Prva laboratorijska vježba iz Oblikovnih obrazaca u programiranju:
dinamički polimorfizam u C-u, C++-u i strojnom kodu, konstrukcija objekata

1. Dinamički polimorfizam u C-u (20% bodova)

Ova vježba razmatra ostvarivanje dinamičkog polimorfizma u programskom jeziku C. Potrebno je napisati kod niže razine koji bi omogućio ispravno izvršavanje priložene ispitne funkcije.

```
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);
}
```

Prikazana ispitna funkcija treba generirati sljedeći ispis.

```
  Hamlet pozdravlja: vau!
  Ofelija pozdravlja: mijau!
  Polonije pozdravlja: vau!
  Hamlet voli kuhanu govedinu
  Ofelija voli konzerviranu tunjevinu
  Polonije voli kuhanu govedinu
```

Pretpostavimo da su funkcije koje definiraju ponašanje konkretnih tipova zadane kako slijedi.

```
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 tunjevinu";
}
```

Potrebno je oblikovati sljedeće elemente rješenja.

- Dvije tablice pokazivača na funkcije koje definiraju ponašanje konkretnih tipova, kao i kôd za njihovo incijaliziranje. Prikladna deklaracija podatkovnog tipa za pohranjivanje elemenata tih dviju tablica bila bi: `typedef char const* (*PTRFUN)();`
- Podatkovni tip struct `Animal` koji sadrži i) pokazivač na ime ljubimca te ii) pokazivač na tablicu funkcija (vidi gore) koja definira ponašanje odgovarajućeg konkretnog tipa. Pojašnjenje: tablicu pokazivača mogli bismo i umetnuti u tip `Animal` ali obično preferiramo rješenje s pokazivačem kako bismo osigurali usklađeno ponašanje objekata istog tipa te što više smanjili memorijski otisak polimorfnih objekata.
- Funkcije `animalPrintGreeting` i `animalPrintMenu` koje generiraju specificirani ispis pozivanjem odgovarajućeg elementa tablice funkcija zadanog polimorfnog objekta.
- Funkcije `constructDog` i `constructCat` koje primaju i) pokazivač na memorijski prostor u kojem treba stvoriti objekt te ii) pokazivač na znakovni niz s imenom ljubimca. Funkcije trebaju u zadanom memorijskom prostoru inicijalizirati objekt odgovarajućeg konkretnog tipa.
- Funkcije `createDog` i `createCat` koje alociraju memoriju i pozivaju funkcije `constructDog` odnosno `constructCat`.
- Uputa: nemojte komplicirati, službeno rješenje ima manje od 70 redaka uredno formatiranog C-a.

Obratite pažnju na to da deklaracija `PTRFUN pfun;` u C-u (ali ne i C++-u!) definira pokazivač na funkciju s nespecificiranim argumentima. To znači da `pfun` može pokazivati na bilo koju funkciju koja vraća `char const*` (**detalji**). Naravno, pri korištenju pokazivača `pfun` moramo paziti da broj i tipovi argumenata navedeni u pozivu odgovaraju argumentima funkcije na koju pokazivač pokazuje (u suprotnom ponašanje programa nije definirano).

Vaše rješenje mora biti takvo da memorijsko zauzeće za svaki primjerak "razreda" (psa, mačke) ne ovisi o broju virtualnih metoda. Drugim riječima, dodavanje nove virtualne metode ne smije **kod svakog primjerka** psa i mačke povećati memorijsko zauzeće.

Pokažite da je konkretne objekte moguće kreirati i na gomili i na stogu (**detalji**). Memorijski prostor na stogu zauzmite lokalnom varijablom, a za zauzimanje memorije na gomili pozovite `malloc`.

Napišite funkciju za stvaranje n pasa, gdje je n argument funkcije (npr. za potrebe vuče saonica). Pokažite kako bismo to ostvarili jednim pozivom funkcije `malloc` i potrebnim brojem poziva funkcije `constructDog`.

Nakon rješavanja zadatka, uspostavite vezu s terminologijom iz objektno orijentiranih jezika. Koji elementi vašeg rješenja bi korespondirali s podatkovnim članovima objekta, metodama, virtualnim metodama, konstruktorima, te virtualnim tablicama?

Ako vas je objektno orijentirano programiranje u C-u očaralo i želite o tome znati više - pogledajte sljedeću **knjigu**. Međutim, prije nego što donesete definitivnu odluku o prelasku s C++-a na C, preporučamo vam da ipak razmislite o iznimkama, predlošcima, STL-u i novim mogućnostima koje nude standardi iz **2011**. i **2014**. godine.

2. Virtualne tablice u C++-u (20% bodova)

Dan je kratak program napisan u programskom jeziku C++ koji koristi razrede, nasljeđivanje, statičke, virtualne i nevirtualne funkcije.

```
#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 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) {
      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) {
      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;
}
```

Upute:

1. funkcije za inicijaliziranje objekata moraju moći podržati različite načine alociranja memorijskog prostora;
2. vaše rješenje ne smije povećati memorijski otisak objekata mjeren operatorom sizeof.

Zadaci.

1. Analizirajte napisani kod. Skicirajte dijagram razreda. Za svaki razred prikažite kako će izgledati njegova tablica virtualnih funkcija.
2. Napišite programsko ostvarenje u jeziku C koje predstavlja identičan program. Pripazite kakve (i koliko) struktura podataka ćete koristiti, gdje će biti pojedini podatkovni članovi i slično. Karakteristike vašeg rješenja trebaju biti slične karakteristikama realizacije u C++-u, u smislu lakoće nadogradnje.

5. Ručno prozivanje virtualne tablice pokazivačima na funkcije (15% bodova)

Cilj ove vježbe je pokazati da virtualne metode objekata možemo pozivati i bez korištenja njihovih simboličkih imena, pod pretpostavkom da se pokazivač na virtualnu tablicu nalazi na samom početku objekta. Ta pretpostavka vrijedi kod svih popularnih prevoditelja, a lako ju je testirati na način kojeg smo pokazali u prethodnoj vježbi. Neka je zadan je sljedeći kod:

```
class B{
```

```
public:
  virtual int prva()=0;
  virtual int druga(int)=0;
};

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

Potrebno je napisati funkciju koja prima pokazivač `pb` na objekt razreda `B` te ispisuje povratne vrijednosti dvaju metoda, ali na način da u kodu ne navodimo simbolička imena `prva` i `druga`. Zadatak riješite primjenom pokazivača na slobodne funkcije kakve smo koristili i do sada. Nemojte koristiti pokazivače na članske funkcije jer bi u tom slučaju vježba bila manje poučna.

Uputa za Windowse. Za ispravan prijenos skrivenog pokazivača na matični objekt, deklarirajte da se metode pozivaju po konvenciji programskog jezika C, kako slijedi:

```
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;}
};
```

Pomoć. Deklaracije pokazivača najgore su projektirani dio jezika C i C++. Zbog tog propusta deklariranje pokazivača na funkcije ponekad podsjeća na alkemiju ili igranje lota. Da bismo vam olakšali pogađanje dobitne kombinacije, dajemo vam nekoliko primjera:

```
  // pfun pokazuje na funkciju bez parametara koja vraća int
  int (*pfun)();
  // pfun pokazuje na funkciju s dva parametara koja vraća int
  int (*pfun)(B*, int);
  // odgovarajući operator pretvaranja izgleda ovako:
  pfun = (int (*)()) 0;
```

Obratite pažnju da C i C++ različito tumače prototipove bez pobrojanih argumenata; gore navedena razmatranja odnose se na C++. Analizu deklaracija možete sebi olakšati primjenom automatiziranog **prevoditelja** na engleski jezik.