**„Thinking in C++”, Bruce Eckel**

**3: The C in C++**

**Ex. 3**

string ram;

int main() {

int k;

while(true) {

cin >> ram;

if (ram== "Bericht") k=1;

else if (ram=="Ankundigung") k=-3;

else if (ram=="Auftrag") k= -7;

else if (ram =="quitter"){

cout << "Ausgang! \n"; break;}

else k=0;

switch (k){

case 1: cout << "Bericht"<<endl; break;

case -3: cout << "Ankundigung ausgewahlt \n"; break;

case -7: cout << "Auftrag ausgewahlt \n"; break;

default: cout << "Keiner gueltige Wahl \n"; break;

}

} return 0; }

**Ex. 4**

int main() {

char c; // To hold response

int k=0;

while(k==0) {

cout << "MAIN MENU:" << endl;

cout << "l: left, r: right, q: quit -> ";

cin >> c;

switch (c) {

case 'q': k=-1; break; // Out of "while(1)"

case 'l': {

cout << "LEFT MENU:" << endl;

cout << "select a or b: ";

cin >> c;

switch(c) {

case 'a':

cout << "you chose 'a'" << endl;

break; // Back to main menu

case 'b':

cout << "you chose 'b'" << endl;

break; // Back to main menu

default:

cout << "you didn't choose a or b!"

<< endl;

break; // Back to main menu

}

break; }

case 'r': {

cout << "RIGHT MENU:" << endl;

cout << "select c or d: ";

cin >> c;

switch(c) {

case 'c': cout << "you chose 'c'" << endl;

break; // Back to main menu

case 'd':

cout << "you chose 'd'" << endl;

break; // Back to main menu

default: cout << "you didn't choose c or d!"

<< endl;

break; // Back to main menu

}break;

} default: cout << "you must type l or r or q!" << endl;

break;}

} cout << "quitting menu..." << endl;

**Ex. 6**

short int dosh;

long int dol; float cat; char bird; long double fish;

void f(int pet) {

cout << "pet id number: " << pet << endl; }

int main() { int i, j, k;

cout << "f(): " << (long)&f << endl;

cout << "dosh: " << (long)&dosh << endl;

cout << "dol: " << (long)&dol << endl;

cout << "cat: " << (long)&cat << endl;

cout << "bird: " << (long)&bird << endl;

cout << "fish: " << (long)&fish << endl;

cout << "i: " << (long)&i << endl;

cout << "j: " << (long)&j << endl;

cout << "k: " << (long)&k << endl;

**Ex. 7**

string& refa(string& san) {

san = "Yokosan!";

return san; }

string pola( string\* sut) {

\*sut = "Kirukara!";

return \*sut; }

int main() {string ingi="";

cout << ingi << endl;

cout << pola(&ingi) << endl;

cout << ingi << endl;

cout << refa(ingi);

cout << ingi << endl;

**Ex. 9**

void func() {

static int i = 0;

for(int x = 0; x < 10; x++)

cout << "i = " << ++i << endl; }

**Ex. 11**

int main() {

double i, j;

**Ex. 12**

cout << "i && j is " << (i and j) << endl;

cout << "i || j is " << (i or j) << endl;

cout << " (i < 10) && (j < 10) is "

<< ((i < 10) and (j < 10)) << endl;

**Ex. 13**

**Ex. 14**

int main() { int i; while(true) {

cout << "type a number and 'Enter'" << endl;

cin >> i;

i > 5? cout << "It's greater than 5" << endl

: i < 5?

cout << "It's less than 5 " << endl

: cout << "It's equal to 5 " << endl;

cout << "type a number and 'Enter'" << endl;

cin >> i;

i < 10 ?

i > 5 ? // "if" is just another statement

cout << "5 < i < 10" << endl

: cout << "i <= 5" << endl // Matches "if(i < 10)"

:

cout << "i >= 10" << endl; }

**Ch. 4 Data abstraction**

**Exercise 13**

#include <iostream>

using namespace std;

struct Video{

string RenterName;

string TapeTitle, RentDate, ReturnDate;

string DurationMinutes;

int RentsNo;

string RentState;

void Initialize(string ClientName, string Title, string Date, string R\_Date, string Duration, int Re\_no, string State);

void print();

};

void Video::Initialize(string ClientName, string Title, string Ren\_Date, string Ret\_Date, string Duration, int Re\_no, string State){

RenterName = ClientName;

TapeTitle = Title;

RentDate = Ren\_Date;

ReturnDate = Ret\_Date;

DurationMinutes = Duration;

RentsNo = Re\_no;

RentState = State;

}

void Video::print( ){

cout << "Holder Name: " <<

RenterName << endl;

cout << "Film Title: " <<

TapeTitle << endl;

cout << "Rent Date: " <<

RentDate << endl;

cout << "Return Date: " <<

ReturnDate << endl;

cout << "Duration: " <<

DurationMinutes << endl;

cout << "Number of Rents: " <<

RentsNo << endl;

cout << "Rent State: " <<

RentState << "\n" << endl;

}

int main()

{

Video Tape1, Tape2;

Tape1.Initialize("Alvaro", "Jurassic Park II",

"23/5/1995", "7/6/1995", "95'", 5, "Out of rent"

);

Tape2.Initialize("Margot", "Twilight",

"11/3/1994", "7/4/1994", "98'", 21, "In ret!");

Tape1.print();

Tape2.print();

// cout << Tape1.TapeTitle;

}

**Exercise 18**

#include <iostream>

using namespace std;

struct Klas{

char\* storage;

char\* X (char\* Ch);

char\* fetch(int k);

};

char\* Klas:: X(char\* Ch){

char\* bar = new char[5];

for(int k = 0; k<5; k++)

bar[k] = Ch[k];

storage = bar;

return storage;

}

char\* Klas::fetch(int k){

if (k>= 5)

return 0;

return &storage[k];

}

int main()

{

Klas TVR;

char Ch[] = {'j', 'e', 's','r','y'};

for (int w = 0; w< 5; w++)

char\* Duo = TVR.X(Ch);

int k =0;

char\* cp;

while((cp = TVR.fetch(k++)) != 0)

cout << k << "\t cp " << cp << endl;

cout << TVR.fetch(3);

delete []TVR.storage;

cout << "\n After deletion of storage: " << TVR.fetch(1);

//char\* Tre = TVR.X(Duo);

/\*

char ch[] = {'j', 'e', 's','r','y'};

char\* cp = ch;

int k = 0;

for (int w = 0; w< 5; w++){

// cout << ch[w] <<endl;

cout << cp[w] << endl;

}\*/

return 0;

}

P.285-286

#include <iostream>

#include <cstring>

using namespace std;

const int grose = 20;

struct Hoelderling{

public:

void initialize();

struct Pointer;

friend Pointer;

struct Pointer{

void initialize(Hoelderling\* h);

void next();

void previous();

void top();

void end();

int read();

void set(int i);

private:

Hoelderling\* h;

int\* p;

};

private:

int a[grose];

};

void Hoelderling::initialize(){

memset(a,0, grose\*sizeof(int));

}

void Hoelderling::Pointer::initialize(Hoelderling\* rv){

h = rv;

p = rv->a;

}

void Hoelderling::Pointer::next(){

if(p< &(h->a[grose-1])) p++;

}

void Hoelderling::Pointer::previous(){

if(p > &(h->a[0])) p--;

}

void Hoelderling::Pointer::top(){

p = &(h->a[0]);

}

void Hoelderling::Pointer::end(){

p = &(h->a[grose-1]);

}

int Hoelderling::Pointer::read(){

return \*p;

}

void Hoelderling::Pointer::set(int i){

\*p = i;

}

int main(){

Hoelderling h;

Hoelderling::Pointer h1, h2;

int i;

h.initialize();

h1.initialize(&h);

h2.initialize(&h);

for(i = 1; i<= grose; i++){

h1.set(i);

h1.next();

}

h1.top();

h2.end();

for(i = 0; i <= grose; i++){

cout << "h1 = " << h1.read()

<< ",\t h2 = " << h2.read() << endl;

h1.next();

h2.previous();

}

}

**Ch.6 Initialize & Clean-up**

#include <iostream>

using namespace std;

class Dielectric

{

public: Dielectric();

int Objk1;

~Dielectric();

};

Dielectric::Dielectric()

{

Objk1 = 2;

cout<< "\n Constructor invoked! Objk1 = " <<

Objk1 << endl;

}

Dielectric::~Dielectric(){

cout<< "\n Destructor called! Objk1 = " << Objk1;

}

int main()

{

{

cout<<"\n Vor Objekt Schaffen."<< endl;

Dielectric fn;

cout<<"\n Nach Objekt Beschaffen."<< endl;

for (int k=1; k<3; k++){

goto Salto1;

cout<< k << endl; //Niets ausgedrückt.

}

}

Salto1:

cout<<"\n Nach Objekt Zerstörung."<< endl;

//Exerscice 5

int ghe = 0;

int a = 0;

for(ghe = 0; ghe<11; ghe++)

cout << a++ << "\t";

cout <<"\n" << &ghe <<endl;

int b = 0;

for (int ghe =0; ghe<11; ghe++){

if (ghe == 6)

break;

cout << b++ << "\t" << endl;

}

int\* r = &ghe;

cout <<"\n"<< r<<"\t" << \*r;

}

**//Exercise 7**

int main()

{

double vetor[3] = {7.0};

for(int i= 0; i<3; i++)

cout << vetor[i] << "\n" <<endl;

cout<< " The size is: "<< sizeof(vetor) << endl;

double vetor2[] = {2.1, 3.0, 1.0};

for(int i= 0; i<3; i++)

cout << "\n"<< vetor2[i] <<endl;

cout<< "The size is: "<<sizeof(vetor2);

return 0;

}

**Exercise 9**

#include <iostream>

using namespace std;

class Dielectric

{

public: Dielectric(int a);

int Objk1;

void Funkcja1(int Objk1);

~Dielectric();

};

Dielectric::Dielectric(int a)

{

Objk1 = a;

cout<< "\n Constructor invoked! Objk1 = "

<< Objk1 << endl;

}

Dielectric::~Dielectric(){

cout<< "\n Destructor called! Objk1 = " << Objk1<< endl;

}

void Dielectric::Funkcja1(int Objk1)

{

cout <<"\n To jest Funkcja1! a = "<<Objk1;

}

int main()

{

Dielectric fn[] = {Dielectric(1), Dielectric(2), Dielectric(3)};

for (int i = 0; i<3; i++)

fn[i].Funkcja1(fn[i].Objk1); //

cout<< "\n "<< sizeof(fn)<< endl;

return 0;

}

**7. Function Overloading**

**Exercise 1**

#include <iostream>

#include <fstream>

#include <string>

#include <cassert>

using namespace std;

class Text{

public:

string Litera;

Text();

Text(string urtu);

string Contenido(string Litera);

~Text();

};

Text::Text(string urtu){

ifstream input;

input.open(urtu);

assert(input);

while (getline(input, Litera))

Litera.c\_str();

}

string Text::Contenido(string Litera){

return Litera;

}

int main()

{

Text Eins;

string ursus;

extern string Litera;

Eins("C:\Proxeiro\_JavaSc");

ursus= Eins.Contenido(Litera);

cout << ursus<< endl;

return 0;

}

**Exercise 2**

#include <iostream>

//#include <string>

using namespace std;

class Bericht{

string hs ="& @";

public:

Bericht(string Poet = "suleva!");

~Bericht();

void print();

void print(string);

};

Bericht::Bericht(string Poet){

Poet = "suleva! " + hs;

cout << Poet << endl;

}

Bericht::~Bericht()

{

cout << "Destruction confirme" << endl;

}

void Bericht::print(){

cout <<"\n" << hs << endl;

}

void Bericht::print(string Tao){

cout << Tao <<" "

<< hs << endl;

}

int main(){

Bericht alto;

alto.print();

alto.print("suleva!");

return 0;

}

Exercise 4

#include <iostream>

using namespace std;

class Klasse{

public:

int Funkci(int a= 0, int = -7, int= -2);

};

int Klasse::Funkci(int a , int er, int x){

int ty= a+ er + x;

return ty;

}

int main()

{

Klasse Rita;

//cout << Rita.Funkci(3,-2,-6);

cout<< Rita.Funkci(3);

cout << endl;

return 0;

}

**Exercise 5**

#include <iostream>

using namespace std;

short Funkcja(short c1, short ){

return (c1);

}

int main()

{

cout<< Funkcja(2,-4);

return 0;

}

Exercises 7, 8, 9, 10 ?

**Exercise 6**

Stash3.h

Class Stash{

... public:

// Stash(int size); This declaration will be removed.

Stash(int size, int initQuantity = 0);

...};

Stash3.cpp

/\* The following def. Will be removed and the next will be retained only.

Stash::Stash(int sz) {

size = sz;

quantity = 0;

next = 0;

storage = 0;

} \*/

Stash::Stash(int sz, int initQuantity) {

size = sz;

quantity = 0;

next = 0;

storage = 0;

inflate(initQuantity);

}

**Exercise 11**

Class Mem{

byte\* mem;

int size;

void ensureMinSize(int minSize);

bool moved(int minsize, int size); //declared before public:

...};

bool Mem::moved(int minsize){

bool call;

if (size < minsize)

call = true;

else call = false;

return call;

}

**9. Inline Functions**

Ex.1 #define F (x) (x+1) // With the gap between F and ( it does not compile.)

Error x cannot be used as function.

#define F(x) (x+1) // Working.

Ex. 2 #define FLOOR(x,b) x>=b?0:1

int main() {

bool k;

int a =0x0f;

k= FLOOR(a bitand 0x0f, 0x07);

cout << k ; //Console for k shows 0.

**Ex. 3** In main instead of BAND(++a) use:

out << "BAND(++a)=" << **BAND(a+1)** << endl;

**Ex. 4-Ex. 5, Ex. 6**

long f1(){

for (int j=0; j< 800; j++)

for (int k=0; k< 800; k++) {

for (int z=0; z< 800; z++)

long sum =j\*k \*z;

}

}

inline long f2(){

for (int j=0; j< 800; j++)

for (int k=0; k< 800; k++) {

for (int z=0; z< 800; z++)

long sum =j\*k \*z;

} }

void print(){

double m = (clock()/CLOCKS\_PER\_SEC);

cout << "\n" << m; }

int main() {

f1();

//f2();

print();

**Ex. 7** class Renatte{

char c[5];

public:

Renatte(char su =' '){

memset(c, su, 5) ; }

void print(){

for(int i =0; i<5; i++)

cout << c[i] << endl; } or alternatively without memset():

struct Renatte{

char c[5];

public:

Renatte(char su){

for (int u=0; u<5; u++)

c[u] = su; }

void print(){

for(int i =0; i<5; i++)

cout << c[i] << endl; }

};

int main() {

Renatte kobi('!');

kobi.print();

**Ex. 8** The lines that changed plus the added keyword inline in front of each function definition.

...public:

Holder();

public:

Pointer(Holder\* h);

inline Holder::Holder() {

memset(a, 0, sz \* sizeof(int)); }...

Holder::Pointer::Pointer(Holder\* rv):h(rv),p(rv->a) { }...

int main() {

Holder h;

Holder::Pointer hp(&h), hp2(&h);

**Ex. 10** enum Hue{

rot= 3, blau, gelb

};

class Farbe{

Hue ha;

public:

Farbe(Hue um):ha(um){}

void holHue() {

cout<< ha << endl; }

void stellHue(Hue y){

ha = y; }

};

**Ex. 11**

enum Hue{

rot= 3, blau, gelb };

struct Farbe{

Hue ha;

public:

Farbe(Hue um):ha(um){}

Hue HolHue() const { //Accessor

return ha;} // cout<< ha << endl;

void stellHue(Hue y) { //Mutator

ha = y; }

};

Ex. 12

**Ex. 13** class Farbe{

int ku;

public:

void feins() {

fzwei(); }

void fzwei(){

cout << ku << endl; }

};

**Ex. 14**

class Alpha{

public:

Alpha() {

cout<< "Alpha"<< endl; } };

class Bravo{ Alpha a4;

public:

Bravo(){ cout<< "Bravo" <<endl; } };

**Ex. 17**

#define TRACE(s) cerr << #s << endl, s

void f(int g){

cout << g << endl;

//return (g-1); }

int main() {

for (int i =0; i<21; i++){

TRACE(f(i));

**Ex. 18**

#define FIELD(a) char\* a##\_string; int a##\_size; int a##\_index

class Rekor{

FIELD(uno);

FIELD(dos);

FIELD(tres);

public:

Rekor():uno\_index(1), dos\_index(2)

, tres\_index(3), uno\_string("hello"), dos\_string("allo"),

tres\_string("woohoo"), uno\_size(3){}

void f(int index);

// cout<< tres\_index << endl; };

void Rekor::f(int index){

switch(index){

case 1: cout<< uno\_string << "\t "

<< uno\_size << endl; break;

case 2: cout<< dos\_string << "\t "

<< dos\_size << endl; break;

case 3: cout<< tres\_string << "\t "

<< tres\_size << endl; break; }

}

Ex. 19

**Ex. 20**

#include <fstream>

int main(int argc, char\* argv[] )

int i=7;

ifstream in(argv[1]);

assure(in, „file name”);

requireArgs(argc, 2);

require(i>= 5 && i<= 10);

**Ex. 21**

#include <fstream>

#define IFOPEN(VAR, NAME)\

Ifstream VAR(NAME);\

assure(VAR, NAME);

int main(int argc, char\* argv[] ) {

IFOPEN(in, argv[0]);

**10. Name Control**

Ex. 1 void fnz(int y[], int );

int main(){

int fo[5];

for (int q = 0; q < 5; q++){

fo[q] = 3\*q;

}

cout << &fo<< endl;

fnz(fo,0);

fnz(fo,5);

fnz(fo,5);

fnz(fo,5);

fnz(fo,0);

fnz(fo,5);

fnz(fo,5);

}

void fnz(int y[], int i= 0 ){

static int\* ew =y;

static int r= 0;

if (i!=0 && (i-r-1)>-1){

cout << ew[r] << endl;

r++;}

else if(i ==0)

cout << ew << endl;

}

**Ex. 2** #include <iostream>

using namespace std;

union Ria{

int y=0 ;

int fnz( bool);

void print();

};

int Ria::fnz( bool ek = false){

//static int y = 0;

if(ek == true) {

int y = 0;

return y; }

else {

y++;

return y;

} // cout << ek << endl;

}

void Ria::print(){

cout << y << endl;

}

int main()

{

Ria v;

v.fnz();

v.fnz();

cout << v.fnz() << endl;

}

Ex. 4

 class Monitor{

    int k;

    public:

    Monitor(int a=0):k(a){}

    void incident();

    void print();

};

void Monitor::incident(){

    ++k;

}

void Monitor::print(){

    cout<< "No of incidents: " << k <<endl;

}

void una(int y){

     static Monitor Esso;

    for(int w =0; w<y; w++)

    Esso.incident();

    Esso.print();

 }

int main()

{ int t;

    cout<< "Enter no of calls: "<< endl;

    cin>> t;

    una(t);

    /\*

    Monitor alto;

    alto.incident();

    alto.incident();

    alto.print();

    alto.incident();

    alto.incident();

    alto.print();\*/

**Ex. 5, Ex.6** class Monitor{

int k;

public:

Monitor(int a=0):k(a){}

void incident();

void decrement();

void print();

// friend Monitor2;

friend void Rico();

}ek;

void Monitor::incident(){

++k;

}

void Monitor::decrement(){

k-=19;

}

void Monitor::print() {

cout<< "No of incidents: " << k <<endl;

}

void una(int y){

static Monitor Esso;

for(int w =0; w<y; w++)

Esso.incident();

Esso.print();

}

struct Monitor2{

Monitor\* r ;

Monitor2(Monitor\* r);

~Monitor2();

};

Monitor2::Monitor2(Monitor\* r):r(&ek){

r->incident();

r->print();

}

Monitor2:: ~Monitor2(){

r->decrement();

r->print();

}

void Rico(){

static Monitor2 arx(&ek);

}

Monitor2 kor(&ek); //Global Monitor2 object.

int main(){

Rico();

return 0;

}

**Ex. 7** The exit() function is not recognized-no cstdlib support.

**Ex.8**

//ofstream out("statdest.out"); // Trace file

class Obj {

char c; // Identifier

public:

Obj(char cc) : c(cc) {

cout << "Obj::Obj() for " << c << endl;

}

~Obj() {

cout << "Obj::~Obj() for " << c << endl;

}

};

Obj a('a'); // Global (static storage)

// Constructor & destructor always called

void f() {

static Obj b('b');

}

void g() {

static Obj c('c');

}

int main() {

cout << "inside main()" << endl;

g();

f(); // Calls static constructor for b

// g() not called

cout << "leaving main()" << endl;

} ///:~

**Ex. 9, 10** ?

**Ex. 11** The two cpp files were not tested!

class Obj {

int c;

public:

Obj(int cc):c(cc){}

void set(int sui);

void print();

};

void Obj::set(int sui){

c= sui;

}

void Obj::print(){

cout << c <<"\n";

}

int main() {

Obj Gh(-9);

Gh.print();

Gh.set(4);

Gh.print();

}

Ex.12,13 The same reason as ex. 11 due to the online compiler.

Ex. 14 //: C08:Volatile.cpp

// From Thinking in C++, 2nd Edition

void Comm::isr() volatile {

flag = 0;

for(int index =0; index<100;){ //This loop added and the cout line.

buf[index++] = byte;

cout << buf[index]<< endl;}

// Wrap to beginning of buffer:

if(index >= bufsize) index = 0; }

**Ex. 16**

namespace jon{

int f1(int i1){

return (i1-=1); }

int f2(int i2){

return i2-=2; }

}

namespace jon{

int f3(int i3){

return i3-=9; }

void f4(int f1);

}

namespace einfach = jon;

void funcj(){

int m\_= einfach::f2(1);

cout << m\_ <<"\n" ; }

void funcj2() {

using namespace einfach;

cout<< f3(3); }

int main() {

funcj();

funcj2();

return 0;

}

**Ex. 17**

**Ex. 18** namespace {

int f1(int i1){

return (i1-=1);} }

int main()

{ cout << f1(-5);

**Ex. 19** namespace Me {

class Us {

friend void you();

};

void you(){

cout << "you des names"<< endl; }

}

//using namespace Me;

void you(int u){

cout << "Another you outside Me: "<< u; }

int main() {

Me::you();

**Ex. 20**

Declaring: int main(){

Integer x; //and the error message is : ‘Integer’ was not declared in this scope

**Ex. 21**

void s() {

//using namespace Math;

using namespace Calculation;

using Math::divide; // divide of Math overrides divide of Calculation

divide(1,2);

}

int main() {

s();

**Ex. 22** namespace Klas{

class tina{

int u;

public:

inline void Juli(int d){u= d;}

inline void print(){cout<< u;}

};

}

namespace tina{

class Klas{

int u2;

public:

inline void tania(int h){u2 = h ;}

inline void print(){cout<< u2;}

}; }

// using namespace Klas;

// using namespace tina;

void funa(){

tina::Klas tery;

tery.tania(-8);

tery.print();

}

int main()

{

funa();

**Ex.23** using tina::Klas ;

Klas tery;

tery.tania(-8);

tery.print(); }

**Ex. 24** namespace zon{

using Bob::Widget;

using Bob::Poppit;

using ::Arm;

using ::Leg;

using ::Head;

using ::Robot;

using ::i;

using ::j;

using ::k; }

namespace rick = zon;

int main()

{using namespace rick;

Leg hua; hua.hol(7);

hua.print(); cout<<j;

**Ex. 25** Does not recongize string type and cout without std namespace!

**Ex. 26** class Widget {

static double qur;

static long ji;

public:

static void print(){

cout << " "<< qur;

}

};

double Widget::qur = -2;

int main()

{

Widget ser;

ser.print();

**Ex. 27** class Widget {

int ras;

public:

Widget():ras(-9){}

void print(){cout << ras; }

};

class gary {

public:

static Widget lucy;

static void f(){

lucy.print(); }

};

Widget gary::lucy;

int main()

{ //either this:

gary sup;

sup.f();

gary::f(); //Or that:

**Ex.28**

class gary {

const int ar[3] = {4, -9, 78 };

static const int ars[];

public:

void meto();

static void meto2();

};

const int gary::ars[] = {

-1, -4 };

void gary::meto(){

for(int j= 0; j< 5; j++)

cout << ar[j]<< endl;

cout << sizeof(ar); }

int main() { gary rinho;

rinho.meto();

Ex. 29

class G{

string ef;

public:

G(string tengo):ef(tengo){}

void print(){

cout<< ef<< endl; }

};

class G2{

static G ulm[];

public:

static void f\_ulm();

// static void f\_rim();

};

G G2::ulm[] = { G("Cheerio"), G("hooray")};

void G2::f\_ulm() {

for (int j=0; j< 2; j++)

ulm[j].print(); }

int main()

{ G2 iso;

iso.f\_ulm();

**Ex. 30**

void funz(){

struct Widget{

int tempo;

Widget(int roko):tempo(roko){}

void print(){cout << tempo<< endl;}

};

Widget Widge(0), Widet(0);

Widget pam[] = {Widge, Widet};

pam[0].print();

pam[1].print(); }

int main()

{ funz();

**Ex. 31**

Copy from Singleton.cpp on p. 399. Substitute Egg with

Printer\_Connection and e with Printer which is the unique object.

**Ex. 32** ,33, 34 Not completed

class Mirror {

Mirror\* pot;

bool ghi;

public:

Mirror():ghi(true), pot(0){}

Mirror(Mirror\* zu):pot(zu), ghi(false){}

bool test(){

if(pot !=0)

return pot->test();

else return pot->ghi; }

};

Mirror Mr1;

extern Mirror Mr1;

Mirror Mr2(&Mr1);