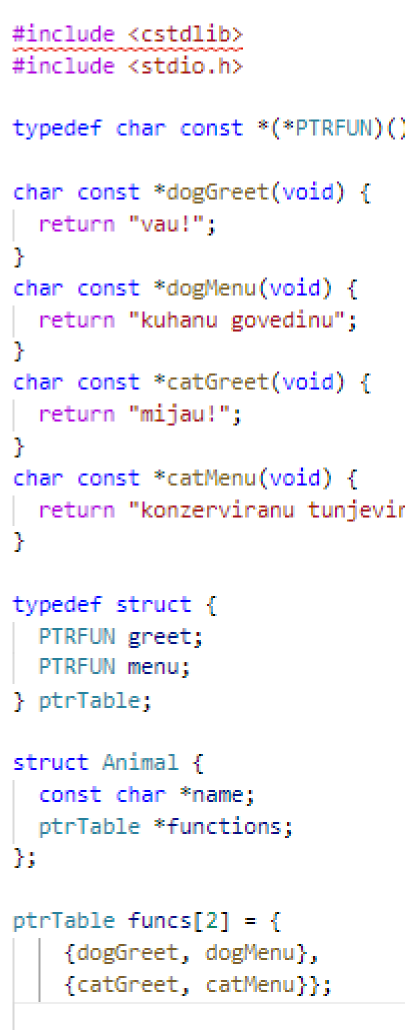
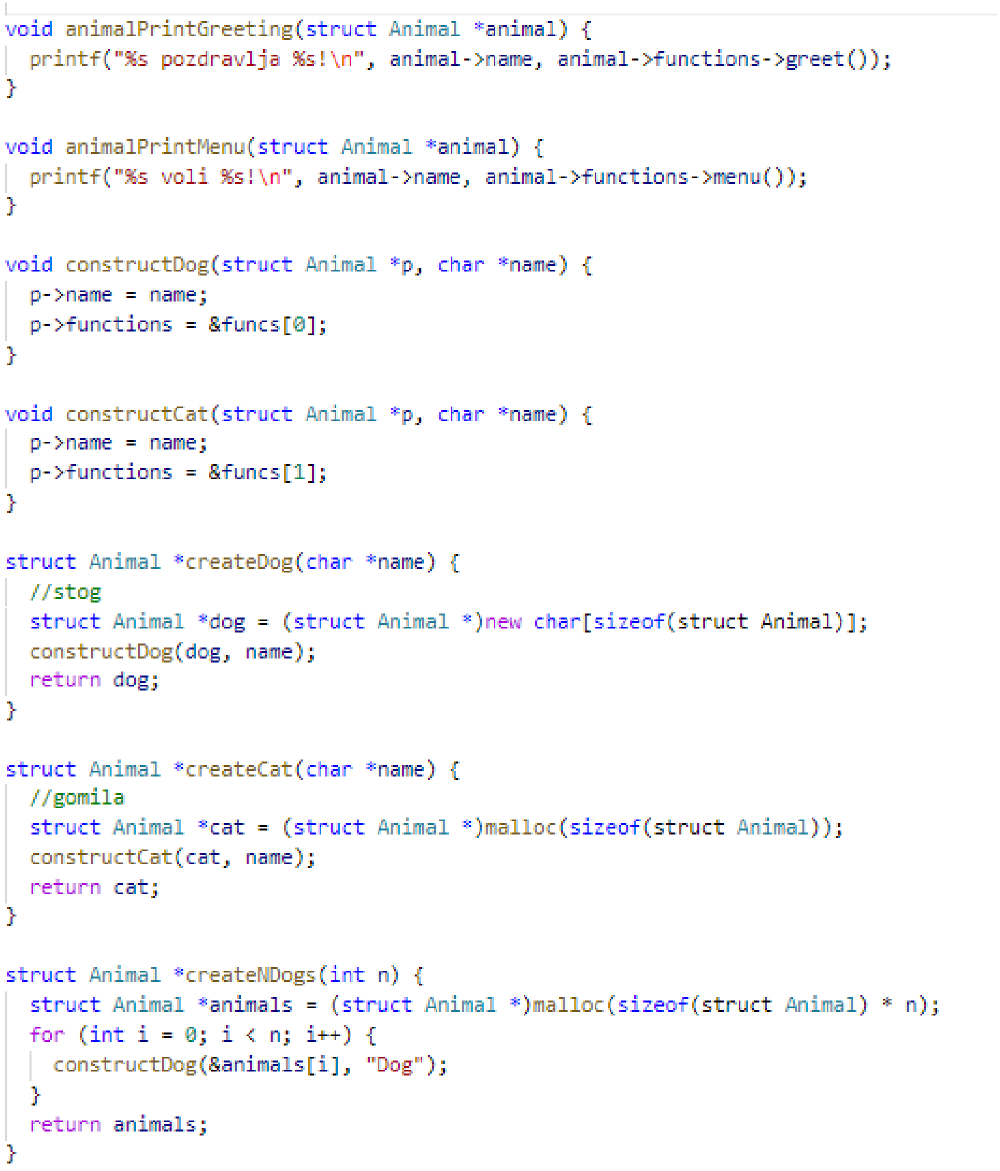
1 zadatak

animal.h

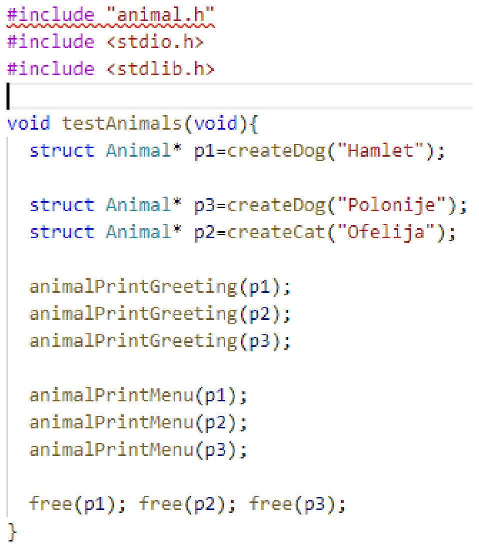


```
#include <cstdlib>
#include <stdio.h>

typedef char const *(*PTRFUN)(

char const *dogGreet(void) {
  return "vau!";
}
char const *dogMenu(void) {
  return "kuhanu govedinu";
}
char const *catGreet(void) {
  return "mijau!";
}
char const *catMenu(void) {
  return "konzerviranu tunjevir
}

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

struct Animal {
  const char *name;
  ptrTable *functions;
};

ptrTable funcs[2] = {
    {dogGreet, dogMenu},
    {catGreet, catMenu}};
```

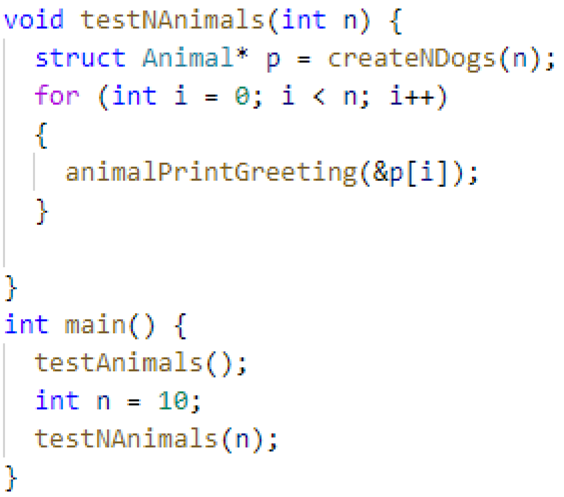


```
void animalPrintGreeting(struct Animal *animal) {
  printf("%s pozdravlja %s!\n", animal->name, animal->functions->greet());
}

void animalPrintMenu(struct Animal *animal) {
  printf("%s voli %s!\n", animal->name, animal->functions->menu());
}

void constructDog(struct Animal *p, char *name) {
  p->name = name;
  p->functions = &funcs[0];
}

void constructCat(struct Animal *p, char *name) {
  p->name = name;
  p->functions = &funcs[1];
}

struct Animal *createDog(char *name) {
  //stog
  struct Animal *dog = (struct Animal *)new char[sizeof(struct Animal)];
  constructDog(dog, name);
  return dog;
}

struct Animal *createCat(char *name) {
  //gomila
  struct Animal *cat = (struct Animal *)malloc(sizeof(struct Animal));
  constructCat(cat, name);
  return cat;
}

struct Animal *createNDogs(int n) {
  struct Animal *animals = (struct Animal *)malloc(sizeof(struct Animal) * n);
  for (int i = 0; i < n; i++) {
    constructDog(&animals[i], "Dog");
  }
  return animals;
}
```

zad1.c

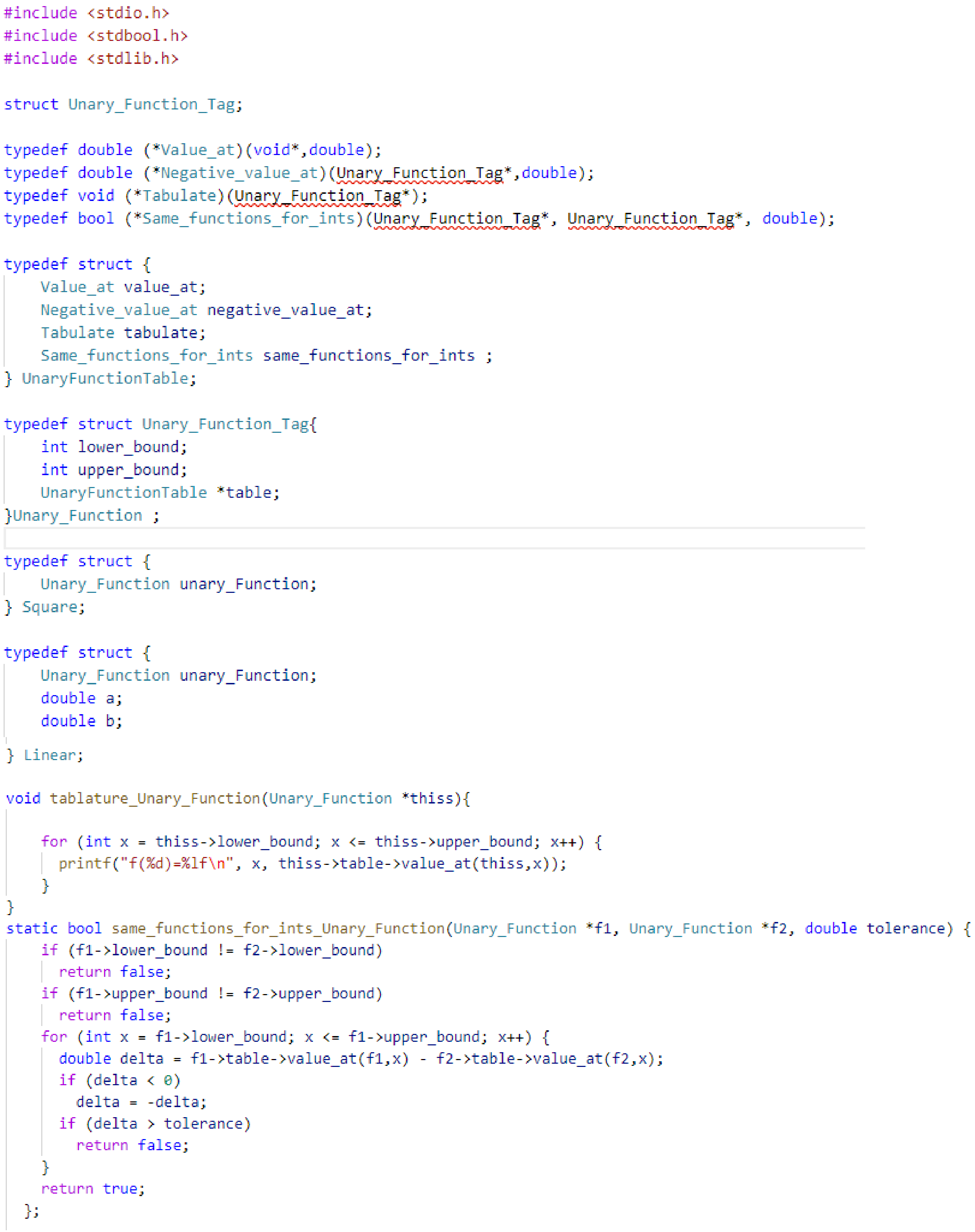


```
#include "animal.h"
#include <stdio.h>
#include <stdlib.h>

void testAnimals(void){
  struct Animal* p1=createDog("Hamlet");

  struct Animal* p3=createDog("Polonije");
  struct Animal* p2=createCat("Ofelija");

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

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

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



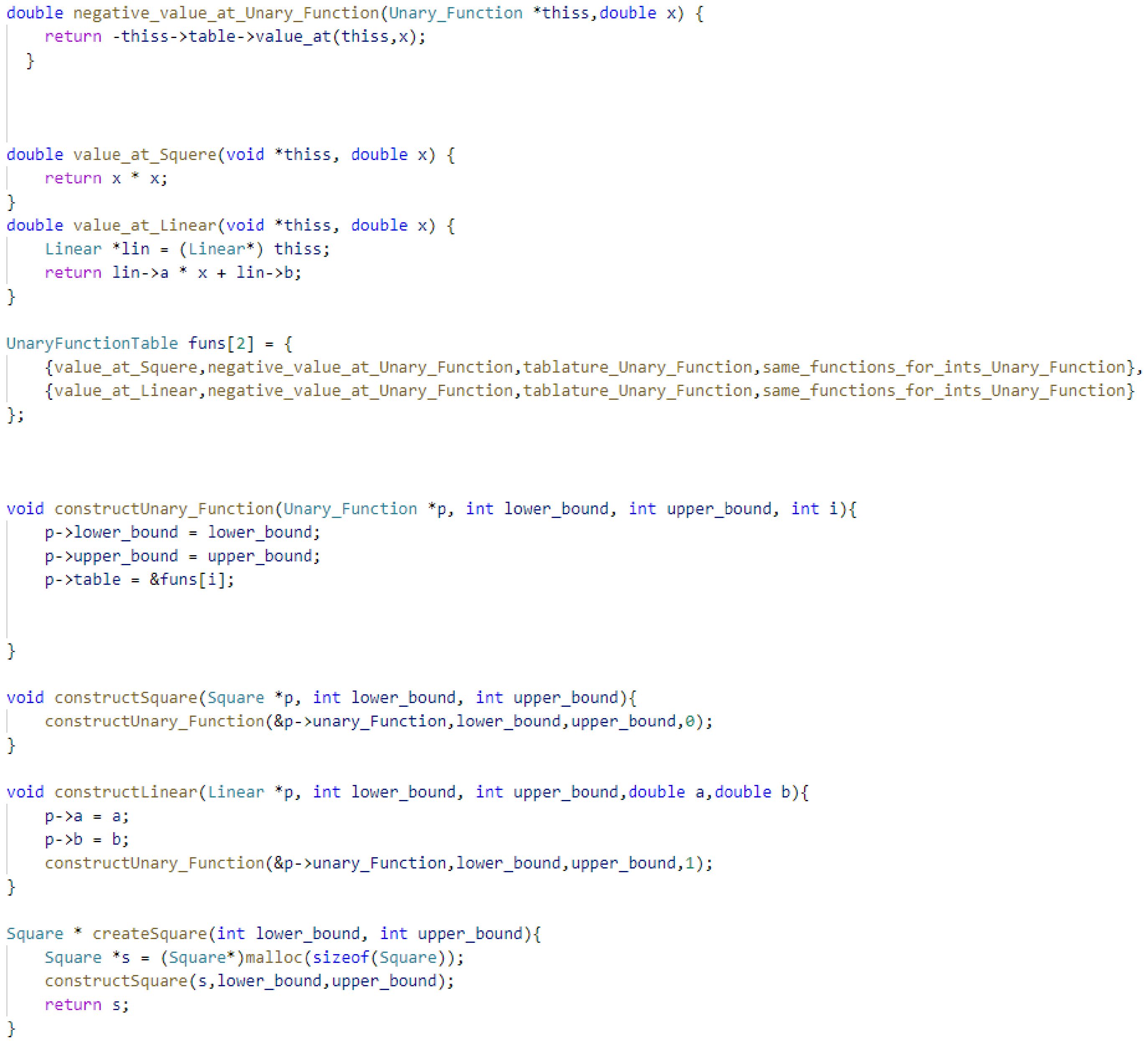
```
void testNAnimals(int n) {
  struct Animal* p = createNDogs(n);
  for (int i = 0; i < n; i++)
  {
    animalPrintGreeting(&p[i]);
  }

}
int main() {
  testAnimals();
  int n = 10;
  testNAnimals(n);
}
```

2 zadatak

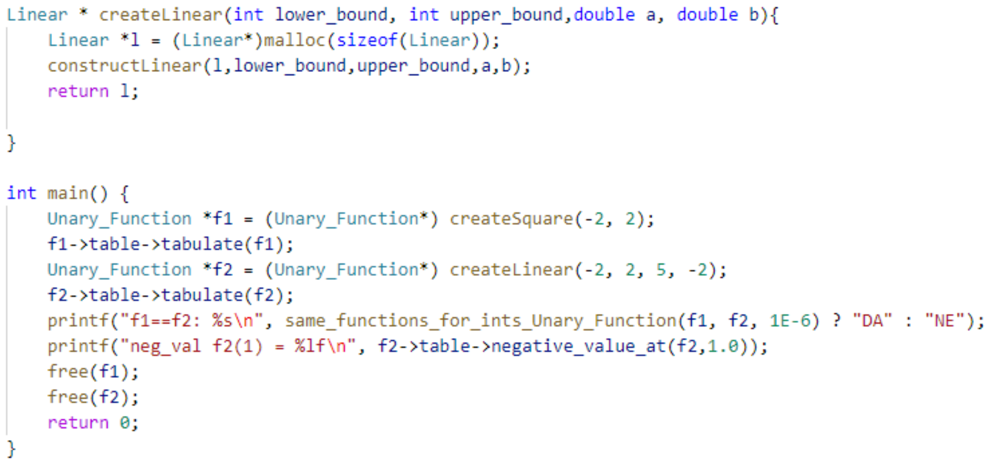


```
#include <stdio.h>
#include <stdbool.h>
#include <stdlib.h>

struct Unary_Function_Tag;

typedef double (*Value_at)(void*,double);
typedef double (*Negative_value_at)(Unary_Function_Tag*,double);
typedef void (*Tabulate)(Unary_Function_Tag*);
typedef bool (*Same_functions_for_ints)(Unary_Function_Tag*, Unary_Function_Tag*, double);

typedef struct {
    Value_at value_at;
    Negative_value_at negative_value_at;
    Tabulate tabulate;
    Same_functions_for_ints same_functions_for_ints ;
} UnaryFunctionTable;

typedef struct Unary_Function_Tag{
    int lower_bound;
    int upper_bound;
    UnaryFunctionTable *table;
}Unary_Function ;

typedef struct {
    Unary_Function unary_Function;
} Square;

typedef struct {
    Unary_Function unary_Function;
    double a;
    double b;

} Linear;

void tablature_Unary_Function(Unary_Function *thiss){

    for (int x = thiss->lower_bound; x <= thiss->upper_bound; x++) {
      printf("f(%d)=%lf\n", x, thiss->table->value_at(thiss,x));
    }
}
static bool same_functions_for_ints_Unary_Function(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->table->value_at(f1,x) - f2->table->value_at(f2,x);
      if (delta < 0)
        delta = -delta;
      if (delta > tolerance)
        return false;
    }
    return true;
  };
```

2 zadatak – nastavak

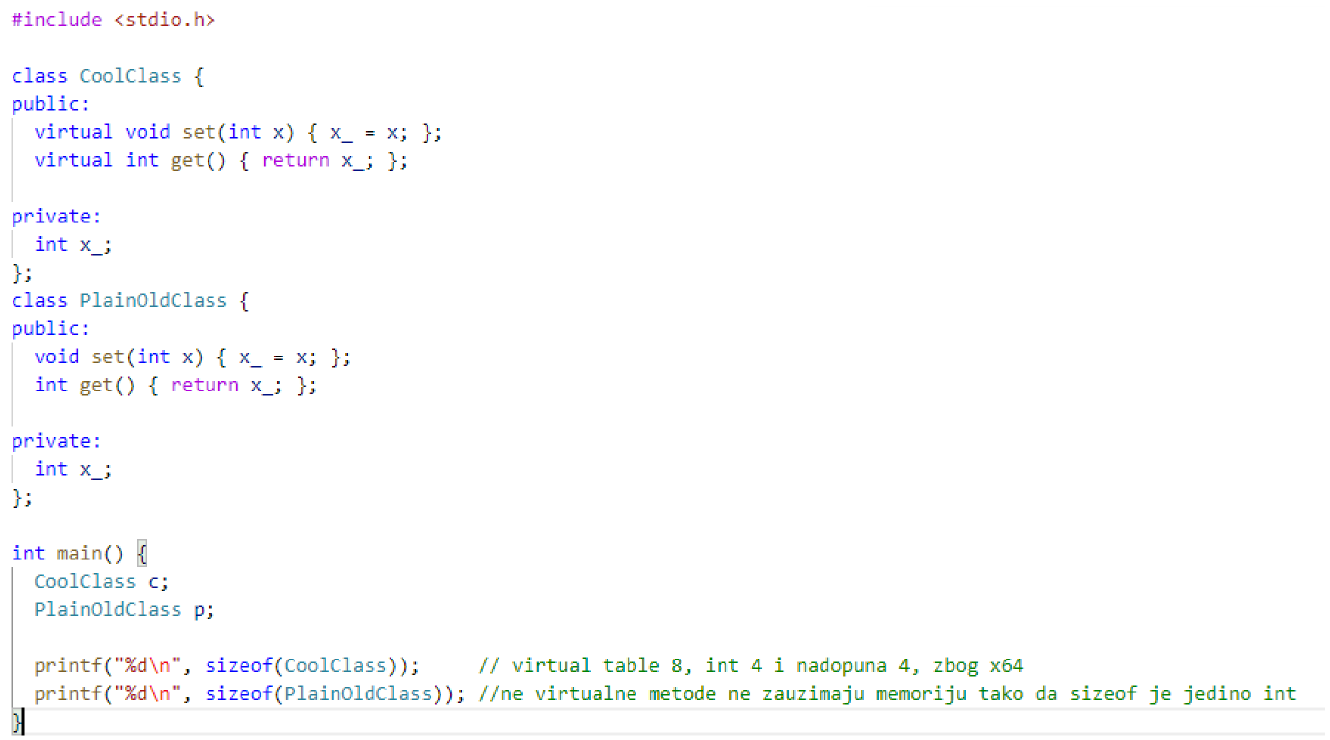


```
double negative_value_at_Unary_Function(Unary_Function *thiss,double x) {
    return -thiss->table->value_at(thiss,x);
  }



double value_at_Squere(void *thiss, double x) {
    return x * x;
}
double value_at_Linear(void *thiss, double x) {
    Linear *lin = (Linear*) thiss;
    return lin->a * x + lin->b;
}

UnaryFunctionTable funs[2] = {
    {value_at_Squere,negative_value_at_Unary_Function,tablature_Unary_Function,same_functions_for_ints_Unary_Function},
    {value_at_Linear,negative_value_at_Unary_Function,tablature_Unary_Function,same_functions_for_ints_Unary_Function}
};



void constructUnary_Function(Unary_Function *p, int lower_bound, int upper_bound, int i){
    p->lower_bound = lower_bound;
    p->upper_bound = upper_bound;
    p->table = &funs[i];


}

void constructSquare(Square *p, int lower_bound, int upper_bound){
    constructUnary_Function(&p->unary_Function,lower_bound,upper_bound,0);
}

void constructLinear(Linear *p, int lower_bound, int upper_bound,double a,double b){
    p->a = a;
    p->b = b;
    constructUnary_Function(&p->unary_Function,lower_bound,upper_bound,1);
}

Square * createSquare(int lower_bound, int upper_bound){
    Square *s = (Square*)malloc(sizeof(Square));
    constructSquare(s,lower_bound,upper_bound);
    return s;
}
```



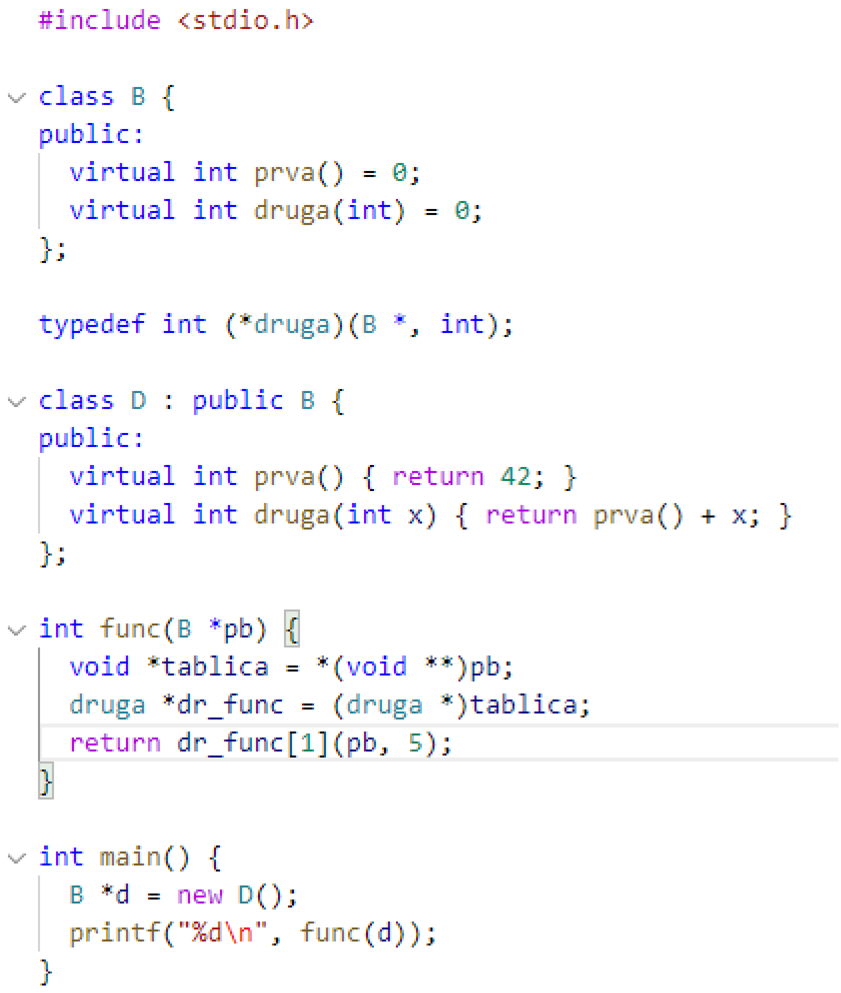
```
Linear * createLinear(int lower_bound, int upper_bound,double a, double b){
    Linear *l = (Linear*)malloc(sizeof(Linear));
    constructLinear(l,lower_bound,upper_bound,a,b);
    return l;

}

int main() {
    Unary_Function *f1 = (Unary_Function*) createSquare(-2, 2);
    f1->table->tabulate(f1);
    Unary_Function *f2 = (Unary_Function*) createLinear(-2, 2, 5, -2);
    f2->table->tabulate(f2);
    printf("f1==f2: %s\n", same_functions_for_ints_Unary_Function(f1, f2, 1E-6) ? "DA" : "NE");
    printf("neg_val f2(1) = %lf\n", f2->table->negative_value_at(f2,1.0));
    free(f1);
    free(f2);
    return 0;
}
```

3 zadatak



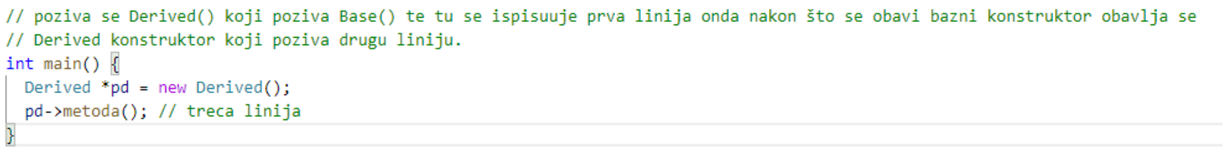
```
#include <stdio.h>

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() {
    CoolClass c;
    PlainOldClass p;

    printf("%d\n", sizeof(CoolClass));     // virtual table 8, int 4 i nadopuna 4, zbog x64
    printf("%d\n", sizeof(PlainOldClass)); //ne virtualne metode ne zauzimaju memoriju tako da sizeof je jedino int
}
```

5 zadatak



```
#include <stdio.h>

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

typedef int (*druga)(B *, int);

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

int func(B *pb) {
    void *tablica = *(void **)pb;
    druga *dr_func = (druga *)tablica;
    return dr_func[1](pb, 5);
}

int main() {
    B *d = new D();
    printf("%d\n", func(d));
}
```

6 zad



```
// poziva se Derived() koji poziva Base() te tu se ispisuuje prva linija onda nakon što se obavi bazni konstruktor obavlja se
// Derived konstruktor koji poziva drugu liniju.
int main() {
    Derived *pd = new Derived();
    pd->metoda(); // treca linija
}
```