

task1/animal.h

```
1 #ifndef ANIMAL_H
2 #define ANIMAL_H
3
4 typedef const char* (*PTRFUN)();
5
6 typedef struct Animal_vtable {
7     PTRFUN greet;
8     PTRFUN menu;
9 } Animal_vtable;
10
11 typedef struct Animal {
12     const char *name;
13     const Animal_vtable *vtable;
14 } Animal;
15
16 void animalPrintGreeting(const Animal *animal);
17 void animalPrintMenu(const Animal *animal);
18
19 #endif
```

task1/animal.c

```
1 #include "animal.h"
2 #include <stdio.h>
3
4 void animalPrintGreeting(const Animal *animal) {
5     const char *greetMsg = animal->vtable->greet();
6     printf("%s pozdravlja: %s\n", animal->name, greetMsg);
7     return;
8 };
9
10 void animalPrintMenu(const Animal *animal) {
11     const char *menuMsg = animal->vtable->menu();
12     printf("%s voli %s\n", animal->name, menuMsg);
13     return;
14 };
```

task1/cat.h

```
1 #ifndef CAT_H
2 #define CAT_H
3
4 #include "animal.h"
5
6 void constructCat(Animal *cat, const char *name);
7 Animal *createCat(const char *name);
8
9 #endif
```

task1/cat.c

```
1 #include "cat.h"
2 #include <stdlib.h>
3
4 static const char *catGreet(void) {
5     return "mijau!";
6 }
7
```

```

8 static const char *catMenu(void) {
9     return "konzerviranu tunjevinu";
10 }
11
12 static Animal_vtable cat_vtable = {
13     .greet = &catGreet,
14     .menu = &catMenu
15 };
16
17 void constructCat(Animal *cat, const char *name) {
18     cat->name = name;
19     cat->vtable = &cat_vtable;
20 }
21
22 Animal *createCat(const char *name) {
23     Animal *cat = malloc(sizeof(Animal));
24     constructCat(cat, name);
25     return cat;
26 }

```

task1/dog.h

```

1 #ifndef DOG_H
2 #define DOG_H
3
4 #include "animal.h"
5
6 void constructDog(Animal *dog, const char *name);
7 Animal *createDog(const char *name);
8 Animal *createManyDogs(int n, const char **names);
9
10 #endif

```

task1/dog.c

```

1 #include "dog.h"
2 #include <stdlib.h>
3
4 static const char *dogGreet(void) {
5     return "vau!";
6 }
7
8 static const char *dogMenu(void) {
9     return "kuhanu govedinu";
10 }
11
12 static Animal_vtable dog_vtable = {
13     .greet = &dogGreet,
14     .menu = &dogMenu
15 };
16
17 void constructDog(Animal *dog, const char *name) {
18     dog->name = name;
19     dog->vtable = &dog_vtable;
20 }
21
22 Animal *createDog(const char *name) {
23     Animal *dog = malloc(sizeof(Animal));
24     constructDog(dog, name);

```

```

25     return dog;
26 }
27
28 Animal *createManyDogs(const int n, const char **names) {
29     if (n<1) return NULL;
30     Animal *dogs = malloc(n * sizeof(Animal));
31     for (int i=0; i<n; i++) {
32         constructDog(dogs + i, names[i]);
33     }
34     return dogs;
35 }

```

task1/main.c

```

1  #include "animal.h"
2  #include "dog.h"
3  #include "cat.h"
4  #include <stdlib.h>
5  #include <stdio.h>
6
7  void testAnimals(void) {
8      struct Animal *p1 = createDog("Hamlet");
9      struct Animal *p2 = createCat("Ofelija");
10     struct Animal *p3 = createDog("Polonije");
11
12     animalPrintGreeting(p1);
13     animalPrintGreeting(p2);
14     animalPrintGreeting(p3);
15
16     animalPrintMenu(p1);
17     animalPrintMenu(p2);
18     animalPrintMenu(p3);
19
20     free(p1);
21     free(p2);
22     free(p3);
23 }
24
25
26 void testAnimalsUsingStack(void) {
27     Animal a1;  Animal *p1 = &a1;  constructDog(p1, "Hamlet");
28     Animal a2;  Animal *p2 = &a2;  constructCat(p2, "Ofelija");
29     Animal a3;  Animal *p3 = &a3;  constructDog(p3, "Polonije");
30
31     animalPrintGreeting(p1);
32     animalPrintGreeting(p2);
33     animalPrintGreeting(p3);
34
35     animalPrintMenu(p1);
36     animalPrintMenu(p2);
37     animalPrintMenu(p3);
38 }
39
40 void testCreateManyDogs(void) {
41     int n = 5;
42     const char *names[] = {"Rex", "Djuro", "Mirko", "Vlado", "Nils"};
43     Animal *dogs = createManyDogs(n, names);

```

```

44     for (int i=0; i<n; i++) {
45         printf("Pas broj %d: %s\n", i+1, dogs[i].name);
46     }
47
48     free(dogs);
49 }
50
51 int main(void)
52 {
53     testAnimals();
54     printf("\n");
55     testCreateManyDogs();
56     printf("\n");
57     testAnimalsUsingStack();
58     return 0;
59 }

```

task2/cpp_code.cpp

```

1  #include <stdio.h>
2  #include <stdlib.h>
3
4  class Unary_Function
5  {
6  private:
7      int lower_bound;
8      int upper_bound;
9
10 public:
11     Unary_Function(int lb, int ub) : lower_bound(lb), upper_bound(
12         ub){};
13     virtual ~Unary_Function() {}
14     virtual double value_at(double x) = 0;
15     virtual double negative_value_at(double x)
16     {
17         return -value_at(x);
18     }
19     void tabulate()
20     {
21         for (int x = lower_bound; x <= upper_bound; x++)
22         {
23             printf("f(%d)=%lf\n", x, value_at(x));
24         }
25     };
26     static bool same_functions_for_ints(Unary_Function *f1,
27         Unary_Function *f2, double tolerance)
28     {
29         if (f1->lower_bound != f2->lower_bound)
30             return false;
31         if (f1->upper_bound != f2->upper_bound)
32             return false;
33         for (int x = f1->lower_bound; x <= f1->upper_bound; x++)
34         {
35             double delta = f1->value_at(x) - f2->value_at(x);
36             if (delta < 0)
37                 delta = -delta;
38             if (delta > tolerance)
39                 return false;

```

```

38     }
39     return true;
40 };
41 };
42
43 class Square : public Unary_Function
44 {
45 public:
46     Square(int lb, int ub) : Unary_Function(lb, ub){};
47     virtual double value_at(double x) override
48     {
49         return x * x;
50     };
51 };
52
53 class Linear : public Unary_Function
54 {
55 private:
56     double a;
57     double b;
58
59 public:
60     Linear(int lb, int ub, double a_coef, double b_coef) :
        Unary_Function(lb, ub), a(a_coef), b(b_coef){};
61     virtual double value_at(double x) override
62     {
63         return a * x + b;
64     };
65 };
66
67 int main()
68 {
69     Unary_Function *f1 = new Square(-2, 2);
70     f1->tabulate();
71     Unary_Function *f2 = new Linear(-2, 2, 5, -2);
72     f2->tabulate();
73     printf("f1==f2: %s\n", Unary_Function::same_functions_for_ints(
        f1, f2, 1E-6) ? "DA" : "NE");
74     printf("neg_val f2(1) = %lf\n", f2->negative_value_at(1.0));
75     delete f1;
76     delete f2;
77     return 0;
78 }

```

task2/unary_function.h

```

1 #ifndef UNARY_FUNCTION_H
2 #define UNARY_FUNCTION_H
3
4 #include <stdbool.h>
5
6 typedef struct Unary_Function_Vtable Unary_Function_Vtable;
7 typedef struct Unary_Function Unary_Function;
8
9 struct Unary_Function {
10     Unary_Function_Vtable *vtbl;
11     int lower_bound;
12     int upper_bound;

```

```

13 };
14
15 void Unary_Function_init(Unary_Function *self, int lb, int ub);
16 void Unary_Function_destruct(Unary_Function *self);
17 void Unary_Function_destruct_impl(Unary_Function *self);
18
19 double Unary_Function_value_at(Unary_Function *self, double x);
20 //double Unary_Function_value_at_impl(Unary_Function *self, double
    x);
21 double Unary_Function_negative_value_at(Unary_Function *self,
    double x);
22 double Unary_Function_negative_value_at_impl(Unary_Function *self,
    double x);
23 void Unary_Function_tabulate(Unary_Function *self);
24 bool Unary_Function_same_functions_for_ints(Unary_Function *f1,
    Unary_Function *f2, double tolerance);
25
26 #endif

```

task2/unary_function.c

```

1 #include "unary_function.h"
2 #include <stdlib.h>
3 #include <stdio.h>
4
5 struct Unary_Function_Vtable {
6     void (*dstr)(Unary_Function *);
7     double (*value_at)(Unary_Function *, double);
8     double (*negative_value_at)(Unary_Function *, double);
9 };
10
11 static Unary_Function_Vtable vtbl = {
12     .dstr = &Unary_Function_destruct_impl,
13     .value_at = NULL,
14     .negative_value_at = &Unary_Function_negative_value_at_impl
15 };
16
17 void Unary_Function_init(Unary_Function *self, int lb, int ub) {
18     self->vtbl = &vtbl;
19     self->lower_bound = lb;
20     self->upper_bound = ub;
21 }
22
23 void Unary_Function_destruct(Unary_Function *self) {
24     self->vtbl->dstr(self);
25 }
26
27 void Unary_Function_destruct_impl(Unary_Function *self) {
28
29 }
30
31 double Unary_Function_value_at(Unary_Function *self, double x) {
32     return self->vtbl->value_at(self, x);
33 }
34
35 double Unary_Function_negative_value_at(Unary_Function *self,
    double x) {
36     return self->vtbl->negative_value_at(self, x);

```

```

37 }
38
39 double Unary_Function_negative_value_at_impl(Unary_Function *self,
40     double x) {
41     return -Unary_Function_value_at(self, x);
42 }
43
44 void Unary_Function_tabulate(Unary_Function *self) {
45     for (int x = self->lower_bound; x <= self->upper_bound; x++) {
46         printf("f(%d)=%lf\n", x, Unary_Function_value_at(self, x));
47     }
48 }
49
50 bool Unary_Function_same_functions_for_ints(Unary_Function *f1,
51     Unary_Function *f2, double tolerance) {
52     if (f1->lower_bound != f2->lower_bound)
53         return false;
54     if (f1->upper_bound != f2->upper_bound)
55         return false;
56     for (int x = f1->lower_bound; x <= f1->upper_bound; x++) {
57         double delta = Unary_Function_value_at(f1, x) -
58             Unary_Function_value_at(f2, x);
59         if (delta < 0)
60             delta = -delta;
61         if (delta > tolerance)
62             return false;
63     }
64     return true;
65 };

```

task2/square.h

```

1 #ifndef SQUARE_H
2 #define SQUARE_H
3
4 typedef struct Square_Vtable Square_Vtable;
5 typedef struct Square Square;
6
7 struct Square {
8     Square_Vtable *vtbl;
9     int lower_bound;
10    int upper_bound;
11 };
12
13
14 Square *Square_create(int lb, int ub); // pozitiv operatora new
15 void Square_init(Square *self, int lb, int ub); // konstruktor
16 void Square_destroy(Square *self); // pozitiv operatora delete
17
18 double Square_value_at_impl(Square *self, double x);
19
20
21 #endif

```

task2/square.c

```

1 #include "square.h"
2 #include "unary_function.h"
3 #include <stdlib.h>

```

```

4
5 struct Square_Vtable {
6     void (*dstr)(Unary_Function *);
7     double (*value_at)(Square *, double);
8     double (*negative_value_at)(Unary_Function *, double);
9 };
10
11 static Square_Vtable vtbl = {
12     .dstr = &Unary_Function_destruct_impl,
13     .value_at = &Square_value_at_impl,
14     .negative_value_at = &Unary_Function_negative_value_at_impl
15 };
16
17 Square *Square_create(int lb, int ub) {
18     Square *sq = malloc(sizeof(Square));
19     Square_init(sq, lb, ub);
20     return sq;
21 }
22
23 void Square_init(Square *self, int lb, int ub) {
24     Unary_Function_init((Unary_Function *)self, lb, ub);
25     self->vtbl = &vtbl;
26 }
27
28 void Square_destroy(Square *self) {
29     Unary_Function_destruct((Unary_Function *)self);
30     free(self);
31 };
32
33 double Square_value_at_impl(Square *self, double x) {
34     return x*x;
35 };

```

task2/linear.h

```

1 #ifndef LINEAR_H
2 #define LINEAR_H
3
4 typedef struct Linear_Vtable Linear_Vtable;
5 typedef struct Linear Linear;
6
7 struct Linear {
8     Linear_Vtable *vtbl;
9     int lower_bound;
10    int upper_bound;
11    double a;
12    double b;
13 };
14
15
16 Linear *Linear_create(int lb, int ub, double a_coef, double b_coef)
17 ; // pozitiv operatora new
18 void Linear_init(Linear *self, int lb, int ub, double a_coef,
19 double b_coef); // konstruktor
20 void Linear_destroy(Linear *self); // pozitiv operatora delete
21
22 double Linear_value_at_impl(Linear *self, double x);

```



```
22 #endif
```

task2/linear.c

```
1 #include "linear.h"
2 #include "unary_function.h"
3 #include <stdlib.h>
4
5 struct Linear_Vtable {
6     void (*dstr)(Unary_Function *);
7     double (*value_at)(Linear *, double);
8     double (*negative_value_at)(Unary_Function *, double);
9 };
10
11 Linear_Vtable vtbl = {
12     .dstr = &Unary_Function_destruct_impl,
13     .value_at = &Linear_value_at_impl,
14     .negative_value_at = &Unary_Function_negative_value_at_impl
15 };
16
17 Linear *Linear_create(int lb, int ub, double a_coef, double b_coef)
18 {
19     Linear *ln = malloc(sizeof(Linear));
20     Linear_init(ln, lb, ub, a_coef, b_coef);
21     return ln;
22 }
23
24 void Linear_init(Linear *self, int lb, int ub, double a_coef,
25                 double b_coef) {
26     Unary_Function_init((Unary_Function *)self, lb, ub);
27     self->vtbl = &vtbl;
28     self->a = a_coef;
29     self->b = b_coef;
30 }
31
32 void Linear_destroy(Linear *self) {
33     Unary_Function_destruct((Unary_Function *)self);
34     free(self);
35 }
36
37 double Linear_value_at_impl(Linear *self, double x) {
38     double val = self->a * x + self->b;
39     return val;
40 }
```

task2/main.c

```
1 #include <stdio.h>
2 #include "unary_function.h"
3 #include "square.h"
4 #include "linear.h"
5
6 int main(void) {
7     Square *sq1 = Square_create(-2, 2);
8     Unary_Function *f1 = (Unary_Function *) sq1;
9     Unary_Function_tabulate(f1);
10
11     Linear *ln1 = Linear_create(-2, 2, 5, -2);
12     Unary_Function *f2 = (Unary_Function *) ln1;
```

```

13     Unary_Function_tabulate(f2);
14
15     printf("f1==f2: %s\n", Unary_Function_same_functions_for_ints(
16         f1, f2, 1E-6) ? "DA" : "NE");
17     printf("neg_val f1(1) = %lf\n",
18         Unary_Function_negative_value_at(f1, 1.0));
19     printf("neg_val f2(1) = %lf\n",
20         Unary_Function_negative_value_at(f2, 1.0));
21
22     Square_destroy(sq1);
23     Linear_destroy(ln1);
24     return 0;
25 }

```

task3/main.cpp

```

1  #include <iostream>
2
3  class CoolClass
4  {
5  public:
6      virtual void set(int x) { x_ = x; };
7      virtual int get() { return x_; };
8
9  private:
10     int x_;
11 };
12
13
14 class PlainOldClass
15 {
16 public:
17     void set(int x) { x_ = x; };
18     int get() { return x_; };
19
20 private:
21     int x_;
22 };
23
24 int main(void) {
25     std::cout << "Size of PlainOldClass: " << sizeof(PlainOldClass)
26         << std::endl;
27     std::cout << "Size of CoolClass: " << sizeof(CoolClass) << std
28         ::endl;
29     std::cout << "Pointer size: " << sizeof(void*) << std::endl;
30     std::cout << "Int size: " << sizeof(int) << std::endl;
31     return 0;
32 }

```

task4/main.s

```

1  .file    "main.cpp"
2  .intel_syntax noprefix
3  .text
4  .section .text$_ZN9CoolClass3setEi,"x"
5  .linkonce discard
6  .align 2
7  .globl _ZN9CoolClass3setEi
8  .def     _ZN9CoolClass3setEi;    .scl     2;    .type    32; .endef

```

```

9      .seh_proc    _ZN9CoolClass3setEi
10 _ZN9CoolClass3setEi:
11 .LFB0:
12     push    rbp
13     .seh_pushreg    rbp
14     mov rbp, rsp
15     .seh_setframe    rbp, 0
16     .seh_endprologue
17     mov QWORD PTR 16[rbp], rcx
18     mov DWORD PTR 24[rbp], edx
19     mov rax, QWORD PTR 16[rbp]
20     mov edx, DWORD PTR 24[rbp]
21     mov DWORD PTR 8[rax], edx
22     nop
23     pop rbp
24     ret
25     .seh_endproc
26     .section      .text$_ZN9CoolClass3getEv,"x"
27     .linkonce discard
28     .align 2
29     .globl _ZN9CoolClass3getEv
30     .def _ZN9CoolClass3getEv; .scl 2; .type 32; .endif
31     .seh_proc    _ZN9CoolClass3getEv
32 _ZN9CoolClass3getEv:
33 .LFB1:
34     push    rbp
35     .seh_pushreg    rbp
36     mov rbp, rsp
37     .seh_setframe    rbp, 0
38     .seh_endprologue
39     mov QWORD PTR 16[rbp], rcx
40     mov rax, QWORD PTR 16[rbp]
41     mov eax, DWORD PTR 8[rax]
42     pop rbp
43     ret
44     .seh_endproc
45     .section      .text$_ZN13PlainOldClass3setEi,"x"
46     .linkonce discard
47     .align 2
48     .globl _ZN13PlainOldClass3setEi
49     .def _ZN13PlainOldClass3setEi; .scl 2; .type 32; .
    endif
50     .seh_proc    _ZN13PlainOldClass3setEi
51 _ZN13PlainOldClass3setEi:
52 .LFB2:
53     push    rbp
54     .seh_pushreg    rbp
55     mov rbp, rsp
56     .seh_setframe    rbp, 0
57     .seh_endprologue
58     mov QWORD PTR 16[rbp], rcx
59     mov DWORD PTR 24[rbp], edx
60     mov rax, QWORD PTR 16[rbp]
61     mov edx, DWORD PTR 24[rbp]
62     mov DWORD PTR [rax], edx
63     nop
64     pop rbp

```

```

65     ret
66     .seh_endproc
67     .section      .text$_ZN4BaseC2Ev,"x"
68     .linkonce discard
69     .align 2
70     .globl _ZN4BaseC2Ev
71     .def      _ZN4BaseC2Ev;      .scl      2; .type      32; .endif
72     .seh_proc    _ZN4BaseC2Ev
73 _ZN4BaseC2Ev:
74 .LFB7:
75     push    rbp
76     .seh_pushreg    rbp
77     mov rbp, rsp
78     .seh_setframe    rbp, 0
79     .seh_endprologue
80     mov QWORD PTR 16[rbp], rcx
81     lea rdx, _ZTV4Base[rip+16]
82     mov rax, QWORD PTR 16[rbp]
83     mov QWORD PTR [rax], rdx
84     nop
85     pop rbp
86     ret
87     .seh_endproc
88     .section      .text$_ZN9CoolClassC1Ev,"x"
89     .linkonce discard
90     .align 2
91     .globl _ZN9CoolClassC1Ev
92     .def      _ZN9CoolClassC1Ev; .scl      2; .type      32; .endif
93     .seh_proc    _ZN9CoolClassC1Ev
94 _ZN9CoolClassC1Ev:
95 .LFB10:
96     push    rbp
97     .seh_pushreg    rbp
98     mov rbp, rsp
99     .seh_setframe    rbp, 0
100    sub rsp, 32
101    .seh_stackalloc 32
102    .seh_endprologue
103    mov QWORD PTR 16[rbp], rcx
104    mov rax, QWORD PTR 16[rbp]
105    mov rcx, rax
106    call     _ZN4BaseC2Ev
107    lea rdx, _ZTV9CoolClass[rip+16]
108    mov rax, QWORD PTR 16[rbp]
109    mov QWORD PTR [rax], rdx
110    nop
111    add rsp, 32
112    pop rbp
113    ret
114    .seh_endproc
115    .def      __main; .scl      2; .type      32; .endif
116    .text
117    .globl main
118    .def      main; .scl      2; .type      32; .endif
119    .seh_proc    main
120 main:
121 .LFB4:

```

```

122     push     rbp
123     .seh_pushreg    rbp
124     push     rbx
125     .seh_pushreg    rbx
126     sub     rsp, 56
127     .seh_stackalloc 56
128     lea     rbp, 128[rsp]
129     .seh_setframe    rbp, 128
130     .seh_endprologue
131     call    __main
132     mov     ecx, 16
133     call    _Znwy
134     mov     rbx, rax
135     mov     rcx, rbx
136     call    _ZN9CoolClassC1Ev
137     mov     QWORD PTR -88[rbp], rbx
138     lea     rax, -92[rbp]
139     mov     edx, 42
140     mov     rcx, rax
141     call    _ZN13PlainOldClass3setEi
142     mov     rax, QWORD PTR -88[rbp]
143     mov     rax, QWORD PTR [rax]
144     mov     rax, QWORD PTR [rax]
145     mov     rcx, QWORD PTR -88[rbp]
146     mov     edx, 42
147     call    rax
148     mov     eax, 0
149     add     rsp, 56
150     pop     rbx
151     pop     rbp
152     ret
153     .seh_endproc
154     .globl  _ZTV9CoolClass
155     .section .rdata$_ZTV9CoolClass,"dr"
156     .linkonce same_size
157     .align 8
158 _ZTV9CoolClass:
159     .quad 0
160     .quad _ZTI9CoolClass
161     .quad _ZN9CoolClass3setEi
162     .quad _ZN9CoolClass3getEv
163     .globl  _ZTV4Base
164     .section .rdata$_ZTV4Base,"dr"
165     .linkonce same_size
166     .align 8
167 _ZTV4Base:
168     .quad 0
169     .quad _ZTI4Base
170     .quad __cxa_pure_virtual
171     .quad __cxa_pure_virtual
172     .globl  _ZTI9CoolClass
173     .section .rdata$_ZTI9CoolClass,"dr"
174     .linkonce same_size
175     .align 8
176 _ZTI9CoolClass:
177     .quad _ZTVN10__cxxabiv120__si_class_type_infoE+16
178     .quad _ZTS9CoolClass

```

```

179     .quad    _ZTI4Base
180     .globl   _ZTS9CoolClass
181     .section .rdata$_ZTS9CoolClass,"dr"
182     .linkonce same_size
183     .align 8
184 _ZTS9CoolClass:
185     .ascii  "9CoolClass\0"
186     .globl   _ZTI4Base
187     .section .rdata$_ZTI4Base,"dr"
188     .linkonce same_size
189     .align 8
190 _ZTI4Base:
191     .quad    _ZTVN10__cxxabiv117__class_type_infoE+16
192     .quad    _ZTS4Base
193     .globl   _ZTS4Base
194     .section .rdata$_ZTS4Base,"dr"
195     .linkonce same_size
196 _ZTS4Base:
197     .ascii  "4Base\0"
198     .ident   "GCC: (x86_64-win32-seh-rev0, Built by MinGW-W64
199     project) 8.1.0"
200     .def     _Znwy; .scl 2; .type 32; .endef
201     .def     __cxa_pure_virtual; .scl 2; .type 32; .endef

```

task5/main.cpp

```

1  #include <iostream>
2
3  using namespace std;
4
5  class B
6  {
7  public:
8      virtual int __cdecl prva() = 0;
9      virtual int __cdecl druga(int) = 0;
10 };
11
12 class D : public B
13 {
14 public:
15     virtual int __cdecl prva() { return 42; }
16     virtual int __cdecl druga(int x) { return prva() + x; }
17 };
18
19 void solution(B *pb, int x) {
20     /*
21     int p = pb->prva();
22     int d = pb->druga(x);
23     cout << p << endl << d << endl;
24     */
25
26     typedef int (*funPrvaType)(void *);
27     typedef int (*funDrugaType)(void *, int);
28     typedef unsigned long long addr_t;
29
30     addr_t vtableAddress = *((addr_t *) pb);
31     addr_t *vtable = (addr_t *) vtableAddress;
32

```

```

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 }

```