Dokumentacja programu SymulatorStudia.

Działanie programu:

Wraz z uruchomieniem programu gracz ma dostępne trzy opcje: Zacznij gre, Twórcy, Wyjdz z gry. Wybranie opcji numer 3 zakończy program, natomiast wybór opcji numer 2, wyświetli listę twórców i zakończy działanie programu. Wybór opcji Zacznij gre pozwala na wybór kolejnych trzech opcji: Poczatkujace studio, Swiatowy potentant i Powrot. Wybór 3 opcji sprawi, że wrócimy do poprzedniego menu. Wybór opcji 1 lub 2 uruchomi grę, od naszego wyboru zależy początkowy stan finansów naszego studia. W przypadku 1 jest to 200 waluty w grze w przypadku 2 jest to 4000000 waluty w grze. Dalsza cześć gry nie różni się dla obu wariantów(zaleca się wybór opcji pierwszej, gdyż łatwiej w niej o porażkę). Po wyborze stanu początkowego zostaniemy przekierowani do tworzenia produkcji filmowej. Wybieramy gatunek filmu, przeznaczamy pieniądze z budżetu naszego studia na budżet produkcyjny, wybieramy czas trwania seansu w minutach, a następnie tytuł produkcji. Następnie otworzy się okno wyboru ekipy filmowej. Zatrudniamy 8 osób z pieniędzy z budżetu produkcyjnego. Na każdą posadę zostanie zaoferowanych 10 kandydatów. Możemy później wybrać usunięcie osoby z listy. Następnie z budżetu produkcyjnego możemy wynająć lub kupić 3 sprzęty. Kupno sprzętu sprawia, że znajduje się on w naszym magazynie do końca gry i nie musimy ponosić żadnych kosztów w związku z używaniem go. Wynajem sprawi, że wykorzystamy sprzęt tylko w najbliższej produkcji. Następnie z niewykorzystanego budżetu produkcyjnego, lub jeśli nie przeznaczyliśmy całego budżetu studia na budżet produkcyjny to także z tej wartości, użytkownik musi wybrać kwotę jaką przeznacza na budżet marketingowy. W budżecie marketingowym gracz przeznacza kwotę na 6 rodzajów marketingu. W przypadku, gdy przeznaczymy wszystkie pieniądze z budżetu marketingowego, a nie wypełniliśmy jeszcze wszystkich rodzajów, to reszta zapełni się automatycznie zerami. Następnie gracz zobaczy raport tego jakie wartości miały wpływ na produkcję, a także porównanie początkowego salda zysku/straty z produkcji, a także ukazany zostanie nowy budżet i zostanie zrobiona jego kopia. Następnie gracz ma wybór, czy chce kontynuować grę, czy wrócić do menu głównego. Jeśli gracz wybierze kontynuację to zaczyna proces produkcji kolejnego filmu z budżetem uzyskanym po pierwszej produkcji studia. Program jest prostą grą polegającą na utrzymaniu stanu konta wytwórni filmowej powyżej zera, gdy wartość ta osiągnie wartość nie większą zero gra kończy się.

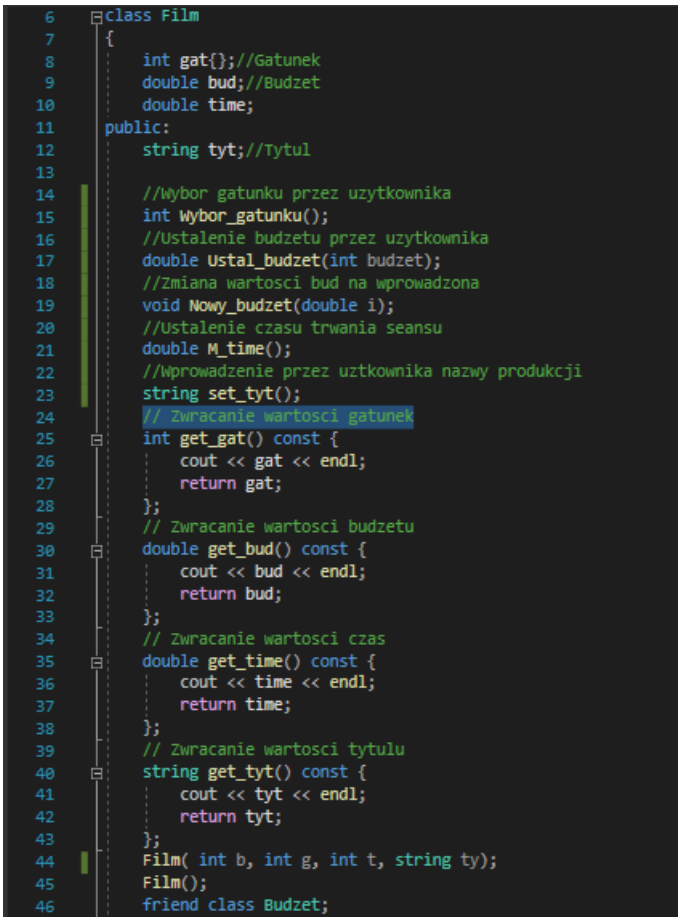
Budowa:

Program zrealizowany jest w paradygmacie obiektowym. Składa się z klas mających własne pola i metod. Dla struktury programu najważniejsze są dwie klasy, w których wykorzystywana jest agregacja: Baza i Budżet. W bazie zaagregowana jest na podwójnym wskaźniku obiekt klasy lista. Dzięki temu można stworzyć listę osób, która składa się z obiektów różnych klas(np.Rezyser,Scenarzysta), które dziedziczą z abstrakcyjnej klasy Persona. Dzięki czemu w funkcji Wybor_ekipy(), można wybrać odpowiednich kandydatów i z sumując wartości ich poszczególnych cech(np.roz-Rozpoznawalność), informacje te przekazywane są do kontenera na dane, a następnie użyte przy ustalaniu ostatecznego budżetu. Drugą klasą, która wykorzystuje agregacje jest Budżet. To w niej znajdują się wskaźniki typu Marketing, Sprzet, Wplyw_M(czyli wpływ marketingowy), a także zmienne finanse, finanse2 ,suma_gat, suma_rep, suma_prof i suma_roz. To właśnie do 4 ostatnich są przekazywane dane z wektora. Następnie na podstawie właściwości obiektów w przeciążonych operatorach dokonywane są obliczenia, które zwracają wartość, która staje się nowym finanse.

Wymogi etapów:

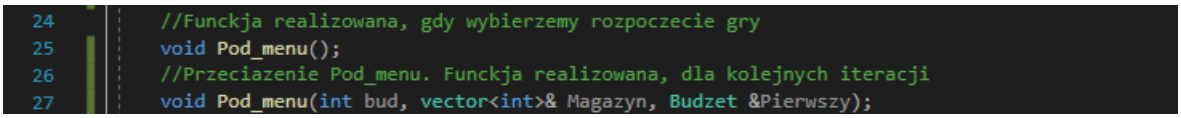
Etap 1:

Przykład klasy Film.h.



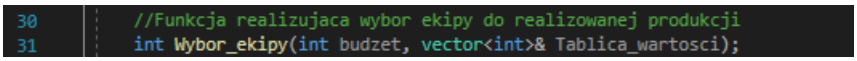
```
class Film
{
    int gat{};//Gatunek
    double bud;//Budzet
    double time;
public:
    string tyt;//Tytul

    //Wybor gatunku przez uzytkownika
    int Wybor_gatunku();
    //Ustalenie budzetu przez uzytkownika
    double Ustal_budzet(int budzet);
    //Zmiana wartosci bud na wprowadzona
    void Nowy_budzet(double i);
    //Ustalenie czasu trwania seansu
    double M_time();
    //Wprowadzenie przez uztkownika nazwy produkcji
    string set_tyt();
    // Zwracanie wartosci gatunek
    int get_gat() const {
        cout << gat << endl;
        return gat;
    };
    // Zwracanie wartosci budzetu
    double get_bud() const {
        cout << bud << endl;
        return bud;
    };
    // Zwracanie wartosci czas
    double get_time() const {
        cout << time << endl;
        return time;
    };
    // Zwracanie wartosci tytulu
    string get_tyt() const {
        cout << tyt << endl;
        return tyt;
    };
    Film( int b, int g, int t, string ty);
    Film();
    friend class Budzet;
```

Przeciążenie. Przykład w Dod_funk.h.



```
    //Funckja realizowana, gdy wybierzemy rozpoczecie gry
    void Pod_menu();
    //Przeciazenie Pod_menu. Funckja realizowana, dla kolejnych iteracji
    void Pod_menu(int bud, vector<int>& Magazyn, Budzet &Pierwszy);
```

Referencja. Przykład w Dod_funk.h.



```
//Funkcja realizujaca wybor ekipy do realizowanej produkcji
int Wybor_ekipy(int budzet, vector<int>& Tablica_wartosci);
```

Etap 2:

Pamięć. Przykład użycia Baza.cpp.

```cpp
//Tworzy lub zwieksza baze osob, rezerwuje pamiec
void Baza::stworz(Persona* osoba)
{

    Lista** temp = new Lista * [ilosc_osob + 1];
    for (size_t i = 0; i < ilosc_osob; ++i)
        temp[i] = lista_osob[i];
    Lista* kart = new Lista{ osoba };
    temp[ilosc_osob] = kart;
    delete[] lista_osob;
    ++lista_osob;
    ++ilosc_osob;
    lista_osob = temp;
}
```

```cpp
    //Usuniecie calej listy osob
    void Baza::usun()
    {
        for (size_t i = 0; i < ilosc_osob; ++i) {
            delete lista_osob[i];
        }
        delete[] lista_osob;
        lista_osob = nullptr;
        ilosc_osob = 0;
        cout << "Usuniety jestes typie" << endl;

    }
    //Usuniecie konkretnej osoby
    void Baza::usun(int z)
    {
    if (z < ilosc_osob) {
        Lista** temp = new Lista * [ilosc_osob - 1];
        short int j{ -1 };
        for (int i = 0; i <ilosc_osob ; ++i)
            if (i != z) {

                ++j;
                temp[j] = lista_osob[i];
            }

        delete[] lista_osob;
        --ilosc_osob;
        lista_osob = temp;
    }
    else
    cout << "Index jest nieprawidlowy" << endl;
    }
```

```cpp
void Baza::wyswietl(int i) {
//   int i = 0;
    cout << "Oferowany kandydat:" << endl;
    cout <<"Dostosowanie do gatunkow: "<< lista_osob[i]->get_gat() << "\t";
    cout << endl;
    cout << "Uznanie krytykow: " << lista_osob[i]->get_rep() << "\t";
    cout << endl;
    cout << "Pobierana pensja w tysiacach: " << lista_osob[i]->get_pen() << "\t";
    cout << endl;
    cout << "Rozpoznawalnosc: " << lista_osob[i]->get_roz() << "\t";
    cout << endl;
    cout << "Profesjonalizm: " << lista_osob[i]->get_prof() << "\t";
    cout << endl;
    cout << "Imie: " << lista_osob[i]->get_imie() << "\t";
    cout << endl;
    cout << "Nazwisko: " << lista_osob[i]->get_nazwisko() << "\t";
    cout << endl;
    cout << endl;
}

//Wyswietlenie calej listy osob
void Baza::wyswietl_all() {
    cout << ilosc_osob<<endl;
    for (int i = 0; i < ilosc_osob; i++) {
        cout << endl;
         cout << "Numer osoby w bazie: " << i << endl;
         cout << endl;
        cout << lista_osob[i]->get_gat() << "\t";
        cout << endl;
        cout << "Uznanie krytykow: " << lista_osob[i]->get_rep() << "\t";
        cout << endl;
        cout << "Pobierana pensja w tysiacach: " << lista_osob[i]->get_pen() << "\t";
        cout << endl;
        cout << "Rozpoznawalnosc: " << lista_osob[i]->get_roz() << "\t";
        cout << endl;
        cout << "Profesjonalizm: " << lista_osob[i]->get_prof() << "\t";
        cout << endl;
        cout << "Imie: " << lista_osob[i]->get_imie() << "\t";
        cout << endl;
        cout << "Nazwisko: " << lista_osob[i]->get_nazwisko() << "\t";
        cout << endl;
    }
}
```

Funkcje nie mogące zmieniać wartości. Przykład Marketing.h i Marketing.cpp

```cpp
public:
    Marketing(int Kupno, int Sniad, int Plakaty,int
    //Wypisanie informacji marketingowych
    Marketing(int Kupno, int Sniad, int Plakaty, int
    Marketing(int Kupno, int Sniad, int Plakaty, int
    Marketing(int Kupno, int Sniad, int Plakaty);
    Marketing(int Kupno, int Sniad);
    Marketing(int Kupno);
    Marketing();
    ~Marketing();
    //Wypisanie wszystkich wartosci na ekranie
    void informacje_marketingowe() const;
    //Wyznaczanie i podzial budzetu marketingowego
    void Rozdzial_budzetu_marketingowego(int bud);
    friend int Sedzia(Marketing mar, Wplyw_M m);
    friend class Budzet;
    // Zwracanie wartosci kupno
    int get_kupno() const;
    // Zwracanie wartosci sniad
    int get_sniad() const;
    // Zwracanie wartosci plakaty
    int get_plakaty() const;
    // Zwracanie wartosci trailer
    int get_trailer() const;
    // Zwracanie wartosci bilb
    int get_bilb() const;
    // Zwracanie wartosci internet
    int get_internet() const;

};
```

```cpp
int Marketing::get_kupno()const
{
    return kupno;
}

int Marketing::get_sniad()const
{
    return sniad;
}

int Marketing::get_plakaty()const
{
    return plakaty;
}

int Marketing::get_trailer()const
{
    return trailer;
}

int Marketing::get_bilb()const
{
    return bilb;
}

int Marketing::get_internet()const
{
    return internet;
}
```

Konstruktor z listą. Przykład w Film.cpp.

```cpp
}
Film::Film(int b, int g, int t, string ty):bud(b),gat(g),time(t),tyt(ty)
{

}
```

Etap 3:

Typ auto. Przykład użycia w Dod_funk.cpp.

```cpp
            auto zmiej = rand() % stala;
            auto zwie = rand() % stala;
```

Klasa zagnieżdżona. Przykład użycia Sprzęt.h.

```cpp
class Sprzet
{
    int jakosc;//Jakos jaka mozna uzyskac danym sprzetem
    class Opcja {
        int kupno;//cena kupna sprzetu
        int wynajem;//cena za wynajem
    public:
        Opcja(int Kupno, int Wynajem):kupno(Kupno),wynajem(Wynajem){}
        void pokaz();
    };
    Opcja set() { return Opcja(44 * jakosc, jakosc); }
public:
    Sprzet(int Jakosc);
```

Konstruktory i destruktory. Przykład użycia:

Konstruktor delegujący Marketing.h i Marketing.cpp.

```
14      public:
15          Marketing(int Kupno, int Sniad, int Plakaty,int Trailer, int Bilb, int Internet);
16          //Wypisanie informacji marketingowych
17          Marketing(int Kupno, int Sniad, int Plakaty, int Trailer, int Bilb);
18          Marketing(int Kupno, int Sniad, int Plakaty, int Trailer);
19          Marketing(int Kupno, int Sniad, int Plakaty);
20          Marketing(int Kupno, int Sniad);
21          Marketing(int Kupno);
22          Marketing();
```

```
37  Marketing::Marketing(int Kupno, int Sniad, int Plakaty, int Trailer, int Bilb, int Internet) :
38      kupno(Kupno),
39      sniad(Sniad),
40  plakaty(Plakaty),
41  trailer(Trailer),
42  bilb(Bilb),
43  internet(Internet){
44      if (kupno < zero)cout<<"Blad kupno";
45      if (sniad < zero)cout << "Blad sniad";
46      if (plakaty < zero)cout << "Blad plakaty";
47      if (trailer < zero)cout << "Blad trailer";
48      if (bilb< zero)cout << "Blad bilb";
49      if (internet < zero)cout << "Blad internet";
50  }
51  Marketing::Marketing(int Kupno, int Sniad, int Plakaty, int Trailer, int Bilb):Marketing(Kupno,Sniad,Plakaty,Trailer,Bilb,zero){}
52  Marketing::Marketing(int Kupno, int Sniad, int Plakaty, int Trailer) : Marketing(Kupno, Sniad, Plakaty, Trailer,zero) {}
53  Marketing::Marketing(int Kupno, int Sniad, int Plakaty) : Marketing(Kupno, Sniad, Plakaty, zero) {}
54  Marketing::Marketing(int Kupno, int Sniad) : Marketing(Kupno, Sniad, zero) {}
55  Marketing::Marketing(int Kupno) : Marketing(Kupno,zero) {}
56  Marketing::Marketing() : Marketing(zero) {}
```

Dekonstruktor Baza.cpp.

```
106 void Baza::usun(int z)
107 {
108 if (z < ilosc_osob) {
109     Lista** temp = new Lista * [ilosc_osob - 1];
110     short int j{ -1 };
111     for (int i = 0; i <ilosc_osob ; ++i)
112         if (i != z) {
113
114             ++j;
115             temp[j] = lista_osob[i];
116         }
117
118         delete[] lista_osob;
119         --ilosc_osob;
120         lista_osob = temp;
121     }
122 else
123 cout << "Index jest nieprawidlowy" << endl;
124 }
125 Baza::~Baza()
126 {
127     usun();
128     if (lista_osob == nullptr) {
129         cout << "Baza usunieta prawidlowo" << endl;
130     }
131     else {
132         cout << "Baza: Cos sie popsulo" << endl;
133
134     }
135
136
```

Konstruktor domyślny Budzet.cpp.

```
125 Budzet::Budzet()
126 {
127     finanse = dzwiescie;
128     mar = new Marketing();
129     wply = new Wplyw_M();
130     sprz = new Sprzet();
131
```

Konstruktor kopiujący Budzet.cpp.

```
132   }
133   Budzet::Budzet(const Budzet& d):mar(new Marketing(d.mar->kupno,d.mar->sniad,d.mar->plakaty,d.mar->trailer,d.mar->bilb,d.mar->internet)),
134   wply(new Wplyw_M(d.wply->zwie,d.wply->zmie)),
135   sprz(new Sprzet(d.sprz->jakosc)),
136   finanse(d.finanse),
137   suma_gat (d.suma_gat),
138   suma_rep     (d.suma_rep),
139   suma_prof    (d.suma_prof),
140   suma_roz     (d.suma_roz)
141   {
142       int k = zero;
143       k = d.mar->kupno + d.mar->bilb + d.mar->internet + d.mar->plakaty + d.mar->sniad + d.mar->trailer;
144       int jako = zero;
145       jako = d.sprz->jakosc;
146       int wplywik = zero;
147       //d.wply->zwie = d.wply->zwie - d.wply->zmie;
148       wplywik = d.wply->zwie - d.wply->zmie;
149       d.get_finanse();
150       int budzet = d.finanse;
151       cout << "Budzet to: " << budzet << endl << "Wplyw marketingowy to: " << d.wply->zwie <<endl<<"Zabawa math: "<< wplywik << endl << "Suma budzetu m
152   }
```

Agregacja. Przykład użycia Budzet.h.

```
7
8     class Budzet
9     {
10        Sprzet *sprz;
11        Marketing *mar;
12        Wplyw_M *wply;
13
14        int finanse;//Suma pieniedzy w budżecie, podawana w tysiacach
15        int suma_gat;//Suma  gat ekipy
16        int suma_rep;//Suma  rep ekipy
17        int suma_prof;//Suma  prof ekipy
18        int suma_roz;//Suma  roz ekipy
19        int finanse2;//Zmienna do porownania zysku/straty
20    public:
```

Mechanizm przyjaźni. Przykład w Marketing.h.

```
25      void informacje_marketingowe() const;
26      //Wyznaczanie i podzial budzetu marketingowe
27      void Rozdzial_budzetu_marketingowego(int bud
28      friend int Sedzia(Marketing mar, Wplyw_M m);
29      friend class Budzet;
30      // Zwracanie wartosci kupno
31      int get_kupno() const;
32      // Zwracanie wartosci sniad
33      int get_sniad() const;
34      // Zwracanie wartosci plakaty
35      int get_plakaty() const;
36      // Zwracanie wartosci trailer
37      int get_trailer() const;
38      // Zwracanie wartosci bilb
39      int get_bilb() const;
40      // Zwracanie wartosci internet
41      int get_internet() const;
42
43  };
44
45  class Wplyw_M {
46      //Zwiekszenie wplywu
47      int zwie;
48      //Pomniejszenie wplywu
49      int zmie;
50  public:
51      Wplyw_M(int Zwie, int Zmie);
52      Wplyw_M(int Zwie);
53      Wplyw_M();
54      friend int Sedzia(Marketing mar, Wplyw_M m);
55      ~Wplyw_M();
56      friend class Agregacja;
57      friend class Budzet;
58      // Zwracanie wartosci zwie
59      int get_zwie() const;
60      // Zwracanie wartosci zmie
61      int get_zmie() const;
62  };
63  //Porownanie marketingu z wplywami
64  int Sedzia(Marketing mar, Wplyw_M m);
```

Etap4:

Systemy nazw. Przykład użycia w Dod_funk.cpp i SymulatorStudia.cpp

```
3   //Przestrzen nazw dla funkcji
4   namespace Funkcje {
5       //Zastapienie liczb w kodzie nazwami
6       enum {
```

```
5   int main()
6   {
7       Funkcje::Glowne_menu();
8       return 0;
9   }
10
```

Przeciążanie operatorów. Przykład użycia w Budzet.cpp i Budzet.h

```
171     }
172     Budzet Budzet:: operator=(Budzet const& d) {
173         if (&d != this) {
174             delete wply;
175             delete mar;
176             delete sprz;
177             wply = d.wply;
178             mar = d.mar;
179             sprz = d.sprz;
180             finanse = d.finanse;
181             finanse2 = d.finanse2;
182             suma_gat = d.suma_gat;
183             suma_rep = d.suma_rep;
184             suma_prof = d.suma_prof;
185                 suma_roz = d.suma_roz;
186         }
187         return *this;
188     }
```

```
36      //Przeciazenie operatora wypisanie skladowych Budzet
37      friend ostream& operator<< (ostream& wyjscie, const Budzet& s);
38      //Przeciazenie operatora obliczanie finansow po odjeciu pieniedzy przeznaczonych na marketing
39      friend ostream& operator>> (ostream& wyjscie,  Budzet& s);
40      //Przeciazenie operatora, obliczanie wynikow finansowych produkcji
41      Budzet operator() (Budzet& s);
42      //Przeciazenie operatora, ukazujace, czy produkcja przyniosla zysk/strate
43      Budzet operator[](Budzet& s);
44      Budzet();
45      //Przeciazenie operatora przypisania
46      Budzet operator=(Budzet const& d);
```

Dziedziczenie i polimorfizm. Przykład użycia w Persona.h i Operator.h

```
9   protected:
10      int rep; //Reputacja osoby
11      int gat{}; //Umiejetnosc w konkretnych gatunkach
12      int pen;//Pensja
13      int roz;//Rozpoznawalnosc
14      int prof;//Profesjonalizm
15      string imie;
16      string nazwisko;
17  public:
18      static Baza* getbaze();
19      Persona()=default;
20      // Zwracanie wartosci gatunek
21      int get_gat() const;
22      // Podawanie wartosci gatunek
23      virtual void set_gat(int s)=0;
24      // Zwracanie wartosci reputacji
25      int get_rep() const;
26      // Podawanie wartosci reputacji
27      virtual void set_rep(int s)=0;
28      // Zwracanie wartosci pensja
29      int get_pen() const;
30      // Podawanie wartosci pensji
31      virtual void set_pen(int s)=0;
32      // Zwracanie wartosci rozpoznawalnosci
33      int get_roz() const;
34      // Podawanie wartosci rozpoznawalnosc
35      virtual void set_roz(int s)=0;
36      // Zwracanie wartosci profesjnalizm
37      int get_prof() const;
38      // Podawanie wartosci profesjonalizm
39      virtual void set_prof(int s)=0;
40      // Zwracanie wartosci imie
41      string get_imie() const;
42      // Podawanie wartosci imie
43      virtual void set_imie(string s)=0;
44      // Zwracanie wartosci nazwisko
45      string get_nazwisko()const;
46      virtual void set_nazwisko(string s) = 0;
47      virtual void stworz(int z) = 0;
48      //Wyswietla kandydata o podanym numerze
49      void Wyswietl(int z) ;
```

```
4   class Operator :
5       public Persona
6   {
7   public:
8       Operator() = default;
9       void set_gat(int s);
10      // Podawanie wartosci reputacji
11      void set_rep(int s);
12      // Podawanie wartosci pensji
13      void set_pen(int s);
14      // Podawanie wartosci rozpoznawalnosc
15      void set_roz(int s);
16      // Podawanie wartosci profesjonalizm
17      void set_prof(int s);
18      // Podawanie wartosci imie
19      void set_imie(string s);
20      //Podawanie wartosci nazwisko
21      void set_nazwisko(string s);
22      void stworz(int z);
```

Obiekt statyczny. Przykład użycia w Persona.h,Persona.cpp i Dod_funk.cpp.

```
5     class Baza;
6   class Persona
7   {
8       static Baza* baza;
9   protected:
10      int rep; //Reputacja osoby
11      int gat{}; //Umiejetnosc w konkretnych
12      int pen;//Pensja
13      int roz;//Rozpoznawalnosc
14      int prof;//Profesjonalizm
15      string imie;
16      string nazwisko;
17  public:
18      static Baza* getbaze();
```

```
Baza* Persona::getbaze() {
    if (!baza) {
        baza = new Baza;
    }
    return baza;
}

Baza* Persona::baza = Persona::getbaze();
```

```
214
215             if (i < ten) {
216
217                 wsk[i] = &z[i];
218                 wsk[i]->stworz(i);
219                 Persona::getbaze()->stworz(wsk[i]);
220                 Persona::getbaze()->wyswietl(i);
221                 cout << endl;
```

Etap5:

Wyeliminowanie tablic i zastąpienie kontenerami. Przykład użycia w Dod_funk.cpp

```
562         array<string, trzy> Tabelka2{ "Kamera","Mikrofon","Efekty specjalne" };
563         float a;
564         vector<int>Magazyn;
565         vector<int>Schowek;
566         vector<int>Wartosci_ekipy;
567         for (int i = zero; i < pietnascie; i++) {
568             Schowek.push_back(zero);
569             Magazyn.push_back(zero);
570
571         }
572         for (int i = zero; i < 4; i++) {
573             Wartosci_ekipy.push_back(zero);
574         }
575         for (int i = zero; i < pietnascie; i++) {
576
577
```

Użycie iteratorów, zwykłej pętli i pętli zakresowej. Przykłady wszystkie w Dod_funk.cpp.

```
895
896                 for (vector<int>::iterator it = Schowek.begin(); it != Schowek.end(); ++it)
897                     suma_elementow += *it;
```

```
394         array<Sprzet, piec> Golem;
395         do {
396             for (Sprzet v : Golem)
397             {
398                 cout << "Nr. " << i << " sprzetu" << endl;
399                 cout << endl;
400                 v.set_jakosc(i * i);
401                 v.get_jakosc();
402                 if (Tab[(rodzaj*piec)+i] != zero) {
403                     cout << "0.Koszt kupna: " << zero << endl << "1.Koszt wynajmu: " << zero << endl;
404                     cout << endl;
405                 }
406                 else {
407                     v.pokaz();
408                 }
409                 i++;
```

```
210         auto it = wsk.begin();
211         while(it!=wsk.end())
212         {
```


Autorstwo:

Wojciech Lidwin

123A