Лабораторное занятие 9
Создание иерархически связанных классов

1 Цель работы

1.1 Приобрести навыки по составлению сложных алгоритмов поиска и ввода-вывода.

2 Литература

2.1 Фленов, М.Е. Библия C#. – 3 изд.– Санкт-Петербург: БХВ-Петербург, 2016. – Режим доступа: https://ibooks.ru/reading.php?productid=353561, только для зарегистрированных пользователей. – Загл. с экрана. – п.5.7.

3 Подготовка к работе

3.1 Повторить теоретический материал (см. п.2).

3.2 Изучить описание практической работы.

4 Основное оборудование

4.1 Персональный компьютер.

5 Задание

Все задания выполняются в одном проекте! Чтоб не потерять код, при создании проекта выберите папку C:\Temp\КСК-31

5.1 Создать абстрактный класс Entity, содержащий только две открытые чистые виртуальные функции getSprite и setSprite.

5.2 Создать класс Zombie, дочерний для класса Entity. Добавить открытые свойства string name, int health, int hit, char sprite, int x, int y. Переопределить функционал метода getSprite так, чтоб она возвращала значение свойства sprite.

5.3 Добавить в класс Zombie конструктор, принимающий два аргумента (x, y). Остальным полям выставить значения по умолчанию, по своему усмотрению.

5.4 Добавить в код класс Player, наследник класса Entity. Добавить открытые свойства string name, int health, int hit, char sprite, int x, int y. Переопределить функционал метода getSprite так, чтоб она возвращала значение свойства sprite. Добавить конструктор, принимающий два аргумента (x, y). Остальным полям выставить значения по умолчанию, по своему усмотрению.

С классами покончили. Остальные функции создаются вне классов.

5.5 Внутри функции main создать экземпляр класса Player со значениями x = 2, y = 1. Там же создать экземпляр класса Zombie со значениями x = 27, y = 15. На этом моменте мы установим положение персонажей на экране.

5.6 Персонажи должны передвигаться по какой-то карте или плоскости. Проще всего (а может и нет) это будет реализовать с помощью массива символов. Добавьте в код вне main следующий массив:

```
const int mapSizeX = 29;
const int mapSizeY = 17;
char map[mapSizeY][mapSizeX] = {
        {'X', 'X','X','X','X','X','X','X','X','X','X','X','X','X','X','X','X','X','X','X','X','X','X','X','X','X','X'},
        {' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ','X'},
        {'X',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ','X'},
        {'X',' ',' ',' ','X','X','X','X','X','X','X',' ',' ',' ','X',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ','X'},
        {'X',' ',' ',' ','X',' ',' ',' ',' ',' ','X',' ',' ',' ','X',' ',' ',' ','X','X','X','X','X','X','X',' ',' ','X'},
```

```
	{'X',' ',' ',' ','X',' ',' ',' ','X','X','X',' ',' ',' ','X',' ',' ',' ','X',' ',' ',' ',' ',' ',' ','X',' ',' ','X'},
	{'X',' ',' ',' ','X',' ',' ',' ',' ',' ',' ',' ',' ',' ','X',' ',' ',' ','X',' ',' ',' ',' ',' ',' ','X',' ',' ','X'},
	{'X','X','X','X','X',' ',' ',' ','X','X','X','X','X','X','X',' ',' ',' ','X',' ',' ',' ',' ',' ',' ','X',' ',' ','X'},
	{'X',' ',' ',' ','X',' ',' ',' ','X',' ',' ',' ',' ','X',' ',' ',' ',' ','X',' ',' ',' ',' ',' ',' ',' ',' ',' ','X'},
	{'X',' ',' ',' ','X',' ',' ',' ','X',' ',' ',' ',' ','X',' ',' ',' ',' ','X',' ',' ',' ',' ',' ',' ',' ',' ',' ','X'},
	{'X',' ',' ',' ','X',' ',' ',' ',' ',' ',' ',' ',' ','X',' ',' ',' ',' ','X',' ',' ',' ',' ',' ',' ',' ',' ',' ','X'},
	{'X',' ',' ',' ','X',' ',' ',' ','X',' ',' ',' ',' ','X',' ',' ',' ',' ','X',' ',' ','X',' ',' ',' ',' ',' ',' ','X'},
	{'X',' ',' ',' ','X',' ',' ',' ','X',' ',' ',' ',' ',' ',' ',' ',' ',' ','X',' ',' ','X',' ',' ',' ',' ',' ',' ','X'},
	{'X',' ',' ',' ','X',' ',' ',' ','X',' ',' ',' ',' ','X','X','X','X','X','X',' ',' ','X',' ',' ',' ',' ',' ',' ','X'},
	{'X',' ',' ',' ','X','X','X','X','X',' ',' ',' ',' ','X',' ',' ',' ',' ',' ',' ',' ','X',' ',' ',' ',' ',' ',' ','X'},
	{'X',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' '},
	{'X', 'X','X','X','X','X','X','X','X','X','X','X','X','X','X','X','X','X','X','X','X','X','X','X','X','X','X','X','X'}
};
```

Символ X (именно латиница капсом) – это преграда или стена, через нее персонаж не сможет пройти.

5.7 Карту нужно вывести на консоль, для этого создадим отдельный метод:

```
//принимает по ссылке принимаем игрока и вампира
void drawMap(Player* player, Vampire*	vampire) {
	//перед отрисовкой кадра очищаем консоль
	system("cls");
	string frame;
	//выводим карту(по факту просто двумерный массив)
	//вывод строк
	for (int y = 0; y < mapSizeY; y++) {
		//вывод столбцов
		for (int x = 0; x < mapSizeX; x++) {
			//если x и y совпадает с координатами персонажа
			if (y == player->y && x == player->x)
			{
				//вывод спрайта персонажа на консоль + покраска с зеленый
				cout << "\033[32m" << player->getSprite() << "\033[0m";
			}
			//иначе если x и y совпадает с координатами вампира
			else if (y == vampire->y && x == vampire->x)
			{
				//выводим вампира, он будет красный
				cout << "\033[31m" << vampire->getSprite() << "\033[0m";
			}
			//иначе, просто отрисовываем символ пробела или X синего цвета
			else {
				cout <<"\033[96m"<<map[y][x]<<"\033[0m";
			}
		}
		cout << endl;
	}

}
```

5.8 Добавить в main бесконечный цикл while(true). Вызвать внутри функцию drawMap(&player, &vampire); //ваши объекты могут называться по-другому

5.9 В следующей строчке добавьте char input = getch();

Мы будем считывать нажатый символ и в соответствии с нажатой клавишей будем передвигать персонажа. Поэтому в следующей строчке можете Добавить вызов функции movePlayer(input, &player);

Листинг функции movePlayer:

```
void movePlayer(char direction, Player* player) {
	switch (direction) {
	case 'w':
		if (player->y > 0 && map[player->y - 1][player->x] != 'X') {
			player->y--;
		}
		break;
		// ... аналогично для других направлений

	case 's':
		if (player->y < mapSizeY - 1 && map[player->y + 1][player->x] != 'X') {
			player->y++;
		}
		break;
	case 'a':
		if (player->x > 0 && map[player->y][player->x - 1] != 'X') {
			player->x--;
		}
		break;
	case 'd':
		if (player->x < mapSizeX - 1 && map[player->y][player->x + 1] != 'X') {
			player->x++;
		}
		break;
	}
}
```

Обратите внимание на то, что при использовании символов в верхнем регистре или кириллицы, персонаж двигаться не будет. При желании расширьте функциональность.

5.10 Дальше добавим возможность сражения между персонажем и вампиром.

Создадим метод isColliding, который будет проверять, произошла ли коллизия (столкновение персонажа и вампира). Листинг функции:

```
//принимает координаты x и y персонажа и врага
bool isColliding(int x1, int y1, int x2, int y2) {
	//возвращает истну, если коллизия произошла и ложь в ином случае
	return x1 == x2 && y1 == y2;
}
```

5.11 Добавим функцию для сражения, которая будет срабатывать, если персонажи столкнулись:

```
//принимает объекты по ссылке чтоб напрямую поменять значения их свойств
void battle(Player* player, Vampire* enemy) {
	//пока у обоих здоровье больше нуля
	while (player->health > 0 && enemy->health > 0) {
		//сначала уменьшаем здоровье врага на силу удара игрока
		enemy->health -= player->hit;
		//затем зомби наносит урон
		player->health -= enemy->hit;
		//вывод информации
		cout << "Ваш ход. Вы нанесли " << player->hit << " урона." << endl;
		cout << "Ход врага. Вам нанесли " << enemy->hit << " урона." << endl;
		//задержка в пол секунды для имитации борьбы
		Sleep(500);
	}
	//если у врага не осталось здорвья
	if (enemy->health == 0)
	{
		cout << "Вы победили!";
		cout << "Осталось здоровья " << player->health;
		//меняем спрайт на пробел, пока так чтоб не усложнять
		//и чтоб враг скрылся на карте
```

```
                enemy->sprite = ' ';
                getch();
        }
        //если у игрока не осталось здоровья
        else
        {
                cout << "Вы проиграли!";
                //о нет... игрок исчез и умер
                player->sprite = ' ';
                getch();
        }

}
```

5.12 Все в тот же бесконечный цикл добавьте проверку на столкновение и в случае, если оно произошло сражения

```
if (isColliding(p.x, p.y, enemy.x, enemy.y)) {
                        battle(&p, &enemy);

                }
```

5.13 Запустите код, можете подумать о том как улучшить игру. Главное чтоб она работала...

6 Порядок выполнения работы

6.1 Запустить Visual Studio и выполнить все задания из п.5.
6.2 Ответить на контрольные вопросы.
6.3 Составить электронный отчет и сохранить C:\Temp\KSK\

7 Содержание отчета

7.1 Титульный лист
7.2 Цель работы
7.3 Ответы на контрольные вопросы
7.4 Вывод

8 Контрольные вопросы

8.1 Что такое ООП?
8.2 Назовите основные парадигмы ООП.
8.3 Для чего создаются конструкторы?

9. Приложение

X:\Абрамова\КСК\ОАиП\Лекции\Абстрактные классы.odp