATE) KLASA (TIPOVA) IMAJU PARAMETAR KOJI JE KLASA. NPR. AKO NAPRAVIMO KLASU stack ČIJI SU ELEMENTI TIPA T, push DOBIJA PARAMETAR ZA KOJI SE NE ZNA ŠTA JE, TEK SE PRI OČITAVANJU DODELJUJE PRAVI TIP. INKLUZIVNI UNIVERZALNI POLIMORFIZAM VEZUJE SE ZA PROMENLJIVE KOJE SE PONAŠAJU KAO DA PRIPADAJU VEĆEM BROJU TIPOVA. KLASE MORAJU BITI POVEZANE AKO OBJEKAT PRIPADA ČAS JEDNOJ ČAS DRUGOJ KLASI, A KLASE SE MORAJU POVEZATI NASLEĐIVANJEM.
PODELA AD HOC POLIMORFIZMA: PREKLAPANJE OPERATORA (OVERLOADING) I KOERCITIVNI (PRINUDNI) POLIMORFIZAM
. PREKLAPANJE OPERATORA JE KORIŠĆENJE ISTIH SIMBOLA ZA RAZLIČITE OPERACIJE (& JE ADRESNI OPERATOR, OPERATOR BIT OPERACIJE, …). POSTOJI MOGUĆNOST PREKLAPANJA FUNKCIJA (SAMIM TIM I METODA) NPR: KOD KONSTRUKTORA, SLOBODNE FUNKCIJE I METODE MOGU NOSITI ISTA IMENA, A RAZLIKOVATI SE PO TIPOVIMA ARGUMENATA (BAR JEDNOG). PRINUDNI POLIMORFIZAM – KONVERZIJA TIPA NPR: (int a) – a SE MORA PONAŠATI KAO INT, IAKO MOŽDA PRE TOGA NIJE BILO int.

12. PREKLAPANJE OPERATORA U C++

PREKLAPANJE **O**PERATORA JE KORIŠĆENJE ISTIH SIMBOLA ZA RAZLIČITE OPERACIJE (& JE ADRESNI OPERATOR, OPERATOR BIT OPERACIJE, …). POSTOJI MOGUĆNOST PREKLAPANJA FUNKCIJA (SAMIM TIM I METODA) NPR: KOD KONSTRUKTORA, SLOBODNE FUNKCIJE I METODE MOGU NOSITI ISTA IMENA, A RAZLIKOVATI SE PO

TIPOVIMA ARGUMENATA (BAR JEDNOG). PO SE MOŽE IZVODITI SAMO U OBJEKTNOM DELU I OMOGUĆUJE UVOĐENJE OPERATORA KOJE ĆEMO SAMI ISPROGRAMIRATI OBAVLJA SE POMOĆU POSEBNIH OPERATORSKIH METODA I OPERATORSKIH PRIJATELJSKIH FUNKCIJA. NPR: Saberi (a,b) MOŽE +(a,b) POZIVAMO a+b. U SKLOPU PROTOTIPA SE NE PIŠE TIP_IME_ARGUMENTI NEGO: TIP OPERATOR +(ARG) – PROGLAŠAVAMO PLUS ZA SABIRANJE PRI POZIVU. SINTAXNA OGRANIČENJA: NE MOGU SE PREKLAPATI OPERATORI (double U OBJEKTNOM) U STANDARDNOM DELU, NE MOGU SE UVODITI NOVI SIMBOLI I PREKLOPLJENI OPERATOR ZADRŽAVA PRIORITET, DODELU I HIJERARHIJU. NE MOGU SE PREKLOPITI TAČKA, :: , ?, sizeof. SINTAXNI POZIV METODE JE SPECIFIČAN Z.m(b) OMOGUĆUJE DA SE IZMENI U NPR Z.+(b) ISTO ŠTO I Z+b. KLASE SE MOGU PODELITI U DVE VRSTE KAD JE REČ O PRAGMATICI (PRIMENI) OPERATOR SE PREKLAPA KAD IMA MATEMATIČKU ULOGU (PRIRODU): 1. **METODSKI ORJENTISANE KLASE** – NEMAJU PREKLOPLJENE OPERATORE OSIM OPERATORA ZA DODELU (=, !=)
I
POSTOJE OPERATORSKI ORJENTISANE KLASE
PREOVLADAVAJU PREKLOPLJENE METODE, SKORO DA NEMA OBIČNIH METODA (NPR KLASA Complex). OPERATORSKA FJA SE PREKLAPA KAD SE MENJA STANJE KLASE. SABIRANJE KOMPLEXNIH BROJEVA JE PREKLOPLJENA PRIJATELJSKA FJA, JER SE NE MENJA STANJE ARGUMENATA. PRAVILA:

PRAVILO OČUVANJA SEMANTIKE
AKO SMO PREKLOPILI OPERATOR ON MORA SADRŽATI OSNOVNE OSOBINE POLAZNOG OPERATORA. NPR: x
=
y ; - IZRAZ VRAĆA VREDNOST DESNE STRANE KADA IZJEDNAČI STANJA x I y. NEKAD TO ZAHTEVA KREIRANJE PRIVREMENOG STEKA, AKO JE OBJEKAT VELIKI, ZBOG VREMENA I ZAUZIMANJA MEMORIJA UGLAVNOM SE ODUSTAJE.

PRAVILO2
UVEK SE TREBA ODLUČITI: METODSKI ILI OPERATORSKI ORJENTISANE KLASE.

PRAVILO3
PLUS UVEK TREBA UVEK ISTO ZNAČITI, KAO I MINUS I SL. SVAKI OBJEKAT SADRŽI JEDAN PODATAK ČLAN: this I SADRŽI ADRESU OBJEKTA (SOPSTVENU ADRESU). OBJEKAT MOŽE SAM SEBE DA DEREFERENCIRA, JER JE this POKAZIVAČ (*this).

PREKLAPANJE OPERATORA DODELE – METODA; PROTOTIP GLASI:

Complex &operator = (const Complex &);

Complex & Complex :: operator = (const Complex &Z)

```
{                                   & referenca zbog izbegavanja konstruktora kopije

    r = Z . r ;

    i =  Z . i;

    return *this;

}
```

PREKLAPANJE RELACIONIH OPERATORA (U C-U KAO I METODE I KAO PRIJATELJSKE FJE). KAO METODA:

```
int Complex :: Complex operator = = (const Complex &Z)

{

   return (r = = Z . r)&&( i = = Z . i);

}
```

KAO PRIJATELJSKA FUNKCIJA:

friend int operator (const Complex &, const Complex &);

realizacija int operator = = (const Complex &Z1, const Complex &Z2)

{

return (Z1 . r = = Z2 . r)&&(Z1 . i = = Z2 . i);

}

OVO ZNAČI a = = b <=> = = (a,b), A NE a . = = (b) ZA a KLASU I b ARGUMENT.

PREKLAPANJE BINARNIH ARITMETIČKIH OPERATORA (OPERATORI SE PREKLAPAJU PRIJATELJSKOM FUNKCIJOM).
PRVI NAČIN
JE PRIVREMENIM OBJEKTOM, A
DRUGI
BEZIMENIM OBJEKTOM (DIREKTNO SE KORISTI KONSTRUKTOR). PRVI NAČIN:

friend Complex operator+(const Complex &, const Complex &);

↑ NE SME DA SE STAVI REFERENCA JER BI SE VRATILA

ADRESA w KOJI SE UNIŠTAVA PO IZVRŠENJU

Complexoperator + (const Complex &Z1, const Complex &Y2)

{

Complex w; ← privremeni objekat

w .r = Z1.r + Z2.r; w .i = Z1.i + Z2.i;

return w; ← to bi bila referenca nepostojećeg objekta

}

DRUGI NAČIN: UMESTO SVEGA U { } :

return Complex (Z1.r + Z2.r, Z1.i + Z2.i);

++ i – SE PREKLAPAJU NA ISTI NAČIN (METODA, JER SE MENJA STANJE i I r OD Z).
PREFIXNI OBLIK:

const Complex &operator++() const zbog bočnih efekata (zbog prioriteta), ako

{ prevodilac može da izvrši neku drugu operaciju

+ + r ; + + i; return *this; u toku ili pre ove operacije; vraća konstantan

} objekat

POSTFIXNI OBLIK:

const Complex &operator++(int k) dodaje se celobrojni operator i služi za

{ razlikovanje liste parametara od prefixnog;

Complex w(r,i); r + +; i + +; vraća originalnu vrednost uz primenu

return w; privremenog objekta

}

MOŽE SE PREKLOPITI I OPERATOR ( ) DA OVJEKAT MOŽE BITI POZVAN SA PARAMETRIMA. [ ] OPERATOR INDEXIRANJA – OD 1 PA NAVIŠE INDEXI, ZBOG P1 MORA SE REALIZOVATI PREKO REFERENCE (JER MOŽE PRIMITI VREDNOST AKO SE NAĐE S LEVE STRANE). DEREFERENCIRANJE MOŽE ALI SE NE PREKLAPA.

13. KONVERZIJA TIPOVA U C++

ISPITIVANJE IZMEĐU OSTALIH KOERCITIVNIH

- AUTOMATSKA

- PRINUDNA - EXPLICITNA

AUTOMATSKA KONVERZIJA TIPOVA ZA STANDARDNE TIPOVE (NE KLASE) IZ PROSTIJIH U SLOŽENIJE I U KONTEXTU FORMALNIH I STVARNIH PARAMETARA (double - int). KOD OBJEKTA PRI PRENOSU KAO STVARNOG PARAMETRA SLUČAJI:

class A ← prvi slučaj

{

A (T t) {…}

};

tip f(A a) {…}

f(Z) poziv nije objekat nego tip (Z) i radi ako se uključi konstruktor

T(Z)

tip g(A a){…} ← drugi slučaj

g(x) x є U, U klasa kojoj pripada x i uz konstruktor A(U&u){…}

EXPLICITNA KONVERZIJA TIPOVA U NEKI OD STANDARDNIH TIPOVA

```
class P {

        operator T ( ) {…}

    };
```

Pp;

(T)p isto što i T(p), ali se ne preporučuje

14. TIPOVI VEZA IZMEĐU KLASA

V1 – VEZA AGREGACIJE, V2 – VEZA NASLEĐIVANJA, V3 – VEZA ASOCIJACIJE, V4 – VEZA ZAVISNOSTI. ODOZGO NADOLE SEMANTIKA OPADA. VEZA AGREGACIJE

MODELUJE ODNOS CELINA – DELOVI (NPR SADRŽAVANJE) PREPOZNAVANJE OPISOM: SADRŽI, SASTOJI SE. VEZA NASLEĐIVANJA JE ODNOS GENERALIZACIJA – SPECIJALIZACIJA. KLJUČNA REČ JE JE JE (NPR PRAVOUGAONIK JE POLIGON). VEZA ASOCIJACIJE JE NAJČEŠĆA. POVEZUJE SEMATIČKI NEZAVISNE KLASE (NPR NASTAVNIK, PREDMET POVEZUJE SE POMOĆU PREDAJE). VEZA ZAVISNOSTI JE NEBITNA, EGZISTENCIJALNA: AKO NEKA KLASA NE MOŽE POSTOJATI SAMOSTALNO.

15. AGREGACIJA

VEZA AGREGACIJE MODELUJE ODNOS CELINA-DEO. PREPOZNAJE SE PO REČI PRIPADA, SADRŽAN_JE_OD, SASTOJI_SE_OD, ... VRSTE AGREGACIJE: **KOMPOZICIJA** (ALTERNATIVNO IME JE SADRŽAVANJE PO VREDNOSTI),
ČISTA
AGREGACIJA
(SADRŽAVANJE PO REFERENCI). PRVA JE NAJČVRŠĆA VEZA I ČESTO JE KOMPONENTA, OTUD ALTERNATIVNO IME. POSTAVLJA OBJEKAT ČLAN. ČISTA SE REALIZUJE POMOĆU POKAZIVAČA. NPR KLASA VALJAK MOŽE SADRŽAVATI POKAZIVAČE NA KLASE KRUG I PRAVOUGAONIK. REALIZACIJA: C++ NE SADRŽI MEHANIZME ZA AGREGACIJU VEĆ GA PROGRAMER REALIZUJE (NPR ZA SLUČAJ NASTAVNA GRUPA→STUDENTI - JEDNOSTRUKO SPREGNUTA LISTA, A UKIDANJEM BI SE UNIŠTILI SAMO POKAZIVAČI, A LISTA BI OSTALA).

16. DEFINICIJA I OSOBINE NASLEĐIVANJA

NASLEĐIVANJE JE PO DEFINICIJI PREUZIMANJE SVIH OSOBINA OD DRUGE KLASE UZ MOGUĆNOST DODAVANJA ČLANOVA I MODIFIKACIJE FUNKCIJA ČLANICA (JAVNE METODE). NASLEĐIVANJE KAO VEZA MODELUJE ODNOS OPŠTE – POJEDINAČNO TJ GENERALIZACIJA – SPECIJALIZACIJA. KLASA A JE PREDAK, OSNOVNA KLASA, KLASA DAVALACA, KLASA B JE KLASA POTOMAK, IZVEDENA ILI KLASA PRIMALACA. n NE PODRAZUMEVA NIKAKVE PROMENE U KODU PRETKA I NIJE NI POTREBAN IZVORNI KOD. OSOBINE n 1. MOGUĆNOST MODIFIKACIJE METODA, 2. REDEFINISANJE, PROŠIRIVANJE KLASE, TEORIJSKI: REDEFINISANJE PODATKA ČLANA IZ PRETKA (NE RADI SE BAŠ ČESTO, NEKI OBJEKTNI C N DOZVOLJAVAJU) TO ZNAČI PODATAK ČLAN MENJA TIP. 3. TRANZITIVNOST A→B, B→C => A→C. 4. VIŠESTRUKO NASLEĐIVANJE (C++ IMA). JEDNOSTRUKIM NASLEĐIVANJEM DOBIJAMO HIJERARHIJU KLASA (STRUKTURA STABLA).

17. DEMETRIN ZAKON I ZAKON SUPSTITUCIJE

PRAVILO KOJE POVEZUJE ENKAPSULACIJU I NASLEĐIVANJE JE **DEMETRIN ZAKON**

(DEMETRA JE BOGINJA PLODNOSTI) KOJI GLASI: METODE DATE KLASE NE SMEJU NI NA KOJI NAČIN DA ZAVISE OD STRUKTURE BILO KOJE DRUGE KLASE OSIM NEPOSREDNOG PRETKA. C NE TREBA DA ULAZI U KLASU A, ALI NASLEĐUJE B KOJA SME.

ZAKON SUPSTITUCIJE (DEFINISALA BARBARA LISHON): AKO ZA SVAKI OBJEKAT s IZ KLASE S POSTOJI OBJEKAT t KLASE T TAKAV DA SE PROIZVOLJAN PROGRAM P DEFINISAN NAD T PONAŠA ISTO KADA SE t ZAMENI SA S TADA JE IZVEDENA KLASA T.

18. REALIZACIJA NASLEĐIVANJA U C++

JEDNOSTRUKO NASLEĐIVANJE – CILJ: ŠTO MANJE SINTAXE. PRIMER:

```
class B: public          A

        {

          protected              jedna od ove 4 opcije;

          private

          (ništa)
```

~

private: .....

protected: ..... imaju pristup svi naslednici po Demetrinom zakonu

public: .....

};

STVARNI NIVO ZAŠTITE MOŽE SE ISKOMBINOVATI (NASLEĐIVANJE JE KOMBINACIJA). POSTOJE 2 PRAVILA I 1 MOGUĆNOST: P1. OPERATOR DODELE SE NE NASLEĐUJE (KONSTRUKTORI I DESTRUKTORI SE NE NASLEĐUJU). P2. KOOPERATIVNOST SE NE NASLEĐUJE (AKO ŽELIMO friend ONDA JE NAVODIMO). M1. METODE SU REDEFINISANE – KVALIFIKOVANJEM A::m( ) (U ZAGRADU IDU PARAMETRI), A KLASA, A m METODA.

19. INKLUZIVNI POLIMORFIZAM U C++

DEF: PROMENLJIVA SE PONAŠA KAO DA PRIPADA VEĆEM BROJU KLASA, RAZLIČITO U RAZLIČITIM KONTEXTIMA (KLASE MORAJU BITI POVEZANE RELACIJOM NASLEĐIVANJA). POTREBNI SU I POKAZIVAČI I VIRTUELNE METODE. PRAVILO: POTOMAK MOŽE ZAMENITI PRETKA, OBRNUTO NE VAŽI. POTOMAK NE MORA BITI

NEPOSREDAN JER JE RELACIJA NASLEĐIVANJA TRANZITIVNA.

Predak Pr, *pPr ; Predak m Po – instanca klase potomak

↑ *pPo – pokazivač na klasu Potomak

Potomak Po, *pPo ; Potomak m

1. Pr = Po; explicitna konverzija tipa ne može

2. *pPr = *pPo;

3. pPr = pPo; kopira se adresa, a dereferenciranje se ne vrši, u stvari vrši se explicitna konverzija tipa tj: pPr = (Predak *)pPo; (ne piše se jer se vrši automatski)

pPr → m (...) - aktivira se metoda iz pretka

POLIMORFIZAM OBJEKATA KAO STVARNIH PARAMETARA: AKO JE FORMALNI PARAMETAR DEFINISAN, KAO PREDAK DEREFERENCIRAN ONDA ĆE SE PONAŠATI KAO PREDAK: tip f(Predak &pr){…}.

20. VIRTUELNE FUNKCIJE U C++

SU TIP FUNKCIJA KOJI OMOGUĆAVA DA SE POKAZIVAČ NA FUNKCIJU PONAŠA POLIMORFNO.

v. Predak k se definiše pozivom m-a k. m

Predak k sadrži adresu k jer k ne postoji

m u to vreme definisanja

Potomak v. Potomak k

pPr = pPo; ← nastaje problem pa se uvode virtuelne funkcije, koje se ponašaju kao

pPr → m (…); virtuelne metode. Naziv: vtabela, sadrži veličinu objekta i adrese

svih virtuelnih metoda. Kada se m pozove obezbeđuje se čitanje k iz

vtabele.

Predak n Ako je n virtuelna metodanjena adresa se čita iz instance

Potomak n klase (ik)tj poziva se donja n

pPr pPo

ik

pPr = pPo;

pPr → n(…)

U PROTOTIPU METODE KAD SE PRVI PUT POJAVLJUJE: virtual tip ime (parametri). U POTOMCIMA SE NE MORA PISATI VIRTUAL NEGO SAMO tip ime (parametri). KADA ĆE METODA BITI PROGLAŠENA VIRTUELNOM? NI JEDNA METODA
NE SME
BITI VIRTUELNA, JER SU VIRTUELNE SPORIJE. AKO POSTOJI I NAJMANJA MOGUĆNOST DA NEKA METODA BUDE REDEFINISANA ONA
SE PROGLAŠAVA
VIRTUELNOM. NPR FIGURA: KRUG, TRPUGAO, PRAVOUGAONIK, KVADRAT I NJIHOVE POVRŠINE.

Figura *pF; Virtuelna metoda Površina se prvi put definiše u Figurida bi bila

pF - > Površina ( ) posle redefinisana, a naziva se: virtual double Površina ( ) = 0;

0 joj je početna adresa

APSTRAKTNE KLASE SE NE MOGU INSTANCIRATI. KONSTRUKTOR UPISUJE ADRESE U INSTANCU KLASE. DESTRUKTOR SE POZIVA IMPLICITNO (AUTOMATSKI), DESTRUKTOR MORA BITI VIRTUALAN.

21. VIŠESTRUKO NASLEĐIVANJE

A m B m ... dve ili više klasa nasleđuje jedna

X m

VIŠESTRUKO NASLEĐIVANJE (VN) NE POSTOJI U C++ KAO NI MEHANIZAM. class X: αA, βB {…}α-PRIVATE, PUBLIC, PROTECTED, NIŠTA. m TREBA REDEFINISATI: UZETI IZ A, UZETI IZ B: A .. m(...), B .. m(...)

22. GENERIČKE KLASE

GENERIČKI APSTRAKTNI TIPOVI PODATAKA ZA PARAMETRE KLASE IMAJU KLASE. TEMPLATE (GENERIČKE KLASE) REŠAVA PRISTUP BINARNOM STABLU PODATAKA, STEKU, LISTI. NPR1. PODACI STEKA SU TIPA T (TIP SE ZADAJE INSTANCIRANJEM KLASE). NPR2.

template <class T_1, class T_2, …, class T_n>

class A {

T_1, …, T_2

}

AKO JE class A PARAMETRIZOVANA ONDA MORA IMATI KONSTRUKTOR; T_1, …, T_n TIPOVI SE MOGU KORISTITI U KLASI. MOGUĆE JE PARAMETRIZOVATI I KONSTANTE:

template <class T_1, class T_2, …, class T_n>

tip A <T_1, T_2, …, T_n> :: f(…){…}

arg_1, ..., arg_n

način1 način2

IME_KLASE<p_1,...,p_n> IZVEDENA KLASA

INSTANCIRANJE OVAKVE KLASE GLASI: X p; A <Z_1, …, Z_n> r ; PRIMER: U OKVIRU STEKA JE NIZ, PARAMETRIZOVANJE KONSTANTI. MAX_TOP DOBIJA VREDNOST U TRENUTKU INSTANCIRANJA:

```
template <class t, unsigned MAX_TOP>

class Stack {

        private:  int top;
```

```
                T S [MAX_TOP+1];

public:     stack ( ) {top = -1;}

                int Empty ( ) const{return top<0;}

                int Full ( ) const{return top = = MAX_TOP;}

                T Top ( ) const {return S[top];}

                void Pop ( ) {top--;}

                void Push (T element) {S [++ top] = element;}

};
```

INSTANCIRANJE (U DRUGOM MODULU):

```
Stack <char, 255> c st;

Stack <int, 63> i st;
```

23. VEZA ASOCIJACIJE

VEZA MEĐUSOBNO SEMANTIČKI NEZAVISNIH KLASA (U UML – UNIFY MODELING LANGUAGE, KOJI SLUŽI ZA OBJEKTNO MODELOVANJE, GRAFIČKI JE).

NASTAVNIK 1..* 0..* PREDMET

PREDAJE

RETKO 4TOG REDA

24. VEZA ZAVISNOSTI

<<USE>>

A - - - - - - - - - - >B A NE MOŽE BEZ B

1. PROBLEMI KOMPATIBILNOSTI, KONTINUITETA I PONOVNOG KORIŠĆENJA

OBJEKTNA METODOLOGIJA NASTALA JE NEGIRANJEM PROCEDURALNE PARADIGME, KOJE JE NASTALO NEGIRANJEM KOMPOZITNOG PROGRAMIRANJA (FORTRAN I BASIC NE MOGU DA SE PIŠU BEZ SKOKOVA). NEREŠEN JE PROBLEM IZMEĐU ALGORITMA I STRUKTURE PODATAKA. STRUKTURNO PROGRAMIRANJE JE U CELOSTI ORJENTISANO. OSTAJU SAMO PROBLEMI KOMPATIBILNOSTI, KONTINUITETA I TZV PONOVNOG KORIŠĆENJA SOFTVERA. **PROBLEM KOMPATIBILNOSTI**

RAZBIJAMO NA POTPROBLEME KOJE REŠAVAMO U SEKVENCI, A SEKVENSU UVEK ZADRŽAVAMO DA NE BI DOŠLO DO SKOKOVA. ALGORITMI SU KOMPATIBILNI, ALI STRUKTURE NISU.

PROBLEM KONTINUITETA

ODNOSI SE NA MODIFIKACIJU SOFTVERA – VERZIJE (TO JE MODIFIKOVANA VERZIJA PROGRAMA). PROGRAM NE MOŽEMO POSMATRATI KAO STATIČKI VEĆ GA POSMATRAMO KAO DINAMIČKU KATEGORIJU. MODIFIKACIJA SE ODNOSI NA MODIFIKACIJU FUNKCIJA PROGRAMA. MODIFIKACIJA SE NAJVIŠE SVODI NA DODAVANJE FUNKCIONALNOSTI. PROCEDURNO MODELIRANJE PROGRAMA – NAJTEŽE JE DODATI FUNKCIJU. ONE SE DODAJU NA SAMOM VRHU.

SOFTWARE REUSE

BEZ PONOVNOG KORIŠĆENJA SOFTVERA NEMA PROGRAMIRANJA. STRUKTURIRANO PROGRAMIRANJE NE PODRŽAVA PONOVNO KORIŠĆENJE. DA BI REUSE MOGLO BITI UPOTREBLJENO MORAMO IMATI IZVORNI KOD – ON SE NE MOŽE ISEĆI IZ MAŠINSKOG KODA. UVEK POLAZIMO OD ČINJENICE DA IZVORNIM KODOM NE RASPOLAŽEMO.

2. ODNOS ALGORITMA I STRUKTURE PODATAKA

UNITI SU SE POJAVILI ’73-’74. U OKVIRU STRUKTURIRANOG PROGRAMIRANJA. PARNAS JE DEFINISAO ŠTA JEDAN MODUL TREBA DA SADRŽI. MODULI SU ODSTUPANJE OD STRUKTURIRANOG PROGRAMIRANJA. OBJEKTNO PROGRAMIRANJE JE REŠILO SVE OVE PROBLEME TAKO ŠTO JE KOMPATIBILNOST SKINULO SA DNEVNOG REDA, JER NEMA JEDINSTVENOG ALGORITMA I JEDINSTVENE STRUKTURE. ONI SU SPAKOVANI U OBJEKTE. BILO KAKVA PROMENA U KLASI ALGORITMA NEMA VEZE SA KLASOM STRUKTURE PODATAKA. DO KONTINUITETA NE MOŽE DOĆI JER ALGORITAM I STRUKTURA NEMAJU NIKAKVE EZE.

OBJEKTNA METODOLOGIJA ZASNOVANA NA NEKOLIKO OSNOVNIH PRINCIPA I TEHNIKA: PRINCIPI : APSTRAKCIJA I SKRIVANJE INFORMACIJA, TEHNIKE KOJIMA SE SPROVODE TA 2 PRINCIPA: INKAPSULACIJA I MODULARNOST, HIJERARHIJA I POLIMORFIZAM (UMESTO HIJERARHIJE BI BOLJE BILO REĆI VEZE IZMEĐU KLASA). **AP STRAKCIJA**
JE RAZDVAJANJE BITNOG OD NEBITNOG U DOMENU PROBLEMA.
SKRIVANJE INFORMACIJA
NASTAVLJA SE NA APSTRAKCIJU I KAŽE DA JE ONO ŠTO JE BITNO TREBA DA BUDE NEDOSTUPNO.
INKAPSULACIJA
JE TEHNIKA ZA OBJEDINJAVANJE DESKRIPTIVNIH I DINAMIČKIH OSOBINA U CELINU.
MODULARNOST
JE RAZBIJANJE PROGRAMA NA MANJE LOGIČKE CELINE KOJE IMAJU ČVRSTU LOGIČKU POVEZANOST.
HIJERARHIJA
MORA DA POSTOJI JER SU OBJEKTNI PROGRAMI SKUP RAZLIČITIH POVEZANIH KLASA.
POLIMORFIZAM
– MOGUĆNOST DA SE ISTA KATEGORIJA RAZLIČITO PONAŠA U RAZLIČITIM USLOVIMA, ZAVISNO OD KONTEXTA.

3. DEFINICIJA KLASE I OBJEKTA. SVOJSTVA OBJEKATA

C++ JE NASTAO POČETKOM 80-TIH GODINA, KASNIJE OD PASKALA I NEJASNE STVARI IZ PASKALA SU OVDE RAŠČIŠĆENE. RAZVOJ OBJEKTNOG SISTEMA POSMATRAMO NA TRI NIVOA. **OBJEKTIVNA REALNOST** – TU EGZISTIRAJU ENTITETI KOJI IMAJU MNOŠTVO OSOBINA. DA BI IZBEGLI TE PROBLEME: DOMEN PROBLEMA – ENTITETI SE UJEDINJUJU U KLASE ENTITETA S TIM ŠTO SE NEKE OSOBINE STAVLJAJU U DRUGI PLAN. NA OSNOVU BITNIH OSOBINA UDRUŽUJU SE U KLASE ENTITETA. ENTITET JE REALAN A KLASA JE APSTRAKTNA. ENTITET JE PO FILOSOFSKOJ DEFINICIJI POSTOJANJE NEČEGA. MODEL – KADA ENTITET TREBA MODELOVATI ONE OSOBINE KOJE SMO ZANEMARILI ODBACUJEMO, JER JE MODEL HOMOMORFAN TJ UPROŠĆENA SLIKA ORIGINALA. OBJEKAT JE PO SVOJOJ PRIRODI MODEL ENTITETA. SVAKI OBJEKAT IMA IDENTITET, STANJE I PONAŠANJE (STANJE I PONAŠANJE SU DVE MEĐUSOBNO USLOVLJENE OSOBINE). NA OSNOVU OVIH OSOBINA OBJEKTI SE IZDVAJAJU OD OSTALIH ENTITETA. **IDENTITET** JE KARAKTERISTIKA DA SE RAZLIKUJE OD SVIH OSTALIH OBJEKATA.

STANJE

JE DEO PROŠLOSTI I SADAŠNJOST OBJEKTA NEOPHODNI ZA ODREĐIVANJE BUDUĆNOSTI.

PONAŠANJE

JE NAČIN NA KOJI OBJEKAT REAGUJE NE POBUDU. SINONIM ZA OBJEKAT JE

INSTANCA KLASE

. OBJEKAT U SEBI SADRŽI DESKRIPTIVNE OSOBINE KAO I DRUGE OBJEKTE (NPR VOZ JE OBJEKAT I SASTOJI SE OD VAGONA KOJI SU OVJEKTI). DESKRIPTI OSOBINE U C++ ZOVU SE

PODACI ČLANOVI,

A DRUGI OBJEKTI OBJEKTI ČLANOVI. NJIHOV UREĐEN SKUP SE ZOVE STRUKTURA OBJEKTA PODATAKA. KLASA JE PO DEFINICIJI: MODEL KLASE ENTITETA KOJI OBUHVATA OBJEKTE SA ISTOM STRUKTUROM I ISTIM PONAŠANJEM.

4. DEKLARISANJE KLASE U C++

U PASKALU: Tzpe Ime = object. DEKLARISANJE KLASE U C++ VRŠI SE POSEBNOM NAREDBOM

```
class ime_klase {

                definicije_podataka_članova

                definicije_objekata_članova

                definicije/deklaracije funkcija_članica

               };
```

PRIMER

```
class Point

{

    private: double x, y;

    public: void SetPoint (double xx, double yy) { x = xx; y = yy; }

            double GetX ( )  (u zagradi ne mora void) {return x;}
```

```
        double GetY ( )  (u zagradi ne mora void) {return y;}

        double Distance ( );

}

inline double Point :: Distance ( ) {return sqrt (x*x+y*y);}
```

INLINE METODE (FUNKCIJE ČLANICE) KARAKTERIŠU SE NAČINOM PREVOĐENJA U MAŠINSKI JEZIK. NA MESTU POZIVA NALAZI SE CELA FUNKCIJA, CILJ JE BRZINA. KAKO SE DEFINIŠU OBJEKTI ILI INSTANCE KLASE? Point p;

```
        double a, b, c;

        pristup metodama:    p.SetPoint(1.3,-0.8);

                             a = p.x;

                             b = p.GetX( );

                             q = p;

                             c = p.Distance( ) +3;
```

U C++ SE PODRAZUMEVA PRIVATE, ZA RAZLIKU OD PASKALA (PUBLIC).