task1/animal.h

```h
#ifndef ANIMAL_H
#define ANIMAL_H

typedef const char* (*PTRFUN)();

typedef struct Animal_vtable {
    PTRFUN greet;
    PTRFUN menu;
} Animal_vtable;

typedef struct Animal {
    const char *name;
    const Animal_vtable *vtable;
} Animal;

void animalPrintGreeting(const Animal *animal);
void animalPrintMenu(const Animal *animal);

#endif
```

task1/animal.c

```c
#include "animal.h"
#include <stdio.h>

void animalPrintGreeting(const Animal *animal) {
    const char *greetMsg = animal->vtable->greet();
    printf("%s pozdravlja: %s\n", animal->name, greetMsg);
    return;
};

void animalPrintMenu(const Animal *animal) {
    const char *menuMsg = animal->vtable->menu();
    printf("%s voli %s\n", animal->name, menuMsg);
    return;
};
```

task1/cat.h

```h
#ifndef CAT_H
#define CAT_H

#include "animal.h"

void constructCat(Animal *cat, const char *name);
Animal *createCat(const char *name);

#endif
```

task1/cat.c

```c
#include "cat.h"
#include <stdlib.h>

static const char *catGreet(void) {
    return "mijau!";
}

```

```c
static const char *catMenu(void) {
    return "konzerviranu tunjevinu";
}

static Animal_vtable cat_vtable = {
    .greet = &catGreet,
    .menu = &catMenu
};

void constructCat(Animal *cat, const char *name) {
    cat->name = name;
    cat->vtable = &cat_vtable;
}

Animal *createCat(const char *name) {
    Animal *cat = malloc(sizeof(Animal));
    constructCat(cat, name);
    return cat;
}
```

task1/dog.h

```c
#ifndef DOG_H
#define DOG_H

#include "animal.h"

void constructDog(Animal *dog, const char *name);
Animal *createDog(const char *name);
Animal *createManyDogs(int n, const char **names);

#endif
```

task1/dog.c

```c
#include "dog.h"
#include <stdlib.h>

static const char *dogGreet(void) {
    return "vau!";
}

static const char *dogMenu(void) {
    return "kuhanu govedinu";
}

static Animal_vtable dog_vtable = {
    .greet = &dogGreet,
    .menu = &dogMenu
};

void constructDog(Animal *dog, const char *name) {
    dog->name = name;
    dog->vtable = &dog_vtable;
}

Animal *createDog(const char *name) {
    Animal *dog = malloc(sizeof(Animal));
    constructDog(dog, name);
```

```
    return dog;
}

Animal *createManyDogs(const int n, const char **names) {
    if (n<1) return NULL;
    Animal *dogs = malloc(n * sizeof(Animal));
    for (int i=0; i<n; i++) {
        constructDog(dogs + i, names[i]);
    }
    return dogs;
}
```

task1/main.c

```
#include "animal.h"
#include "dog.h"
#include "cat.h"
#include <stdlib.h>
#include <stdio.h>

void testAnimals(void) {
    struct Animal *p1 = createDog("Hamlet");
    struct Animal *p2 = createCat("Ofelija");
    struct Animal *p3 = createDog("Polonije");

    animalPrintGreeting(p1);
    animalPrintGreeting(p2);
    animalPrintGreeting(p3);

    animalPrintMenu(p1);
    animalPrintMenu(p2);
    animalPrintMenu(p3);

    free(p1);
    free(p2);
    free(p3);
}


void testAnimalsUsingStack(void) {
    Animal a1;  Animal *p1 = &a1;   constructDog(p1, "Hamlet");
    Animal a2;  Animal *p2 = &a2;   constructCat(p2, "Ofelija");
    Animal a3;  Animal *p3 = &a3;   constructDog(p3, "Polonije");

    animalPrintGreeting(p1);
    animalPrintGreeting(p2);
    animalPrintGreeting(p3);

    animalPrintMenu(p1);
    animalPrintMenu(p2);
    animalPrintMenu(p3);
}

void testCreateManyDogs(void) {
    int n = 5;
    const char *names[] = {"Rex", "Djuro", "Mirko", "Vlado", "Nils"
    };
    Animal *dogs = createManyDogs(n, names);
```

```
    for (int i=0; i<n; i++) {
        printf("Pas broj %d: %s\n", i+1, dogs[i].name);
    }

    free(dogs);
}

int main(void)
{
    testAnimals();
    printf("\n");
    testCreateManyDogs();
    printf("\n");
    testAnimalsUsingStack();
    return 0;
}
```

task2/cpp_code.cpp

```
#include <stdio.h>
#include <stdlib.h>

class Unary_Function
{
private:
    int lower_bound;
    int upper_bound;

public:
    Unary_Function(int lb, int ub) : lower_bound(lb), upper_bound(
    ub){};
    virtual ~Unary_Function() {}
    virtual double value_at(double x) = 0;
    virtual double negative_value_at(double x)
    {
        return -value_at(x);
    }
    void tabulate()
    {
        for (int x = lower_bound; x <= upper_bound; x++)
        {
            printf("f(%d)=%lf\n", x, value_at(x));
        }
    };
    static bool same_functions_for_ints(Unary_Function *f1,
    Unary_Function *f2, double tolerance)
    {
        if (f1->lower_bound != f2->lower_bound)
            return false;
        if (f1->upper_bound != f2->upper_bound)
            return false;
        for (int x = f1->lower_bound; x <= f1->upper_bound; x++)
        {
            double delta = f1->value_at(x) - f2->value_at(x);
            if (delta < 0)
                delta = -delta;
            if (delta > tolerance)
                return false;
```

```
        }
        return true;
    };
};

class Square : public Unary_Function
{
public:
    Square(int lb, int ub) : Unary_Function(lb, ub){};
    virtual double value_at(double x) override
    {
        return x * x;
    };
};

class Linear : public Unary_Function
{
private:
    double a;
    double b;

public:
    Linear(int lb, int ub, double a_coef, double b_coef) :
    Unary_Function(lb, ub), a(a_coef), b(b_coef){};
    virtual double value_at(double x) override
    {
        return a * x + b;
    };
};

int main()
{
    Unary_Function *f1 = new Square(-2, 2);
    f1->tabulate();
    Unary_Function *f2 = new Linear(-2, 2, 5, -2);
    f2->tabulate();
    printf("f1==f2: %s\n", Unary_Function::same_functions_for_ints(
    f1, f2, 1E-6) ? "DA" : "NE");
    printf("neg_val f2(1) = %lf\n", f2->negative_value_at(1.0));
    delete f1;
    delete f2;
    return 0;
}
```

task2/unary_function.h

```
#ifndef UNARY_FUNCTION_H
#define UNARY_FUNCTION_H

#include <stdbool.h>

typedef struct Unary_Function_Vtable Unary_Function_Vtable;
typedef struct Unary_Function Unary_Function;

struct Unary_Function {
    Unary_Function_Vtable *vtbl;
    int lower_bound;
    int upper_bound;
```

```
};

void Unary_Function_init(Unary_Function *self, int lb, int ub);
void Unary_Function_destruct(Unary_Function *self);
void Unary_Function_destruct_impl(Unary_Function *self);

double Unary_Function_value_at(Unary_Function *self, double x);
//double Unary_Function_value_at_impl(Unary_Function *self, double
    x);
double Unary_Function_negative_value_at(Unary_Function *self,
    double x);
double Unary_Function_negative_value_at_impl(Unary_Function *self,
    double x);
void Unary_Function_tabulate(Unary_Function *self);
bool Unary_Function_same_functions_for_ints(Unary_Function *f1,
    Unary_Function *f2, double tolerance);

#endif
```

task2/unary_function.c

```
#include "unary_function.h"
#include <stdlib.h>
#include <stdio.h>

struct Unary_Function_Vtable {
    void (*dstr)(Unary_Function *);
    double (*value_at)(Unary_Function *, double);
    double (*negative_value_at)(Unary_Function *, double);
};

static Unary_Function_Vtable vtbl = {
    .dstr = &Unary_Function_destruct_impl,
    .value_at = NULL,
    .negative_value_at = &Unary_Function_negative_value_at_impl
};

void Unary_Function_init(Unary_Function *self, int lb, int ub) {
    self->vtbl = &vtbl;
    self->lower_bound = lb;
    self->upper_bound = ub;
}

void Unary_Function_destruct(Unary_Function *self) {
    self->vtbl->dstr(self);
}

void Unary_Function_destruct_impl(Unary_Function *self) {

}

double Unary_Function_value_at(Unary_Function *self, double x) {
    return self->vtbl->value_at(self, x);
}

double Unary_Function_negative_value_at(Unary_Function *self,
    double x) {
    return self->vtbl->negative_value_at(self, x);
```

```
}

double Unary_Function_negative_value_at_impl(Unary_Function *self,
    double x) {
    return -Unary_Function_value_at(self, x);
}

void Unary_Function_tabulate(Unary_Function *self) {
    for (int x = self->lower_bound; x <= self->upper_bound; x++) {
        printf("f(%d)=%lf\n", x, Unary_Function_value_at(self, x));
    }
}

bool Unary_Function_same_functions_for_ints(Unary_Function *f1,
    Unary_Function *f2, double tolerance) {
    if (f1->lower_bound != f2->lower_bound)
        return false;
    if (f1->upper_bound != f2->upper_bound)
        return false;
    for (int x = f1->lower_bound; x <= f1->upper_bound; x++) {
        double delta = Unary_Function_value_at(f1, x) -
    Unary_Function_value_at(f2, x);
        if (delta < 0)
            delta = -delta;
        if (delta > tolerance)
            return false;
    }
    return true;
};
```

task2/square.h

```
#ifndef SQUARE_H
#define SQUARE_H

typedef struct Square_Vtable Square_Vtable;
typedef struct Square Square;

struct Square {
    Square_Vtable *vtbl;
    int lower_bound;
    int upper_bound;
};


Square *Square_create(int lb, int ub);  // poziv operatora new
void Square_init(Square *self, int lb, int ub); // konstruktor
void Square_destroy(Square *self);  // poziv operatora delete

double Square_value_at_impl(Square *self, double x);


#endif
```

task2/square.c

```
#include "square.h"
#include "unary_function.h"
#include <stdlib.h>
```

```

struct Square_Vtable {
    void (*dstr)(Unary_Function *);
    double (*value_at)(Square *, double);
    double (*negative_value_at)(Unary_Function *, double);
};

static Square_Vtable vtbl = {
    .dstr = &Unary_Function_destruct_impl,
    .value_at = &Square_value_at_impl,
    .negative_value_at = &Unary_Function_negative_value_at_impl
};

Square *Square_create(int lb, int ub) {
    Square *sq = malloc(sizeof(Square));
    Square_init(sq, lb, ub);
    return sq;
}

void Square_init(Square *self, int lb, int ub) {
    Unary_Function_init((Unary_Function *)self, lb, ub);
    self->vtbl = &vtbl;
}

void Square_destroy(Square *self) {
    Unary_Function_destruct((Unary_Function *)self);
    free(self);
};

double Square_value_at_impl(Square *self, double x) {
    return x*x;
};
```

task2/linear.h

```
#ifndef LINEAR_H
#define LINEAR_H

typedef struct Linear_Vtable Linear_Vtable;
typedef struct Linear Linear;

struct Linear {
    Linear_Vtable *vtbl;
    int lower_bound;
    int upper_bound;
    double a;
    double b;
};


Linear *Linear_create(int lb, int ub, double a_coef, double b_coef)
    ;    // poziv operatora new
void Linear_init(Linear *self, int lb, int ub, double a_coef,
    double b_coef);   // konstruktor
void Linear_destroy(Linear *self);  // poziv operatora delete

double Linear_value_at_impl(Linear *self, double x);

```

```
#endif
```

task2/linear.c

```
#include "linear.h"
#include "unary_function.h"
#include <stdlib.h>

struct Linear_Vtable {
    void (*dstr)(Unary_Function *);
    double (*value_at)(Linear *, double);
    double (*negative_value_at)(Unary_Function *, double);
};

Linear_Vtable vtbl = {
    .dstr = &Unary_Function_destruct_impl,
    .value_at = &Linear_value_at_impl,
    .negative_value_at = &Unary_Function_negative_value_at_impl
};

Linear *Linear_create(int lb, int ub, double a_coef, double b_coef) {
    Linear *ln = malloc(sizeof(Linear));
    Linear_init(ln, lb, ub, a_coef, b_coef);
    return ln;
}

void Linear_init(Linear *self, int lb, int ub, double a_coef, double b_coef) {
    Unary_Function_init((Unary_Function *)self, lb, ub);
    self->vtbl = &vtbl;
    self->a = a_coef;
    self->b = b_coef;
}

void Linear_destroy(Linear *self) {
    Unary_Function_destruct((Unary_Function *)self);
    free(self);
}

double Linear_value_at_impl(Linear *self, double x) {
    double val = self->a * x + self->b;
    return val;
}
```

task2/main.c

```
#include <stdio.h>
#include "unary_function.h"
#include "square.h"
#include "linear.h"

int main(void) {
    Square *sq1 = Square_create(-2, 2);
    Unary_Function *f1 = (Unary_Function *) sq1;
    Unary_Function_tabulate(f1);

    Linear *ln1 = Linear_create(-2, 2, 5, -2);
    Unary_Function *f2 = (Unary_Function *) ln1;
```

```
    Unary_Function_tabulate(f2);

    printf("f1==f2: %s\n", Unary_Function_same_functions_for_ints(
    f1, f2, 1E-6) ? "DA" : "NE");
    printf("neg_val f1(1) = %lf\n",
    Unary_Function_negative_value_at(f1, 1.0));
    printf("neg_val f2(1) = %lf\n",
    Unary_Function_negative_value_at(f2, 1.0));

    Square_destroy(sq1);
    Linear_destroy(ln1);
    return 0;
}
```

task3/main.cpp

```
#include <iostream>

class CoolClass
{
public:
    virtual void set(int x) { x_ = x; };
    virtual int get() { return x_; };

private:
    int x_;
};


class PlainOldClass
{
public:
    void set(int x) { x_ = x; };
    int get() { return x_; };

private:
    int x_;
};

int main(void) {
    std::cout << "Size of PlainOldClass: " << sizeof(PlainOldClass)
     << std::endl;
    std::cout << "Size of CoolClass: " << sizeof(CoolClass) << std
    ::endl;
    std::cout << "Pointer size: " << sizeof(void*) << std::endl;
    std::cout << "Int size: " << sizeof(int) << std::endl;
    return 0;
}
```

task4/main.s

```
    .file   "main.cpp"
    .intel_syntax noprefix
    .text
    .section    .text$_ZN9CoolClass3setEi,"x"
    .linkonce discard
    .align 2
    .globl  _ZN9CoolClass3setEi
    .def    _ZN9CoolClass3setEi;    .scl    2;  .type   32; .endef
```

```
    .seh_proc   _ZN9CoolClass3setEi
_ZN9CoolClass3setEi:
.LFB0:
    push    rbp
    .seh_pushreg    rbp
    mov rbp, rsp
    .seh_setframe   rbp, 0
    .seh_endprologue
    mov QWORD PTR 16[rbp], rcx
    mov DWORD PTR 24[rbp], edx
    mov rax, QWORD PTR 16[rbp]
    mov edx, DWORD PTR 24[rbp]
    mov DWORD PTR 8[rax], edx
    nop
    pop rbp
    ret
    .seh_endproc
    .section    .text$_ZN9CoolClass3getEv,"x"
    .linkonce discard
    .align 2
    .globl  _ZN9CoolClass3getEv
    .def    _ZN9CoolClass3getEv;    .scl    2;  .type   32; .endef
    .seh_proc   _ZN9CoolClass3getEv
_ZN9CoolClass3getEv:
.LFB1:
    push    rbp
    .seh_pushreg    rbp
    mov rbp, rsp
    .seh_setframe   rbp, 0
    .seh_endprologue
    mov QWORD PTR 16[rbp], rcx
    mov rax, QWORD PTR 16[rbp]
    mov eax, DWORD PTR 8[rax]
    pop rbp
    ret
    .seh_endproc
    .section    .text$_ZN13PlainOldClass3setEi,"x"
    .linkonce discard
    .align 2
    .globl  _ZN13PlainOldClass3setEi
    .def    _ZN13PlainOldClass3setEi;   .scl    2;  .type   32; .
    endef
    .seh_proc   _ZN13PlainOldClass3setEi
_ZN13PlainOldClass3setEi:
.LFB2:
    push    rbp
    .seh_pushreg    rbp
    mov rbp, rsp
    .seh_setframe   rbp, 0
    .seh_endprologue
    mov QWORD PTR 16[rbp], rcx
    mov DWORD PTR 24[rbp], edx
    mov rax, QWORD PTR 16[rbp]
    mov edx, DWORD PTR 24[rbp]
    mov DWORD PTR [rax], edx
    nop
    pop rbp
```

```
    ret
    .seh_endproc
    .section    .text$_ZN4BaseC2Ev,"x"
    .linkonce discard
    .align 2
    .globl  _ZN4BaseC2Ev
    .def    _ZN4BaseC2Ev;   .scl    2;  .type   32; .endef
    .seh_proc   _ZN4BaseC2Ev
_ZN4BaseC2Ev:
.LFB7:
    push    rbp
    .seh_pushreg    rbp
    mov rbp, rsp
    .seh_setframe   rbp, 0
    .seh_endprologue
    mov QWORD PTR 16[rbp], rcx
    lea rdx, _ZTV4Base[rip+16]
    mov rax, QWORD PTR 16[rbp]
    mov QWORD PTR [rax], rdx
    nop
    pop rbp
    ret
    .seh_endproc
    .section    .text$_ZN9CoolClassC1Ev,"x"
    .linkonce discard
    .align 2
    .globl  _ZN9CoolClassC1Ev
    .def    _ZN9CoolClassC1Ev;  .scl    2;  .type   32; .endef
    .seh_proc   _ZN9CoolClassC1Ev
_ZN9CoolClassC1Ev:
.LFB10:
    push    rbp
    .seh_pushreg    rbp
    mov rbp, rsp
    .seh_setframe   rbp, 0
    sub rsp, 32
    .seh_stackalloc 32
    .seh_endprologue
    mov QWORD PTR 16[rbp], rcx
    mov rax, QWORD PTR 16[rbp]
    mov rcx, rax
    call    _ZN4BaseC2Ev
    lea rdx, _ZTV9CoolClass[rip+16]
    mov rax, QWORD PTR 16[rbp]
    mov QWORD PTR [rax], rdx
    nop
    add rsp, 32
    pop rbp
    ret
    .seh_endproc
    .def    __main; .scl    2;  .type   32; .endef
    .text
    .globl  main
    .def    main;   .scl    2;  .type   32; .endef
    .seh_proc   main
main:
.LFB4:
```

```
    push    rbp
    .seh_pushreg    rbp
    push    rbx
    .seh_pushreg    rbx
    sub rsp, 56
    .seh_stackalloc 56
    lea rbp, 128[rsp]
    .seh_setframe   rbp, 128
    .seh_endprologue
    call    __main
    mov ecx, 16
    call    _Znwy
    mov rbx, rax
    mov rcx, rbx
    call    _ZN9CoolClassC1Ev
    mov QWORD PTR -88[rbp], rbx
    lea rax, -92[rbp]
    mov edx, 42
    mov rcx, rax
    call    _ZN13PlainOldClass3setEi
    mov rax, QWORD PTR -88[rbp]
    mov rax, QWORD PTR [rax]
    mov rax, QWORD PTR [rax]
    mov rcx, QWORD PTR -88[rbp]
    mov edx, 42
    call    rax
    mov eax, 0
    add rsp, 56
    pop rbx
    pop rbp
    ret
    .seh_endproc
    .globl  _ZTV9CoolClass
    .section    .rdata$_ZTV9CoolClass,"dr"
    .linkonce same_size
    .align 8
_ZTV9CoolClass:
    .quad   0
    .quad   _ZTI9CoolClass
    .quad   _ZN9CoolClass3setEi
    .quad   _ZN9CoolClass3getEv
    .globl  _ZTV4Base
    .section    .rdata$_ZTV4Base,"dr"
    .linkonce same_size
    .align 8
_ZTV4Base:
    .quad   0
    .quad   _ZTI4Base
    .quad   __cxa_pure_virtual
    .quad   __cxa_pure_virtual
    .globl  _ZTI9CoolClass
    .section    .rdata$_ZTI9CoolClass,"dr"
    .linkonce same_size
    .align 8
_ZTI9CoolClass:
    .quad   _ZTVN10__cxxabiv120__si_class_type_infoE+16
    .quad   _ZTS9CoolClass
```

```
    .quad   _ZTI4Base
    .globl  _ZTS9CoolClass
    .section    .rdata$_ZTS9CoolClass,"dr"
    .linkonce same_size
    .align 8
_ZTS9CoolClass:
    .ascii "9CoolClass\0"
    .globl  _ZTI4Base
    .section    .rdata$_ZTI4Base,"dr"
    .linkonce same_size
    .align 8
_ZTI4Base:
    .quad   _ZTVN10__cxxabiv117__class_type_infoE+16
    .quad   _ZTS4Base
    .globl  _ZTS4Base
    .section    .rdata$_ZTS4Base,"dr"
    .linkonce same_size
_ZTS4Base:
    .ascii "4Base\0"
    .ident  "GCC: (x86_64-win32-seh-rev0, Built by MinGW-W64
    project) 8.1.0"
    .def    _Znwy;  .scl    2;  .type   32; .endef
    .def    __cxa_pure_virtual; .scl    2;  .type   32; .endef
```

task5/main.cpp

```
#include <iostream>

using namespace std;

class B
{
public:
    virtual int __cdecl prva() = 0;
    virtual int __cdecl druga(int) = 0;
};

class D : public B
{
public:
    virtual int __cdecl prva() { return 42; }
    virtual int __cdecl druga(int x) { return prva() + x; }
};

void solution(B *pb, int x) {
    /*
    int p = pb->prva();
    int d = pb->druga(x);
    cout << p << endl << d << endl;
    */

    typedef int (*funPrvaType)(void *);
    typedef int (*funDrugaType)(void *, int);
    typedef unsigned long long addr_t;

    addr_t vtableAddress = *((addr_t *) pb);
    addr_t *vtable = (addr_t *) vtableAddress;

```

```
33      funPrvaType funPrva = (funPrvaType) vtable[0];
34      funDrugaType funDruga = (funDrugaType) vtable[1];
35
36      int p = funPrva(pb);
37      int d = funDruga(pb, x);
38
39      cout << "Poziv prva(): " << p << '\n';
40      cout << "Poziv druga(" << x << "): " << d << endl;
41
42      return;
43  }
44
45  #define XDEF 7
46
47  int main(int argc, char *argv[]) {
48      int x = XDEF;
49      if (argc >= 2)
50          x = stoi(argv[1]);
51      cout << "x = " << x << '\n';
52
53      D d;
54      B *pb = &d;
55
56      solution(pb, x);
57      return 0;
58  }
```