

Лабораторная работа №2

Класс векторов

Демидова Екатерина Алексеевна

Содержание

1	Цель работы	4
2	Задание	5
3	Выполнение проекта	6
4	Выводы	13

Список иллюстраций

3.1	Создание вектора и его печать	8
3.2	Вычитание и сложение векторов	10
3.3	Скалярное умножение	11
3.4	Унарный минус	11
3.5	Присваивание	12

1 Цель работы

Написать на C++ класс векторов и программу для работы с этим классом.

2 Задание

Написать программу на C ++, которая реализует Класс Vector. Класс Vector должен иметь следующие поля private :

- Размерность вектора
- Массив значений вектора
- порядковый номер вектора

Класс Vector должен иметь следующие поля public:

- Количество созданных векторов (static)

Необходимо реализовать следующие функции или методы класса:

- Конструктор класса
- Деструктор
- Функция отображения вектора и его номера(print)

Оператор функции:

- сложения / вычитания векторов
- унарный минус
- скалярное умножение
- присваивание

3 Выполнение проекта

Private-методы и поля класса определяют его реализацию. Доступ к ним разрешен только из методов данного класса. Были объявлены private-поля класса Vector, а именно arr - массив значений вектора, n - текущее количество точек вектора, capacity - возможное количество точек.

Public-методы и поля класса определяют его интерфейс, доступ к ним возможен из любой части кода. Был создан конструктор класса, в нём массив значений вектора задаётся по умолчанию длины 1, так как далее память под него будет выделяться динамически.

```
Vector(){  
    arr = new int[1];  
    capacity = 1;  
    n = 0;  
  
}
```

Кроме того был создан деструктор класса.

```
~Vector() {  
    delete[] arr;  
}
```

Также была реализована функция добавления точки, она выделяет память под точку и добавляет её в конец массива значений:

```

void push(int data){

    if (n == capacity){

        int* tmp = new int[2*capacity];

        for (int i = 0; i < capacity; i++) {
            tmp[i] = arr[i];
        }

        delete[] arr;
        capacity *= 2;
        arr = tmp;
    }
    arr[n] = data;
    n++;

} ;

```

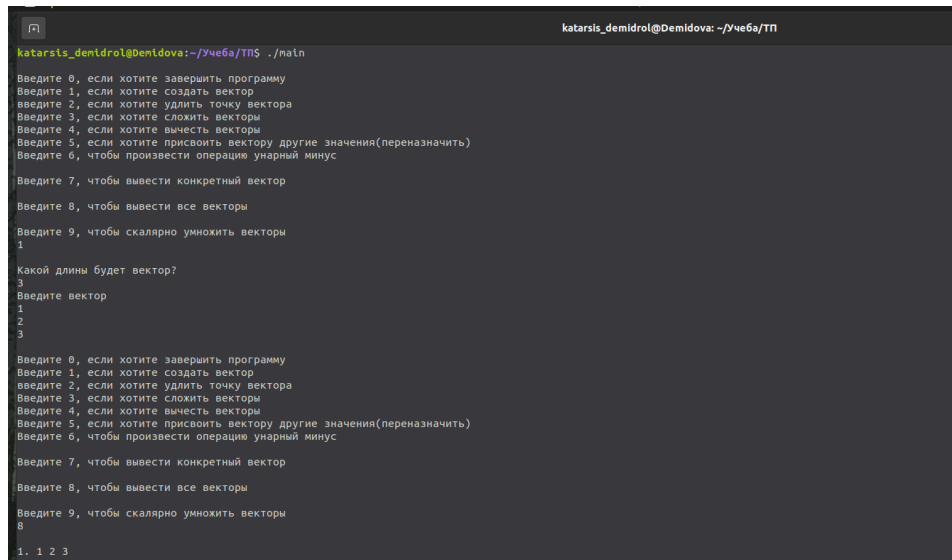
Была написана функция для печати вектора.

```

void print()
{
    for (int i = 0; i < n; i++) {
        cout << arr[i] << " ";
    }
    cout << endl;
}

```

Приведём пример использования этих функций. (рис. 3.1)



```
katarsts_demidrol@Demidova: ~/Учеба/ТН$ ./main
Введите 0, если хотите завершить программу
Введите 1, если хотите создать вектор
Введите 2, если хотите удалить точку вектора
Введите 3, если хотите сложить векторы
Введите 4, если хотите вычесть векторы
Введите 5, если хотите присвоить вектору другие значения(переназначить)
Введите 6, чтобы произвести операции унарный минус
Введите 7, чтобы вывести конкретный вектор
Введите 8, чтобы вывести все векторы
Введите 9, чтобы скалярно умножить векторы
1
Какой длины будет вектор?
3
Введите вектор
1
2
3
Введите 0, если хотите завершить программу
Введите 1, если хотите создать вектор
Введите 2, если хотите удалить точку вектора
Введите 3, если хотите сложить векторы
Введите 4, если хотите вычесть векторы
Введите 5, если хотите присвоить вектору другие значения(переназначить)
Введите 6, чтобы произвести операции унарный минус
Введите 7, чтобы вывести конкретный вектор
Введите 8, чтобы вывести все векторы
Введите 9, чтобы скалярно умножить векторы
8
1. 1 2 3
```

Рис. 3.1: Создание вектора и его печать

Механизм переопределения действия большинства операций C++ в отношении объектов классов – описание оператор-функций. При перегрузке операций сохраняется количество операндов, приоритеты выполнения и правила ассоциации. Все операторы написаны вне класса и объявлены внутри него.

Были перегружены операторы сложения/вычитания. Они поэлементно проводят операцию над векторами, проверяя совпадение длины, и возвращают результирующий вектор.

```
Vector Vector::operator + (Vector &r)
{ Vector res;
  if(this->n != r.n)
  {
    exit(1);
  }
  else{
    for(int i = 0; i < r.n; i++){
      res.push(r.arr[i] + this->arr[i]);
    }
  }
}
```



```

    }
    return res;
}

Vector Vector::operator - (Vector &r)
{ Vector res;
  if(this->n != r.n)
  {
    exit(1);
  }
  else{
    for(int i = 0; i < r.n; i++){
      res.push(this->arr[i]-r.arr[i]);
    }
  }
  return res;
}

```

Приведём пример использования этих операторов. (рис. 3.2)

```

kataris_demidrol@Demidova: ~/Челб/ТП
Введите 6, чтобы произвести операцию унарный минус
Введите 7, чтобы вывести конкретный вектор
Введите 8, чтобы вывести все векторы
Введите 9, чтобы скалярно умножить векторы
3
Введите номер первого вектора
1
Введите номер второго вектора
2
Сумма равна:
3 5 4
Введите 0, если хотите завершить программу
Введите 1, если хотите создать вектор
Введите 2, если хотите удалить точку вектора
Введите 3, если хотите сложить векторы
Введите 4, если хотите вычесть векторы
Введите 5, если хотите присвоить вектору другие значения(переназначить)
Введите 6, чтобы произвести операцию унарный минус
Введите 7, чтобы вывести конкретный вектор
Введите 8, чтобы вывести все векторы
Введите 9, чтобы скалярно умножить векторы
4
Введите номер первого вектора
2
Введите номер второго вектора
1
Разность равна:
1 1 -2
Введите 0, если хотите завершить программу
Введите 1, если хотите создать вектор
Введите 2, если хотите удалить точку вектора
Введите 3, если хотите сложить векторы
Введите 4, если хотите вычесть векторы
Введите 5, если хотите присвоить вектору другие значения(переназначить)
Введите 6, чтобы произвести операцию унарный минус
Введите 7, чтобы вывести конкретный вектор
Введите 8, чтобы вывести все векторы
Введите 9, чтобы скалярно умножить векторы
8
1. 1 2 3
2. 2 3 1

```

Рис. 3.2: Вычитание и сложение векторов

Были реализованы оперторы унарный минус и скалярное умножение векторов:

```
Vector Vector::operator - (Vector &r)
```

```
{ Vector res;
    if(this->n != r.n)
    {
        exit(1);
    }
    else{
        for(int i = 0; i < r.n; i++){
            res.push(this->arr[i]-r.arr[i]);
        }
    }
    return res;
}
```

```
int Vector::operator * (Vector &r){
```

```

int res=0;
for(int i =0; i< r.n; i++){
res += (this->arr[i])*r.arr[i];
}
return res;
}

```

Приведём пример использования этих операторов. (рис. 3.3, 3.3)

```

katarsis_demidrol@Demidova: ~/Учеба/ТП
Введите 8, чтобы вывести все векторы
Введите 9, чтобы скалярно умножить векторы
8
1. 1 2 3
2. 2 3 1
Введите 0, если хотите завершить программу
Введите 1, если хотите создать вектор
Введите 2, если хотите удалить точку вектора
Введите 3, если хотите сложить векторы
Введите 4, если хотите вычесть векторы
Введите 5, если хотите присвоить вектору другие значения(переназначить)
Введите 6, чтобы произвести операцию унарный минус
Введите 7, чтобы вывести конкретный вектор
Введите 8, чтобы вывести все векторы
Введите 9, чтобы скалярно умножить векторы
6
Введите порядковый номер вектора
1
Введите 0, если хотите завершить программу
Введите 1, если хотите создать вектор
Введите 2, если хотите удалить точку вектора
Введите 3, если хотите сложить векторы
Введите 4, если хотите вычесть векторы
Введите 5, если хотите присвоить вектору другие значения(переназначить)
Введите 6, чтобы произвести операцию унарный минус
Введите 7, чтобы вывести конкретный вектор
Введите 8, чтобы вывести все векторы
Введите 9, чтобы скалярно умножить векторы
7
Введите порядковый номер вектора
1
-1 -2 -3

```

Рис. 3.3: Скалярное умножение

```

katarsis_demidrol@Demidova: ~/Учеба/ТП
Введите 0, если хотите завершить программу
Введите 1, если хотите создать вектор
Введите 2, если хотите удалить точку вектора
Введите 3, если хотите сложить векторы
Введите 4, если хотите вычесть векторы
Введите 5, если хотите присвоить вектору другие значения(переназначить)
Введите 6, чтобы произвести операцию унарный минус
Введите 7, чтобы вывести конкретный вектор
Введите 8, чтобы вывести все векторы
Введите 9, чтобы скалярно умножить векторы
9
Введите номер первого вектора
1
Введите номер второго вектора
2
Произведение равно:
-11

```

Рис. 3.4: Унарный минус

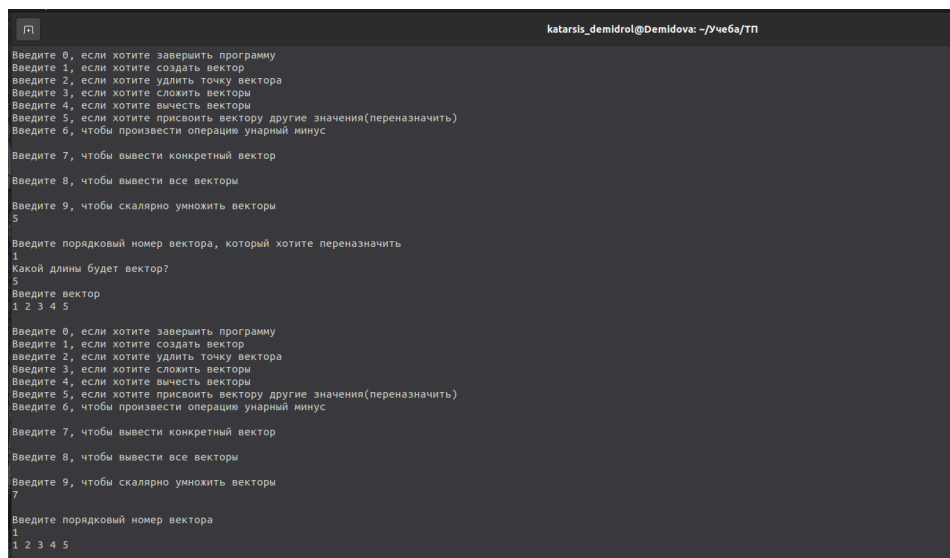
Также был переопределён оператор присваивания:

```

Vector& Vector::operator = (const Vector &r)
{ delete [] arr;
  this->n = r.n;
  arr=new int[n];
  for (int i = 0; i < n; i++) this->arr[i] = r.arr[i];
  return *this;
}

```

Приведём пример его использования. (рис. 3.5)



```

kataris_demidrol@Demidova: ~/Учеба/ТП
Введите 0, если хотите завершить программу
Введите 1, если хотите создать вектор
Введите 2, если хотите удалить точку вектора
Введите 3, если хотите сложить векторы
Введите 4, если хотите вычесть векторы
Введите 5, если хотите присвоить вектору другие значения(переназначить)
Введите 6, чтобы произвести операцию унарный минус

Введите 7, чтобы вывести конкретный вектор
Введите 8, чтобы вывести все векторы
Введите 9, чтобы скалярно умножить векторы
5
Введите порядковый номер вектора, который хотите переназначить
1
Какой длины будет вектор?
5
Введите вектор
1 2 3 4 5

Введите 0, если хотите завершить программу
Введите 1, если хотите создать вектор
Введите 2, если хотите удалить точку вектора
Введите 3, если хотите сложить векторы
Введите 4, если хотите вычесть векторы
Введите 5, если хотите присвоить вектору другие значения(переназначить)
Введите 6, чтобы произвести операцию унарный минус

Введите 7, чтобы вывести конкретный вектор
Введите 8, чтобы вывести все векторы
Введите 9, чтобы скалярно умножить векторы
7
Введите порядковый номер вектора
1
1 2 3 4 5

```

Рис. 3.5: Присваивание

4 Выводы

В результате выполнения лабораторной работы были получены практические навыки работы с классами, была написана программа на языке C++, в которой реализован класс для создания и работы с векторами, а также программа, демонстрирующая возможности этого класса.