
**СУ “Св. Климент Охридски”,**
**ФМИ – Софтуерно инженерство**
**Курсов проект по Обектно-ориентирано програмиране**

# Вектор

Веселина Валериева Стоянова, Факултетен № 61794

# Съдържание

1. Въведение ..... 2
2. Описание на приложените алгоритми ..... 2
3. Описание на програмния код ..... 2
4. Използвани технологии ..... 7

## 1. Въведение

Целта на проекта е да бъде реализиран шаблонен клас за динамичен масив – Vector  и да бъде показана примерна употреба на реализирания клас.  В тази документация ще разгледаме кратко описание на програмния код, което включва: основните методи използвани в проекта Vector и примерна употреба, и използвани техники за създаване на проекта.

Кодът на проекта е качен и в Github:

## 2. Описание на приложените алгоритми

## 3. Описание на програмния код

Стуктурата на класа  е представена по долу:

```
template<typename T>
class Vector
{
public:
	Vector();
	Vector(size_t count);
	Vector(size_t count, const T& value);
	Vector(const Vector& other);
	Vector& operator=(const Vector& other);
	T& operator[](size_t index)
	{
		return data[index];
	}
	size_t capacity() const
	{
		return capacity_;
```

```
        }
        size_t size() const
        {
                return size_;
        }
        void reserve(size_t newCapacity);
        void resize(size_t newSize);
        T& back();
        T& front();
        void shrink_to_fit();
        void swap(Vector<T> &other);
        void insert(size_t index, const T& value);
        void erase(size_t index);
        void print() const;
        void push_back(const T& value);
        void pop_back();
        bool empty();
        void clear();
        ~Vector();
private:
        size_t capacity_;
        size_t size_;
        T* data;
```

Създаден е шаблонен клас Vector, в който са имплементирани различни методи, чрез които се реализира векторите като изключително удобен и ефективен заместител на стандартните масиви.

Програмният код се състои от .h файл, в които се декларира шаблонен клас Vector и .cpp файл, в който са дефинициите на функциите. Също така има и .h файл, който е създаден клас Animal и който създава животни с техните характеристики, чрез които се показва употребата на методите на вектора.

В класа Vector са дефинирани:

1) Конструктор по подразбиране: Vector();

Създава празен вектор с капацитет 0 и брой елементи 0.

2) Конструктор за запълване: Vector(size_t count);

Създава вектор с капацитет count и брой елементи count.

3) Конструктор за запълване: Vector(size_t count, const T& value);

Създава вектор с капацитет cout и брой елементи cout.

(value - стойност с която да се запълни векторът)

4) Конструктор за копиране: Vector(const Vector& other);

Създава вектор, който съдържа копие на всеки от елементите на other.

5)Оператор за присвояващо копиране: Vector& operator=(const Vector& other);

Копира данните от векторът other.

Заделянето на памет се извършва чрез помощна функция: allocateMemory(size_t capacityAlloc): Заделянето на паметта става чрез operator new като връща указател към нея.

Добавени са и други помощни функции:

copyRange(T* begin, T* end, T* destination): Копира елементите от посочения обхват (опеределн от begin и end) в destination.

constructRange(T* begin, T* end): Конструира с конструктор по подразбиране обектите върху паметта , определена от обхвата begin и end.

constructRange(T* begin, T* end, const T& fillValue): Конструира с конструктор, приемащ стойност fill_value ,обектите върху паметта, определена от обхвата begin и end.

deleteRange(T* begin, T* end):Извиква деструкторите на обектите върху паметта, определна от обхвата begin и end.

## *Описание на основните функции на класа Vector:*

Функцията push_back променя размера на вектора, добавяйки един елемент със стойност value след текущия последен елемент .

Друга член-функция pop_back отстранява последния елемент на вектор, като намалява размера му с единица. Забележете, че функцията pop_back не връща отстранения елемент.

Функциите могат да модифицират вектор. Могат да изтрият елементите на вектор, да се преоразмеряват или да се модифицират отделни елементи.

Функцията resize() и shrink_to_fit() преоразмеряват вектора. Resize променя размера на вектора, така, че да съдържа нов брой елемента, а shrink_to_fit свива вектора, така,че размера да се побере в капацитета.

Erase() изстрива елемент на позиция index, а insert вмъква елемент със стойност value на позиция index.

Функцията swap ( ) , разменя стойностите на двата аргумента . Swap ( ) трябва да бъде извикана със адреса на аргументите .

Функцията empty() проверява дали векторът е празен. Функцията не модифицира. За изчистване на съдържанието на вектора се използва функцията clear().

Функциите, които връщат референции са съответно front(),back(), operator[], където съответно: back()-връща референция към елемента в края на вектора,back()-връща елемента в началото на вектора,Operator[]връща референция към елемента на позиция index.

Член-променливи на класа Vector:

size_t capacity_-Капацитет на вектора

size_t size_-Размер на вектора

## *Примерна употреба:*

```
int main()
{
        Animal a1("Mammal", "Lion", "Grassland", 5);
        Animal a2("Bird", "Pengiun", " Antarctica", 1);
        Animal a3("Reptiles", "Snake", "Desert", 2);
        Animal a4("Aquatic Animal", "Dolphin", "Sea", 7);
        Vector<Animal> animals;
        animals.push_back(a1);
        animals.push_back(a2);
        animals.push_back(a3);
        animals.push_back(a4);
        Vector<int> v1;
        v1.push_back(1);
```

```
v1.push_back(3);
v1.push_back(5);
v1.push_back(7);
v1.push_back(9);
v1.push_back(11);
v1.push_back(13);
v1.pop_back();
v1.push_back(777);
v1.insert(5, 100);
v1.erase(6);
v1.print();
cout << endl;
cout << v1[4] << endl;

Vector<string> Penguin;
Penguin.push_back("Penguins");
Penguin.push_back("were once");
Penguin.push_back("able to");
Penguin.push_back("fly");
for (int i = 0; i < Penguin.size(); i++)
{
        cout << Penguin[i] << " ";
}
cout << endl;
cout << Penguin.front()<<" "<<"don't"<<" "<< Penguin.back() << endl;

Vector<string> Lion;
Lion.push_back("Lion");
Lion.push_back("is common symbols");
Lion.push_back(" for royalty and stateliness");
Lion.insert(1, "The");
for (int i = 0; i < Lion.size(); i++)
{
        cout << Lion[i] << " ";
}
cout << endl;
Lion.erase(1);
```

```
	for (int i = 0; i < Lion.size(); i++)
	{
		cout << Lion[i] << " ";
	}
	cout << endl;

	Vector<string> Snake;
	Snake.push_back("Snakes");
	Snake.push_back("are");
	Snake.push_back("dreadful");

	Vector<string> Dolphin;
	Dolphin.push_back("Dolphins");
	Dolphin.push_back("are");
	Dolphin.push_back("lovely");

	Snake.swap(Dolphin);
	for (int i = 0; i < Snake.size(); i++)
	{
		cout << Snake[i] << " ";
	}
	cout << endl;
	for (int i = 0; i < Dolphin.size(); i++)
	{
		cout << Dolphin[i] << " ";
	}
	cout << endl;

	return 0;
}
```

## 4. Използвани технологии

Интегрираната среда за разработка, която съм използвала за моя проект е Microsoft Visual Studio 2013. Проектът е писан на програмния език C++.