### 16.06.2017 Wrocław

### Wydział Elektronika

### Kierunek: Informatyka

### Nazwisko i Imię: Bocheńska Inga

### NR indeksu: 235062

### Grupa: Piątek godz. 15:15

### Semestr: 2016/17 LATO

### Prowadzący: mgr inż. Michał Koziarski

### PROGRAMOWANIE OBIEKTOWE

### Program semestralny w C++

### wersja Visual Studio *2015*

TEMAT

Schronisko dla zwierząt

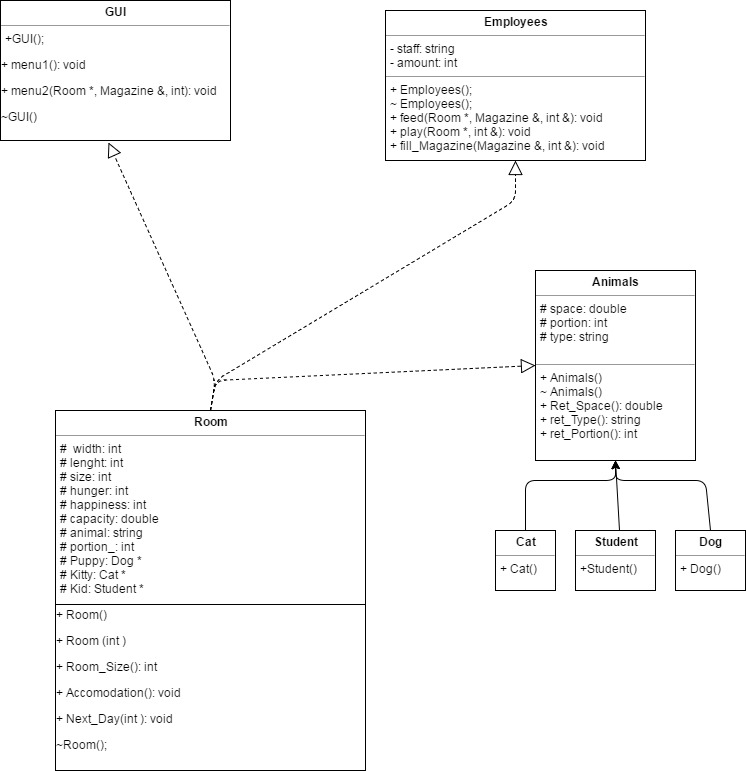
1. **Opis programu**

Program będzie zarządzał schroniskiem dla zwierząt. Zawarte będą różnego rodzaju boksy dla poszczególnych zwierząt. Schronisko zatrudnia pracowników, którzy mogą bawić się ze zwierzętami, karmić je oraz uzupełniać magazyn zamawiając wybraną karmę.

Będą przechowywane dane na temat ilości zwierząt z danych gatunków, boksów i zasobów. Na początku użytkownik może wybrać rozmiar boksów i jakie gatunki zwierząt trafią do których boksów (psy, koty czy studenci). Komunikaty będą informować o konieczności wykonania zadań. Będzie wyświetlana ilość zasobów i aktualny stan zwierząt.

Po 15 wykonanych czynnościach dzień dla schroniska będzie się kończył i aktualne potrzeby zwierząt będą zmieniane na kolejny dzień.

1. **Diagram UML**

****

1. **Kod klas**
2. **Klasa bazowa Animals**

#include <string>

#include <iostream>

using namespace std;

class Animals

{

protected:

double space =0;

int portion;

string type ="type of the animal";

public:

Animals();

double Ret\_Space();

string ret\_Type();

int ret\_Portion();

~Animals() {};

};

**Klasy pochodne: Dog, Cat, Student:**

class Dog : public Animals

{

public:

Dog();

};

class Cat : public Animals

{

public:

Cat();

};

class Student : public Animals

{

public:

Student();

};

1. **Room:**

#include <string>

#include <iostream>

#include <windows.h>

#include "Animals.h"

class Room

{

friend class Employees;

friend class GUI;

protected:

int width=0, lenght=0, size=0;

int hunger, happiness;

double capacity=0;

string animal;

int portion\_=0;

Dog \* Puppy;

Cat \* Kitty;

Student \* Kid;

public:

Room() {};

Room (int x);

int Room\_Size();

void Accomodation();

void Next\_Day(int i);

~Room();

};

1. **Employees**

#include <string>

#include <iostream>

#include <windows.h>

#include <time.h>

#include "Room.h"

struct Magazine

{

int Dog\_Food, Cat\_Food, Beer;

};

class Employees

{

protected:

string staff;

int amount;

public:

Employees();

void feed(Room \*r, Magazine &M, int &act);// w mainie będzie przekazana tablica pomieszczeń + sprawdzenie czy r nie jest za duże/za małe

void play(Room \*r, int &act);

void fill\_Magazine(Magazine &M, int &act);

~Employees() {};

};

1. **GUI:**

#include <iostream>

#include <iomanip>

#include "Room.h"

#include "Employees.h"

class GUI

{

public:

GUI();

void menu1();

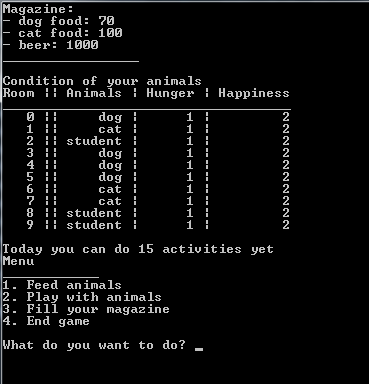
void menu2(Room \*r, Magazine &M, int act);

~GUI();

};

**Menu programu**

Program zawiera proste menu:



Po wcześniejszym wyborze wielkości pokoi i rodzajów zwierząt w nich mieszkających należy rozpocząć opiekę nad nimi. Wybierając aktywności z menu można nakarmić zwierzęta, pobawić się z nimi, uzupełnić magazyn wybranym pokarmem lub zakończyć grę.

Stan magazynu i zwierząt będzie się na bieżąco zmieniał w zależności od wykonanych czynności.

1. **Listing kodu**

**Room.cpp**

#include "Room.h"

Room::Room(int x)

{

this -> size = x;

hunger = 1;

happiness = 2;

}

int Room::Room\_Size()

{

cout << "Now you can choose rooms' size" << endl;

cout << "Width = "; cin >> width;

cout << "Lenght = "; cin >> lenght;

if (width <= 0 || lenght <= 0 || width > 100 || lenght > 100)

{

cout << "Wrong size chosen! My size = 25";

size = 25;

}

else

size = width\*lenght;

return size;

}

void Room:: Accomodation()

{

int a;

cout << "1 - Dogs" << endl << "2 - Cats" << endl << "3 - Students" << endl;

cin >> a;

switch (a)

{

case 1:

{

Dog Dog1;

animal = Dog1.ret\_Type();

portion\_ = Dog1.ret\_Portion();

capacity = size / Dog1.Ret\_Space();

Puppy = new Dog[(int)capacity];

break;

}

case 2:

{

Cat Cat1;

animal = Cat1.ret\_Type();

portion\_ = Cat1.ret\_Portion();

capacity = size / Cat1.Ret\_Space();

Kitty = new Cat[(int)capacity];

break;

}

case 3:

{

Student Student1;

animal = Student1.ret\_Type();

portion\_= Student1.ret\_Portion();

capacity = size / Student1.Ret\_Space();

Kid = new Student[(int)capacity];

break;

}

default:

{

cout << "Wrong number. I will fill this room with students" << endl;

Student Student1;

animal = Student1.ret\_Type();

portion\_ = Student1.ret\_Portion();

capacity = size / Student1.Ret\_Space();

Kid = new Student[(int)capacity];

}

Sleep(2000);

}

}

void Room:: Next\_Day(int i)

{

if (hunger >= 2)

happiness=1;

hunger++;

happiness--;

if (happiness <= 0)

cout << "Well, you suck at it - your animals in room " << i << " aren't happy" << endl;

}

Room:: ~Room()

{

delete[] Puppy;

delete[] Kitty;

delete[] Kid;

Puppy = 0;

Kitty = 0;

Kid = 0;

}

**GUI.cpp**

GUI::GUI()

{

}

void GUI:: menu1()

{

cout << "Menu" << endl;

cout << "\_\_\_\_\_\_\_\_\_\_\_\_" << endl;

cout << "1. Feed animals" << endl;

cout << "2. Play with animals" << endl;

cout << "3. Fill your magazine" << endl;

cout << "4. End game" << endl<<endl;

}

void GUI:: menu2(Room \*r, Magazine &M, int act)

{

system("cls");

cout << "Magazine: " << endl;

cout << "- dog food: " << M.Dog\_Food << endl;

cout << "- cat food: " << M.Cat\_Food << endl;

cout << "- beer: " << M.Beer << endl;

cout << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl << endl;

cout << "Condition of your animals" << endl;

cout << "Room || Animals | Hunger | Happiness" << endl;

cout << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl;

for (int i = 0; i < 10; i++)

{

cout <<setw(4)<< i <<" || "<< setw(7)<<r[i].animal<< " | " <<setw(6)<< r[i].hunger << " | "<<setw(9)<< r[i].happiness<<endl;

}

cout << endl;

cout << "Today you can do " << act << " activities yet" << endl;

}

GUI::~GUI()

{

}

**Employees.cpp**

Employees::Employees()

{

system("cls");

staff = "animal caretakers";

amount = 4;

cout << endl<< "You employ " << amount << " " << staff<<endl;

cout << "They are here to take care of animals" << endl <<endl;

Sleep(3000);

}

void Employees::feed(Room \*r, Magazine &M, int &act)

{

int number;

cout << "Animals in which room you want to feed? (0-9)" << endl;

cin >> number;

if (number < 0 || number >= 10)

{

cout << "Wrong number chosen! My random number = ";

srand(time(NULL));

number = rand() % 10;

cout << number << endl;

}

if (r[number].animal == "dog")

{

if (M.Dog\_Food < ((int)r[number].capacity)\*r[number].portion\_)

cout << "Not enough dog food in magazine. I recommend to fill it" << endl;

else

{

M.Dog\_Food -= ((int)r[number].capacity)\*r[number].portion\_;

r[number].hunger = 0;

act--;

cout << "Animals feeded" << endl;

}

}

else if (r[number].animal == "cat")

{

if (M.Cat\_Food < ((int)r[number].capacity)\*r[number].portion\_)

cout << "Not enough cat food in magazine. I recommend to fill it" << endl;

else

{

M.Cat\_Food -= ((int)r[number].capacity)\*r[number].portion\_;

r[number].hunger = 0;

act--;

cout << "Animals feeded" << endl;

}

}

else if (r[number].animal == "student")

{

if (M.Beer < ((int)r[number].capacity)\*r[number].portion\_)

cout << "Not enough beer in magazine. I recommend to fill it" << endl;

else

{

M.Beer -= ((int)r[number].capacity)\*r[number].portion\_;

r[number].hunger = 0;

act--;

cout << endl << "Animals feeded" << endl;

}

}

else

cout << "Empty cage" << endl;

Sleep(2000);

}

void Employees::play(Room \*r, int &act)

{

int number;

cout << "Animals in which room you want to play with? (0-9)" << endl;

cin >> number;

if (number < 0 || number >= 10)

{

cout << "Wrong number chosen! My random number = ";

srand(time(NULL));

number = rand() % 10;

cout << number << endl;

}

r[number].happiness = 3;

cout << "Played." << endl;

act--;

Sleep(2000);

}

void Employees::fill\_Magazine(Magazine &M, int &act)

{

int a;

cout << "Which resources do you want to order from supplier?"<<endl;

cout << "1 - DogFood\n2 - CatFood\n3 - Beer"<<endl;

cin >> a;

switch (a)

{

case 1: M.Dog\_Food += 35; break;

case 2: M.Cat\_Food += 35; break;

case 3: M.Beer += 80; break;

default: cout << "Wrong choise" << endl; Sleep(1500); act++;

}

act--;

}

**Animals.cpp**

Animals::Animals()

{

}

double Animals:: Ret\_Space()

{

return space;

}

string Animals::ret\_Type()

{

return type;

}

int Animals::ret\_Portion()

{

return portion;

}

Dog::Dog()

{

space = 4;

portion = 2;

type = "dog";

}

Cat::Cat()

{

space = 1;

portion = 1;

type = "cat";

}

Student::Student()

{

space = 0.5;

portion = 20;

type = "student";

}

**Source.cpp**

#include "Animals.h"

#include "Employees.h"

#include "Room.h"

#include "GUI.h"

#include <string>

#include <iostream>

#include <windows.h>

using namespace std;

int main()

{

int Size, activities = 15, choise, day=1;

GUI Gui;

Magazine Magazine1{ 70, 100, 1000 };

cout << "Welcome to your Animal Shelter. Take good care of your animals" << endl;

Sleep(3000);

Room \*Room\_ = new Room[10];

Size = Room\_[0].Room\_Size();

for (int i = 0; i < 10; i++)

{

system("cls");

Room\_[i] = \*new Room(Size);

cout << "What species do you want to accomodate in room " << i << " ?" << endl;

Room\_[i].Accomodation();

}

Employees Employees;

do {

system("cls");

cout << "It is day " << day << endl;

Sleep(4000);

for (activities = 15; activities > 0;)

{

Gui.menu2(Room\_, Magazine1, activities);

Gui.menu1();

cout << "What do you want to do? ";

cin.clear();

cin.sync();

cin >> choise;

switch (choise)

{

case 1:

{

Employees.feed(Room\_, Magazine1, activities);

break;

}

case 2:

{

Employees.play(Room\_, activities);

break;

}

case 3:

{

Employees.fill\_Magazine(Magazine1, activities);

break;

}

case 4:

{

cout << "Game over" << endl;

Sleep(2000);

return 0;

}

default:

{

cout << "Wrong naumber, try again" << endl;

Sleep(2000);

}

}

}

day++;

for (int i = 0; i < 10; i++)

{

Room\_[i].Next\_Day(i);

}

} while (day<30);

return 0;

}

1. **Wnioski**

Udało mi się stworzyć uproszczoną wersję symulacji działania schroniska dla zwierząt. W wielu funkcjach domyślne dane są tworzone w prosty sposób, co minimalizuje możliwości wprowadzenia złych danych przez użytkownika. Można dodać na przykład warunkową śmierć zwierząt (czyszczenie pokoju), gdy długo będą one nie karmione, jednak uznajmy, że to wersja dla dzieci.