Московский авиационный институт
(Национальный исследовательский университет)

Факультет: «Информационные технологии и прикладная математика»
Кафедра: 806 «Вычислительная математика и программирование»
Дисциплина: «Объектно-ориентированное программирование»

Лабораторная работа № 7

Тема: Программирование классов на языке C++

Студент: Бронников М. А.
Группа: 80-204
Вариант: 10
Преподаватель: Чернышов Л.Н.
Дата:
Оценка:

Москва, 2018

Целью лабораторной работы является:

1) Создание сложных динамических структур данных.

2) Закрепление принципа OCP.

Задание :

Необходимо реализовать динамическую структуру данных – «Хранилище объектов» и алгоритм работы с ней. «Хранилище объектов» представляет собой массив.

Каждым элементом контейнера, в свою очередь, является динамическая структура - массив.

То есть в моем варианте –массив в массиве.

Элементом второго контейнера является объект-фигура, определенная вариантом задания.

При этом должно выполняться правило, что количество объектов в контейнере второго уровня не больше 5. Т.е. если нужно хранить больше 5 объектов, то создается еще один контейнер второго уровня.

При удалении объектов должно выполняться правило, что контейнер второго уровня не должен быть пустым. Т.е. если он становится пустым, то он должен удалится.

Нельзя использовать:

1. Стандартные контейнеры std.

Программа должна позволять:

1. Вводить произвольное количество фигур и добавлять их в контейнер.
2. Распечатывать содержимое контейнера (1-го и 2-го уровня).
3. Удалять фигуры из контейнера по критериям:
    a) По типу (например, все квадраты).

1. Решения задачи

Программа содержит класс Figure, который является абстрактным.
Наследники класса Figure: Triangle, Rectangle, Quadrate, содержат перегрузки операторов ввода и вывода, проверки ввода, оператора сравнения элементов.
Наследником классов фигур Triangle, Rectangle, Quadrate является класс Item, содержащий методы работы со связным списком: поиск элемента, добавление, удаление, получение значения элемента.
Класс Massive наследник класса Item, использует методы, описанные в Item для работы с массивом.
Класс Item также является наследником класса AllocationItem, в котором описан принцип работы с памятью.
AllocationItem является наследником класса, где описаны метода работы с блоками занятой и активной памяти контейнера.

Листинг программы

Figure.cpp

```
#include "Figure.h"
#include <iostream>
#include <cstdlib>

std::ostream& operator<<(std::ostream& os, Figure& obj){
  obj.Print();
  return os;
}
```

Figure.h

```
#include <iostream>
#ifndef FIGURE_H
#define        FIGURE_H

class Figure {
public:
       virtual double Square() = 0;
       virtual void   Print() = 0;
       friend std::ostream& operator<<(std::ostream& os, Figure& obj);
       virtual ~Figure() {};
};

#endif
```

Quadrate.h

```
#ifndef QUADRATE_H
#define QUADRATE_H
#include <cstdlib>
#include <iostream>
#include "Figure.h"

class Quadrate : public Figure{
public:
        Quadrate();
        Quadrate(std::istream &is);
        Quadrate(size_t i);
        Quadrate(const Quadrate& orig);

        double Square() override;
        void   Print() override;
        friend std::ostream& operator<<(std::ostream& os, const
Quadrate& obj);
        friend std::istream& operator>>(std::istream& is, Quadrate&
obj);

        virtual ~Quadrate();
private:
        size_t side_a;
};

#endif
```

Quadrate.cpp

```
#include "Quadrate.h"
#include <iostream>
#include <cmath>

Quadrate::Quadrate() : Quadrate(0) {
}

Quadrate::Quadrate(size_t i) : side_a(i){
        std::cout << "Quadrate created: " << side_a << std::endl;
}

Quadrate::Quadrate(std::istream &is) {
  int a;
  is >> a;
  if(a>=0){
        side_a=a;
}
else{
  std::cout << "Quadrate not created!" << '\n';
}
}

Quadrate::Quadrate(const Quadrate& orig) {
```

```
	std::cout << "Quadrate copy created" << std::endl;
	side_a = orig.side_a;
}

double Quadrate::Square() {
	return (double)(side_a*side_a);
}

void Quadrate::Print() {
	std::cout << "Quadrate:" << *this << std::endl;
}

Quadrate::~Quadrate() {
	std::cout << "Quadrate deleted" << std::endl;
}

std::ostream& operator<<(std::ostream& os, const Quadrate& obj){
	os << "Size of sides:" << obj.side_a << std::endl;
	return os;
}

std::istream& operator>>(std::istream& is, Quadrate& obj){
	int a;
	is >> a;
	if(a < 0){
		std::cout << "Wrong sizes! Not changed!" << '\n';
	} else{
		obj.side_a=a;
		std::cout << "Quadrate changed!" << '\n';
	}
	return is;
}
```

Rectangle.h

```
#ifndef RECTANGLE_H
#define RECTANGLE_H
#include <cstdlib>
#include <iostream>
#include "Figure.h"

class Rectangle : public Figure{
public:
	Rectangle();
	Rectangle(std::istream &is);
	Rectangle(size_t i,size_t j);
	Rectangle(const Rectangle& orig);

	double Square() override;
	void   Print() override;
	friend std::ostream& operator<<(std::ostream& os, const
Rectangle& obj);
	friend std::istream& operator>>(std::istream& is, Rectangle&
obj);
```

```
	virtual ~Rectangle();
private:
	size_t side_a;
	size_t side_b;
};

#endif
```

Rectangle.cpp

```
#include "Triangle.h"
#include <iostream>
#include <cmath>
#include <stdbool.h>

int max(int a, int b);
int min(int a, int b);

int max(int a, int b){
		return a>b ? a:b;
}
int min(int a, int b){
	return a<b ? a:b;
}


Triangle::Triangle() : Triangle(0, 0, 0) {
}

Triangle::Triangle(size_t ai, size_t bi, size_t ci) {
	if(max(ai, max(bi, ci)) > min(bi, ci)+min(ai, max(bi, ci))){
		std::cout << "Wrong sides! Triangle not created!" <<
'\n';
	}	else if((ai>=0) && (bi>=0) && (ci>=0)){
			a=ai;
			b=bi;
			c=ci;
			std::cout << "Triangle created: " << a << ", " <<
b << ", " << c << std::endl;
	} else{
			std::cout << "Wrong sizes! Triangle not created!"
<< '\n';
		}
}


Triangle::Triangle(std::istream &is) {
	int ai, bi, ci;
	is >> ai;
	is >> bi;
	is >> ci;
	if(max(ai, max(bi, ci)) >= min(bi, ci)+min(ai, max(bi, ci))){
		std::cout << "Wrong sides! Triangle not created!" <<
'\n';
```

```
	}
	else if(ai>=0&&bi>=0&&ci>0){
		a=ai;
		b=bi;
		c=ci;
		std::cout << "Triangle created: " << a << ", " << b << ",
" << c << std::endl;
	}
	else{
		std::cout << "Wrong sizes! Triangle not created!" <<
'\n';
	}
}


Triangle::Triangle(const Triangle& orig) {
	std::cout << "Triangle copy created" << std::endl;
	a = orig.a;
	b = orig.b;
	c = orig.c;
}

double Triangle::Square(){
	double p = double(a + b + c) / 2.0;
	return sqrt(p * (p - double(a))*(p - double(b))*(p -
double(c)));
}

Triangle& Triangle::operator=(const Triangle& right) {

	if (this == &right){
		return *this;
	}
	a = right.a;
	b = right.b;
	c = right.c;

	return *this;
}

Triangle& Triangle::operator++() {

	++a;
	++b;
	++c;

	return *this;
}

Triangle operator+(const Triangle& left,const Triangle& right) {

	return Triangle(left.a+right.a,left.b+right.b,left.c+right.c);
}
```

```
Triangle::~Triangle() {
	std::cout << "Triangle deleted" << std::endl;
}

std::ostream& operator<<(std::ostream& os, const Triangle& obj) {

	os << "a=" << obj.a << ", b=" << obj.b << ", c=" << obj.c <<
std::endl;
	return os;
}

void Triangle::Print(){
	std::cout << "Triangle" << *this;
	return;
}

bool Triangle::operator==(const Triangle& right){
	return (a==right.a && b==right.b && c==right.c);
}

std::istream& operator>>(std::istream& is, Triangle& obj) {
	int ai, bi, ci;
	is >> ai;
	is >> bi;
	is >> ci;
	if(max(ai, max(bi, ci)) >= min(bi, ci)+min(ai, max(bi, ci))){
		std::cout << "Wrong sides! Triangle not changed!" <<
'\n';
	}
	else if(ai>=0&&bi>=0&&ci>0){
		obj.a=ai;
		obj.b=bi;
		obj.c=ci;
		std::cout << "Triangle changed! " << std::endl;
	}
	else{
		std::cout << "Wrong sizes! Triangle not changed!" <<
'\n';
	}

	return is;
}
```

Triangle.h

```
#ifndef TRIANGLE_H
#define	TRIANGLE_H
#include <cstdlib>
#include <iostream>
#include "Figure.h"
#include <stdbool.h>

class Triangle : public Figure{
public:
	Triangle();
```

```
        Triangle(std::istream &is);
        Triangle(size_t ai,size_t bi,size_t ci);
        Triangle(const Triangle& orig);

        Triangle& operator++();
        double Square() override;
        void Print() override;
        friend Triangle operator+(const Triangle& left,const Triangle&
right);
        friend std::ostream& operator<<(std::ostream& os, const
Triangle& obj);
        friend std::istream& operator>>(std::istream& is,  Triangle&
obj);

        Triangle& operator=(const Triangle& right);
        bool operator==(const Triangle& right);

        virtual ~Triangle();
private:
        size_t a;
        size_t b;
        size_t c;
};

#endif
```

Triangle.cpp

```
#include "Triangle.h"
#include <iostream>
#include <cmath>
#include <stdbool.h>

int max(int a, int b);
int min(int a, int b);

int max(int a, int b){
               return a>b ? a:b;
}
int min(int a, int b){
        return a<b ? a:b;
}


Triangle::Triangle() : Triangle(0, 0, 0) {
}

Triangle::Triangle(size_t ai, size_t bi, size_t ci) {
        if(max(ai, max(bi, ci)) > min(bi, ci)+min(ai, max(bi, ci))){
               std::cout << "Wrong sides! Triangle not created!" <<
'\n';
        }      else if((ai>=0) && (bi>=0) && (ci>=0)){
                       a=ai;
                       b=bi;
                       c=ci;
```

```
                        std::cout << "Triangle created: " << a << ", " <<
b << ", " << c << std::endl;
        } else{
                        std::cout << "Wrong sizes! Triangle not created!"
<< '\n';
                }
}


Triangle::Triangle(std::istream &is) {
        int ai, bi, ci;
        is >> ai;
        is >> bi;
        is >> ci;
        if(max(ai, max(bi, ci)) >= min(bi, ci)+min(ai, max(bi, ci))){
                std::cout << "Wrong sides! Triangle not created!" <<
'\n';
        }
        else if(ai>=0&&bi>=0&&ci>0){
                a=ai;
                b=bi;
                c=ci;
                std::cout << "Triangle created: " << a << ", " << b << ",
" << c << std::endl;
        }
        else{
                std::cout << "Wrong sizes! Triangle not created!" <<
'\n';
        }
}


Triangle::Triangle(const Triangle& orig) {
        std::cout << "Triangle copy created" << std::endl;
        a = orig.a;
        b = orig.b;
        c = orig.c;
}

double Triangle::Square(){
        double p = double(a + b + c) / 2.0;
        return sqrt(p * (p - double(a))*(p - double(b))*(p -
double(c)));
}

Triangle& Triangle::operator=(const Triangle& right) {

        if (this == &right){
                return *this;
        }
        a = right.a;
        b = right.b;
        c = right.c;
```

```
	return *this;
}

Triangle& Triangle::operator++() {

	++a;
	++b;
	++c;

	return *this;
}

Triangle operator+(const Triangle& left,const Triangle& right) {

	return Triangle(left.a+right.a,left.b+right.b,left.c+right.c);
}

Triangle::~Triangle() {
	std::cout << "Triangle deleted" << std::endl;
}

std::ostream& operator<<(std::ostream& os, const Triangle& obj) {

	os << "a=" << obj.a << ", b=" << obj.b << ", c=" << obj.c <<
std::endl;
	return os;
}

void Triangle::Print(){
	std::cout << "Triangle" << *this;
	return;
}

bool Triangle::operator==(const Triangle& right){
	return (a==right.a && b==right.b && c==right.c);
}

std::istream& operator>>(std::istream& is, Triangle& obj) {
	int ai, bi, ci;
	is >> ai;
	is >> bi;
	is >> ci;
	if(max(ai, max(bi, ci)) >= min(bi, ci)+min(ai, max(bi, ci))){
		std::cout << "Wrong sides! Triangle not changed!" <<
'\n';
	}
	else if(ai>=0&&bi>=0&&ci>0){
		obj.a=ai;
		obj.b=bi;
		obj.c=ci;
		std::cout << "Triangle changed! " << std::endl;
	}
	else{
```

```
                std::cout << "Wrong sizes! Triangle not changed!" <<
'\n';
        }

        return is;
}
```

Massive.h

```
#ifndef MASSIVE_H
#define MASSIVE_H
#include "Figure.h"
#include "Triangle.h"
#include <memory>

class TrMassive {
public:

  TrMassive();
  TrMassive(unsigned int l);
  TrMassive(const TrMassive& orig);

  bool Empty();
  friend std::ostream& operator<<(std::ostream& os, const TrMassive&
mass);
  std::shared_ptr<Figure>& operator[](const int index);
  int Lenght();
  void Resize(int l);
  ~TrMassive();

private:
  std::shared_ptr<Figure>* data;
  int len;
};

#endif
```

Massive.cpp

```
#include "Massive.h"
#include "Figure.h"
#include "Triangle.h"
#include <iostream>
#include <cstdlib>
#include <memory>

TrMassive::TrMassive() : TrMassive(0){}

TrMassive::TrMassive(unsigned int l) {
  data=nullptr;
  len=l;
  if(len>0){
    data = new std::shared_ptr<Figure>[len];
```

```
  }
  for (short int i=0; i<len; i++){
    data[i]=nullptr;
  }
  std::cout << "massive created!" << std::endl;
}

TrMassive::TrMassive(const TrMassive& orig) {
  len=orig.len;
  data= new std::shared_ptr<Figure>[len];
  for(short int i=0; i<len; i++){
    data[i]=orig.data[i];
  }
  std::cout << "massive copied!" << '\n';
}

bool TrMassive::Empty(){
  return (len==0);
}

std::shared_ptr<Figure>& TrMassive::operator[](const int index){
  if ((index >= len)||(index < 0)){
    std::cout << "Wrong index! Returning element with index 0!" <<
'\n';
    return data[0];
  }
  return data[index];
}

int TrMassive::Lenght(){
  return len;
}

TrMassive::~TrMassive(){
  delete[] data;
  len=0;
  std::cout << "Massive deleted!" << '\n';
}

void TrMassive::Resize(int l){
  if(l<0){
    std::cout << "Wrong size!" << '\n';
    return;
  }
  std::shared_ptr<Figure>* data1;
  if(l==0){
    data1 = nullptr;
  } else{
    data1 = new std::shared_ptr<Figure>[l];
  }
  if (l<len){
    for (short int i = 0; i < l; i++) {
      data1[i]=data[i];
    }} else{
```

```
      short int i;
      for(i=0; i < len; i++){
        data1[i]=data[i];
      }
      while(i<l){
        data1[i]=nullptr;
        ++i;
      }}
  delete[] data;
  len=l;
  data=data1;
  data1=nullptr;
  return;
}

std::ostream& operator<<(std::ostream& os, const TrMassive& mass){
  std::cout << "Massive:" << '\n';
  std::cout << "Size:" << mass.len << '\n' << "Elements:" << std::endl;
  if(mass.len==0){
    std::cout << "Empty!" << '\n';
  }
  for(short int i=0; i<mass.len; i++){
    std::cout << "[" << i << "]" << ":";
    if(mass.data[i]!=nullptr){
      mass.data[i]->Print();
    }else{
      std::cout << "empty" << '\n';
    }
  }
  return os;
}
```

Iterator.cpp

```
#ifndef ITERATOR_H
#define ITERATOR_H
#include <memory>
#include <iostream>

template <class node, class T>
class Iterator
{
public:
  Iterator(node* n){
    node_ptr = n;
    index = 0;
    while(node_ptr[index]==nullptr){
      ++index;
    }
      std::cout << "Iterator on elem with index:" << index << '\n';
  }

  Iterator(node* n, int i){
    node_ptr = n;
    index = i;
```

```
    while(node_ptr[index]==nullptr) {
      ++index;
    }
      std::cout << "Iterator on elem with index:" << index << '\n';
  }
  std::shared_ptr<T> operator *(){
    return node_ptr[index];
  }
  std::shared_ptr<T> operator ->(){
    return node_ptr[index];
  }
  void operator ++() {
    do{
      ++index;
    }while(node_ptr[index]==nullptr);
  }
  Iterator operator ++(int){
    Iterator iter(*this);
    ++(*this);
    return iter;
  }
  bool operator ==(Iterator const& i){
    return (node_ptr == i.node_ptr && i.index == index);
  }
  bool operator !=(Iterator const& i){
    return !(*this == i);
  }
private:
  node* node_ptr;
  int index;
};

#endif
```

main.cpp

```
#include "Massive.h"
#include "Triangle.h"
#include "Figure.h"
#include <memory>
#include <iostream>
#include <cstdlib>
#include "Rectangle.h"
#include "Quadrate.h"
#include "Allocator.h"
//Лабораторная работа №6

int main(){
  short int i, j;
  std::shared_ptr<Figure> abc = nullptr;
  TrMassive<Figure> mass1(10);
  TrMassive<Figure> mass2;
  TrMassive<Figure> mass3(mass1);
  while(1){
```

```
   std::cout <<
"_____________________________________________________________________" <<
'\n';
   std::cout << "Menu\n1-Print massive №1\n2-Print massive №2\n3-Print
massive №3\n4-Enter figure in №1\n5-Enter figure in №2\n6-Enter figure
in №3\n7-Resize №1\n8-Resize №2\n9-Resize №3\n10-Make
Itterations\n0-Exit\nEnter your choise:";
   std::cin >> i;
   std::cout <<
"_____________________________________________________________________" <<
'\n';
   switch (i) {
    case 1:
      std::cout << mass1 << '\n';
      break;
    case 2:
      std::cout << mass2 << '\n';
      break;
    case 3:
      std::cout << mass3 << '\n';
      break;
    case 4:
      std::cout << "Enter index:";
      std::cin >> i;
      if(i<0){
        std::cout << "Wrong index!" << '\n';
        break;
      }
        std::cout << "Enter:\n1-If want to add triangle\n2-If want to
add quadrate\n3-If want to add rectangle" << '\n';
        std::cout << "Your choice:";
        std::cin >> j;
        if(j==1){
          std::cout << "Enter triangle:" << '\n';
          abc.reset(new Triangle(std::cin));
          mass1[i] = abc;
        }
        else if(j==2){
          std::cout << "Enter quadrate:" << '\n';
          abc.reset(new Quadrate(std::cin));
          mass1[i] = abc;
        }
        else if(j==3){
          std::cout << "Enter rectangle:" << '\n';
          abc.reset(new Rectangle(std::cin));
          mass1[i] = abc;
        } else{
            std::cout << "Wrong choice!" << '\n';
         }
      break;
    case 5:
      std::cout << "Enter index:";
      std::cin >> i;
      if(i<0){
```

```
        std::cout << "Wrong index!" << '\n';
        break;
      }
        std::cout << "Enter:\n1-If want to add triangle\n2-If want to
add quadrate\n3-If want to add rectangle" << '\n';
        std::cout << "Your choice:";
        std::cin >> j;
        if(j==1){
          std::cout << "Enter triangle:" << '\n';
          abc.reset(new Triangle(std::cin));
          mass2[i] = abc;
        }
        else if(j==2){
          std::cout << "Enter quadrate:" << '\n';
          abc.reset(new Quadrate(std::cin));
          mass2[i] = abc;
        }
        else if(j==3){
          std::cout << "Enter rectangle:" << '\n';
          abc.reset(new Rectangle(std::cin));
          mass2[i] = abc;
        } else{
            std::cout << "Wrong choice!" << '\n';
         }
      break;
    case 6:
      std::cout << "Enter index:";
      std::cin >> i;
      if(i<0){
        std::cout << "Wrong index!" << '\n';
        break;
      }
        std::cout << "Enter:\n1-If want to add triangle\n2-If want to
add quadrate\n3-If want to add rectangle" << '\n';
        std::cout << "Your choice:";
        std::cin >> j;
        if(j==1){
          std::cout << "Enter triangle:" << '\n';
          abc.reset(new Triangle(std::cin));
          mass3[i] = abc;
        }
        else if(j==2){
          std::cout << "Enter quadrate:" << '\n';
          abc.reset(new Quadrate(std::cin));
          mass3[i] = abc;
        }
        else if(j==3){
          std::cout << "Enter rectangle:" << '\n';
          abc.reset(new Rectangle(std::cin));
          mass3[i] = abc;
        } else{
            std::cout << "Wrong choice!" << '\n';
         }
        break;
```

```
    case 7:
      std::cout << "Enter new size:";
      std::cin >> i;
      mass1.Resize(i);
      std::cout << "New lenght:" << mass1.Lenght() <<'\n';
      break;
    case 8:
      std::cout << "Enter new size:";
      std::cin >> i;
      mass2.Resize(i);
      std::cout << "New lenght:" << mass2.Lenght() <<'\n';
      break;
    case 9:
      std::cout << "Enter new size:";
      std::cin >> i;
      mass3.Resize(i);
      std::cout << "New lenght:" << mass3.Lenght() <<'\n';
      break;
    case 10:
    std::cout << "Enter:\n1-If want to itterate massive №1\n2-If want
to itterate massive №2\n3-If want to itterate massive №3" << '\n';
    std::cout << "Your choice:";
    std::cin >> j;
    if(j==1){
      for(auto it : mass1) std::cout << *it << std::endl;
    }
    else if(j==2){
      for(auto it : mass2) std::cout << *it << std::endl;
    }
    else if(j==3){
      for(auto it : mass3) std::cout << *it << std::endl;

    } else{
        std::cout << "Wrong choice!" << '\n';
     }
      break;
    case 0:
      std::cout << "Корнеев Роман(#1) М80-204" << '\n';
      std::cout <<
"_____________________________________________________________" <<
std::endl;
      return 0;
      break;
}}}
```

MassiveItem.cpp

```
template <class T> MassiveItem<T>::MassiveItem(){
  item = nullptr;
}

template <class T> std::shared_ptr<T>& MassiveItem<T>::GetValue(){
  return item;
```

```
}

template <class T> void MassiveItem<T>::SetValue(std::shared_ptr<T>&
n){
  item = n;
  return;
}

template <class T> MassiveItem<T>& MassiveItem<T>::operator=(const
MassiveItem<T>& right){
  item = right.item;
  return *this;
}

// template <class T>  TAllocationBlock
MassiveItem<T>::allocator(sizeof(MassiveItem<T>),OPT_NUM);
//
// template <class T> void* MassiveItem<T>::operator new[](size_t size)
{
//     void* p = allocator.allocate(size);
//   // int* a;
//   // *a = 5;
//   // p = a;
//   return p;
// }
//
//
// template <class T> void MassiveItem<T>::operator delete[](void *p) {
//   int *a = (int *)p;
//     allocator.deallocate(p, *a + 1);
// }
```

MassiveItem.h

```
#ifndef MASSIVEITEM_H
#define MASSIVEITEM_H

#include "Allocator.h"

template <class T> class MassiveItem {
public:
  MassiveItem();
  std::shared_ptr<T>& GetValue();
  void SetValue(std::shared_ptr<T>& n);
  MassiveItem<T>& operator=(const MassiveItem<T>& right);
  // void* operator new[](size_t size);
  // void operator delete[](void *p);

private:
  std::shared_ptr<T> item;
 //static TAllocationBlock allocator;
};

#include "MassiveItem.cpp"
```

```
#endif
```

Massives.cpp

```
#include "SMassive.h"
#include <iostream>
#include <cstdlib>

 SMassive :: SMassive() :  SMassive(0){}

 SMassive :: SMassive(size_t l){
  sword = 0;
  len=l;
  if(len>0){
    data = (void**)malloc(sizeof(void*)*len);
  }
  std::cout << "Massive of free blocks created!" << std::endl;
}

bool  SMassive::Empty(){
  return (len==0);
}

void* &SMassive::operator[](const size_t index){
  return data[index];
}

void* SMassive::GetBlock(size_t s){
  size_t res = sword;
  if(res + s < len){
    sword = res + s + 1;
    return data[res];
  } else{
    std::cout << "Full Allocator!" << '\n';
    return nullptr;
  }
}

void SMassive::SetBlock(void* pointer, size_t o){
  sword - o;
  data[sword] = pointer;
  return;
}

size_t SMassive::Sword(){
  return sword;
}

size_t SMassive::Lenght(){
  return len;
}

bool SMassive::FreeAloc(size_t s){
  return (sword + s < len);
```

```
}

 SMassive::~ SMassive(){
  if(len>0){
  free(data);
  len=0;
}
  std::cout << "Massive of free blocks deleted!" << '\n';
}

  void  SMassive ::Resize(size_t l){
  if(l<0){
    std::cout << "Wrong size!" << '\n';
    return;
  }
  void** data1;
  if(l==0){
  } else{
    data1 = (void**)malloc(l*sizeof(void*));
  }
  if (l<len){
    for ( size_t i = 0; i < l; i++) {
      data1[i]=data[i];
    }} else{
       size_t i;
      for(i=0; i < len; i++){
        data1[i]=data[i];
      }}
  if(len > 0){
    free(data);
  }
  len=l;
  data=data1;
  data1=nullptr;
  return;
}
```

Massives.h

```
#ifndef SMASSIVE_H
#define SMASSIVE_H
#include <stdbool.h>
#include <cstdlib>

class SMassive{
public:

   SMassive();
   SMassive(size_t l);

  bool Empty();
  bool FreeAloc(size_t s);
  void* &operator[](const size_t index);
  size_t Lenght();
```

```
  void Resize(size_t l);
  void* GetBlock(size_t s);
  void SetBlock(void* pointer, size_t o);
  size_t Sword();

  ~SMassive();

private:
  void **data;
  size_t len;
  size_t sword;

};

#endif
```

Тесты:

Enter:
1|If you want remove from Massive #1
2|If you want remove from Massive #2
3|If you want remove from Massive #3
1
Enter:
1|If you want remove by Value
2|If you want remove by Type
3|If you want remove by Index
2
Enter:
1|If want remove triangles
2|If want remove rectangles
3|If want remove quadreates
3
Quadrate deleted
Massive deleted!

Menu
1-Print massive №1
2-Print massive №2
3-Print massive №3
4-Enter figure in №1
5-Enter figure in №2
6-Print figure in №3
7-Resize №1
8-Resize №2
9-Resize №3
10-Remove Elements from massive

11-Push elements in massive
0-Exit
Enter your choise:1

Massive:
Size:10
Elements:
[0]:empty
[1]: Massive:
Size:5
Elements:
[0]:Triangle: a=5, b=4, c=3
[1]:Rectangle:Size of sides: a=3, b=4
[2]:empty
[3]:empty
[4]:empty

[2]:empty
[3]:empty
[4]:empty
[5]:empty
[6]:empty
[7]:empty
[8]:empty
[9]:empty

Menu
1-Print massive №1
2-Print massive №2
3-Print massive №3
4-Enter figure in №1
5-Enter figure in №2
6-Print figure in №3
7-Resize №1
8-Resize №2
9-Resize №3
10-Remove Elements from massive
11-Push elements in massive
0-Exit
Enter your choise:4

Enter index of conteiner #1:3
Enter:
1-If want to add triangle
2-If want to add quadrate

3-If want to add rectangle
Your choice:2
Enter quadrate:
6
massive 2nd container created!

Menu
1-Print massive №1
2-Print massive №2
3-Print massive №3
4-Enter figure in №1
5-Enter figure in №2
6-Print figure in №3
7-Resize №1
8-Resize №2
9-Resize №3
10-Remove Elements from massive
11-Push elements in massive
0-Exit
Enter your choise:4

Enter index of conteiner #1:3
Enter:
1-If want to add triangle
2-If want to add quadrate
3-If want to add rectangle
Your choice:1
Enter triangle:
3 4 5
Triangle created: 3, 4, 5

Menu
1-Print massive №1
2-Print massive №2
3-Print massive №3
4-Enter figure in №1
5-Enter figure in №2
6-Print figure in №3
7-Resize №1
8-Resize №2
9-Resize №3
10-Remove Elements from massive
11-Push elements in massive
0-Exit
Enter your choise:4

Enter index of conteiner #1:4
Enter:
1-If want to add triangle
2-If want to add quadrate
3-If want to add rectangle
Your choice:2
Enter quadrate:
7
massive 2nd container created!

Menu
1-Print massive №1
2-Print massive №2
3-Print massive №3
4-Enter figure in №1
5-Enter figure in №2
6-Print figure in №3
7-Resize №1
8-Resize №2
9-Resize №3
10-Remove Elements from massive
11-Push elements in massive
0-Exit
Enter your choise:4

Enter index of conteiner #1:3
Enter:
1-If want to add triangle
2-If want to add quadrate
3-If want to add rectangle
Your choice:1
Enter triangle:
7 7 7
Triangle created: 7, 7, 7

Menu
1-Print massive №1
2-Print massive №2
3-Print massive №3
4-Enter figure in №1
5-Enter figure in №2
6-Print figure in №3
7-Resize №1
8-Resize №2
9-Resize №3
10-Remove Elements from massive

11-Push elements in massive
0-Exit
Enter your choise:1

Massive:
Size:10
Elements:
[0]:empty
[1]: Massive:
Size:5
Elements:
[0]:Triangle: a=5, b=4, c=3
[1]:Rectangle:Size of sides: a=3, b=4
[2]:empty
[3]:empty
[4]:empty

[2]:empty
[3]: Massive:
Size:5
Elements:
[0]:Triangle: a=3, b=4, c=5
[1]:Triangle: a=7, b=7,

c=7
[2]:Quadrate:Size of sides:6
[3]:empty
[4]:empty

[4]: Massive:
Size:5
Elements:
[0]:Quadrate:Size of sides:7
[1]:empty
[2]:empty
[3]:empty
[4]:empty

[5]:empty
[6]:empty
[7]:empty
[8]:empty
[9]:empty

Menu

1-Print massive №1
2-Print massive №2
3-Print massive №3
4-Enter figure in №1
5-Enter figure in №2
6-Print figure in №3
7-Resize №1
8-Resize №2
9-Resize №3
10-Remove Elements from massive
11-Push elements in massive
0-Exit
Enter your choise:10

Enter:
1|If you want remove from Massive #1
2|If you want remove from Massive #2
3|If you want remove from Massive #3
1
Enter:
1|If you want remove by Value
2|If you want remove by Type
3|If you want remove by Index
1
Enter value of square: 20
Triangle deleted
Rectangle deleted
Massive deleted!
Triangle deleted

Menu
1-Print massive №1
2-Print massive №2
3-Print massive №3
4-Enter figure in №1
5-Enter figure in №2
6-Print figure in №3
7-Resize №1
8-Resize №2
9-Resize №3
10-Remove Elements from massive
11-Push elements in massive
0-Exit
Enter your choise:1

Massive:

Size:10
Elements:
[0]:empty
[1]:empty
[2]:empty
[3]: Massive:
Size:5
Elements:
[0]:empty
[1]:Triangle: a=7, b=7, c=7
[2]:Quadrate:Size of sides:6
[3]:empty
[4]:empty

[4]: Massive:
Size:5
Elements:
[0]:Quadrate:Size of sides:7
[1]:empty
[2]:empty
[3]:empty
[4]:empty

[5]:empty
[6]:empty
[7]:empty
[8]:empty
[9]:empty

Menu
1-Print massive №1
2-Print massive №2
3-Print massive №3
4-Enter figure in №1
5-Enter figure in №2
6-Print figure in №3
7-Resize №1
8-Resize №2
9-Resize №3
10-Remove Elements from massive
11-Push elements in massive
0-Exit
Enter your choise:0

Made by Bronnikov Max(#1) M80-204

```
Massive deleted!
Massive deleted!
Quadrate deleted
Massive deleted!
Quadrate deleted
Triangle deleted
Massive deleted!
Massive deleted!
```

4.Вывод

Написана программа на языке C++, которая позволяет работать с фигурами трёх видов, создаёт контейнер первого уровня и второго, позволяет заполнять одну ячейку пятью фигурами и производить различные операции по их занесению и типу удаления и массива. Работа довольно интересна так как позволяет познакомится с работой контейнеров в более подробном виде и учет работе с памятью в нейком роде.

5.Список литературы

1.http://cppstudio.com/post/8482/ [электронный ресурс]
2.https://prog-cpp.ru/data-dls/ [электронный ресурс]
3.https://ru.cppreference.com/w/cpp/container/ [электронный ресурс]
4.https://metanit.com/cpp/tutorial/7.3.php [электронный ресурс]