Абстрактные типы данных

Стек. Очередь. Список.

Введение

- Почему рассматриваемые типы называются «абстрактными»?
- Примеры реализации АТД на основе массивов и структур.

Типы данных

- Базовые
- Составные
- АТД

Словарь

- Полиморфизм
- Шаблон функции
- Инкапсуляция
- Класс
- Шаблон класса

Полиморфизм

- возможность описания нескольких функций с различным набором параметров, имеющих одно и тоже имя
- int abs(int a) { return a>=0?a:-a; }
- double abs(double a) { return a>=0?a:-a; }

Шаблоны функций

- template<typename T> T abs(T a) {
  return a>=0?a:-a; }
- int x; … y = abs(x); // int abs(int);
- float u; … v = abs(u); // float abs(float);

Инкапсуляция

```
struct complex { float Re,Im;
complex() { Re = Im = 0; }
complex(float R, float I) { Re=R; Im=I; }
float abs() { return sqrt(Re*Re + Im*Im); }
complex operator+(complex a) {
    return complex(Re+a.Re, Im+a.Im); }
friend ostream& operator<<(ostream& a, const
   complex& b) {
    return a << '(' << b.Re << ',' << b.Im << ')' << endl; }
};
```

Класс

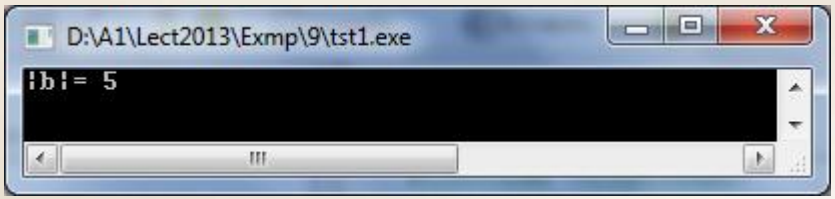
```
class complex { float Re,Im;
public:
complex() { Re = Im = 0; }
complex(float R, float I) { Re=R; Im=I; }
float abs() { return sqrt(Re*Re + Im*Im); }
complex operator+(complex a) {
    return complex(Re+a.Re, Im+a.Im); }
friend ostream& operator<<(ostream& a, const
   complex& b) {
    return a << '(' << b.Re << ',' << b.Im << ')' << endl; }
};
```

Шаблон класса

```
template<typename T> class complex {
   T Re,Im;
public:
   complex(T R=0, T I=0) { Re=R; Im=I; }
   T abs() { return sqrt(Re*Re + Im*Im); }
   complex operator+(complex a) {
      return complex(Re+a.Re, Im+a.Im); }
friend ostream& operator<<(ostream& a, const
   complex& b) {
     return a << '(' << b.Re << ',' << b.Im << ')';}
};
```

Использование шаблона класса

```
#include <iostream>
using namespace std;
#include "complex.h"
int main() { complex<double> a(1.,2.),b,c(2.,2.);
   b = a + c;
   cout << "|b|= " << b.abs() << endl;
return 0; }
```



СПИСОК, ОЧЕРЕДЬ, СТЕК

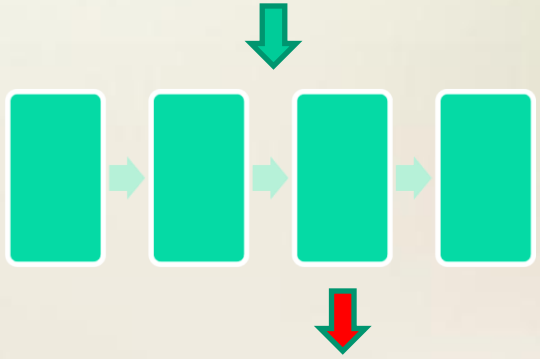
- Очередь

FIFO – First In First Out

- Стек

LIFO – Last In First Out

- Список



Casual In Casual Out

РАЗРАБОТКА АТД НА ОСНОВЕ ИСПОЛЬЗОВАНИЯ МАССИВОВ

ФУНКЦИЯ ДЛЯ ВЫВОДА СООБЩЕНИЙ ОБ ОШИБКАХ

```
#include <iostream>
using namespace std;
inline void ERR(const char* s) {
        cerr << s << endl;  exit(1); }
```

Шаблон класса Stack

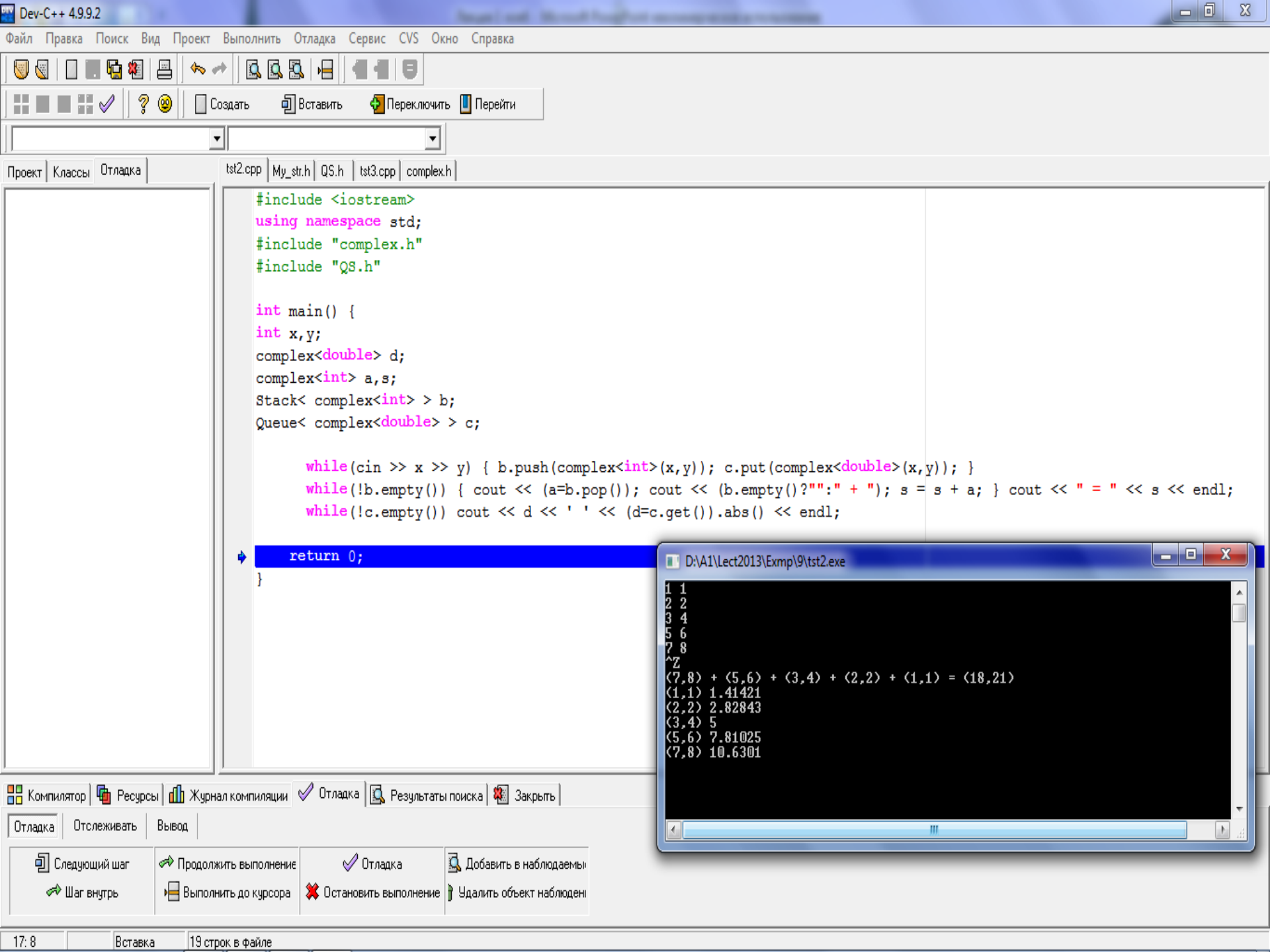
```
template <typename T, int SIZE=100> class Stack {
    T *elems; int numElems;
public:
    Stack() { elems = new T[SIZE]; numElems = 0; }
    void push(T const& elem) {
        if(numElems==SIZE) ERR("Stack::push: stack is full");
        elems[numElems++]=elem; }
    T& pop() {
        if(numElems<1) ERR("Stack::pop: stack is empty");
        return elems[--numElems]; }
    bool empty() { return numElems == 0; }
    bool full() { return numElems==SIZE; }
    ~Stack() { delete[] elems; }                    };
```

Шаблон класса Queue

```
template <typename T, int SIZE> class Queue {
    T *elems; int numElems, IP;
public:
    Queue() { elems = new T[SIZE]; IP=SIZE/2; numElems=0; }
    bool full() { return numElems==SIZE; }
    bool empty() { return numElems==0;}
    void put(T const& elem) {
        if(numElems==SIZE) ERR("Queue::put: queue is full");
        elems[IP++] = elem; numElems++; if(IP==SIZE) IP=0; }
    T& get() {
        if(numElems<1) ERR("Queue::get: queue is empty");
        return elems[IP + (IP<numElems?SIZE:0) - numElems--];}
    ~Queue() { delete[] elems; }                                    };
```

C:\Windows\System32\cmd.exe

```
D:\A1\Lect2013\Exmp\9>tst3 <tst3.cpp
}
    return 0;

      while(!c.empty()) cout << c.get();
      while(!b.empty()) cout << b.pop(); cout << endl;
      while(cin >> a) { b.push(a); c.put(a); }

Queue<My_string> c;
Stack<My_string> b;
My_string a;
int main() {

#include "QS.h"
#include "My_str.h"
using namespace std;
#include <iostream>

#include <iostream>
using namespace std;
#include "My_str.h"
#include "QS.h"

int main() {
My_string a;
Stack<My_string> b;
Queue<My_string> c;

      while(cin >> a) { b.push(a); c.put(a); }
      while(!b.empty()) cout << b.pop(); cout << endl;
      while(!c.empty()) cout << c.get();

    return 0;
}
```


Dev-C++ 4.9.9.2
Файл Правка Поиск Вид Проект Выполнить Отладка Сервис CVS Окно Справка
Создать Вставить Переключить Перейти
Проект Классы Отладка
tst2.cpp My_str.h QS.h tst3.cpp complex.h
#include <iostream>
using namespace std;
#include "complex.h"
#include "QS.h"
int main() {
int x,y;
complex<double> d;
complex<int> a,s;
Stack< complex<int> > b;
Queue< complex<double> > c;
while(cin >> x >> y) { b.push(complex<int>(x,y)); c.put(complex<double>(x,y)); }
while(!b.empty()) { cout << (a=b.pop()); cout << (b.empty()?"":" + "); s = s + a; } cout << " = " << s << endl;
while(!c.empty()) cout << d << ' ' << (d=c.get()).abs() << endl;
return 0;
}
D:\A1\Lect2013\Exmp\9\tst2.exe
1 1
2 2
3 4
5 6
7 8
^Z
(7,8) + (5,6) + (3,4) + (2,2) + (1,1) = (18,21)
(1,1) 1.41421
(2,2) 2.82843
(3,4) 5
(5,6) 7.81025
(7,8) 10.6301
Компилятор Ресурсы Журнал компиляции Отладка Результаты поиска Закрыть
Отладка Отслеживать Вывод
Следующий шаг
Шаг внутрь
Продолжить выполнение
Выполнить до курсора
Отладка
Остановить выполнение
Добавить в наблюдаемы
Удалить объект наблюдени
17: 8
Вставка
19 строк в файле

РАЗРАБОТКА АТД НА ОСНОВЕ ИСПОЛЬЗОВАНИЯ СТРУКТУР

Шаблон узла набора данных

```
template <typename T> class Item {
   T node; Item* next;
public:
   Item(const T &elem, Item* n=0) {node=elem; next=n;}
   T& get_node() { return node; }
   Item* &get_next() { return next; }                    };
```

Шаблон класса Stack

```
template <typename T> class Stack {
        Item<T> *top; T rab;
public:
Stack() { top = 0; }
void push(const T& elem) { top=new Item<T>(elem,top); }
T& pop() { if(!top) ERR("Stack::pop: empty stack");
        rab = top->get_node(); Item<T> *p=top;
        top = p->get_next(); delete p; return rab; }
bool empty() {return top == 0; }
};
```

Шаблон класса Queue

```
template <typename T> class Queue {
        Item<T> *head,*tail; T rab;
public:
Queue() { tail = head =0; }
bool empty() { return head == 0;}
void put(const T& elem) {
        if(tail==0) tail = head = new Item<T>(elem);
        else tail = (tail->get_next() = new Item<T>(elem)); }
T& get() { if(!head) ERR("Queue::get: queue is empty");
        Item<T> *p=head; rab = head->get_node();
        head=head->get_next(); delete p; if(head==0) tail=0;
        return rab; }                    };
```

Шаблон класса List

```
template <typename T> class List {
  Item<T> *front,*back; T rab;
  Item<T>* find(Item<T>* &F, const T& k) {
      if(front==NULL) return (F=NULL);
      Item<T> *ptr=F=front;
      if(front->get_node()==k) return 0;
      while((F=ptr->get_next())!=NULL) {
            if(F->get_node()==k) break; ptr=F; }
      return ptr; }
public:
```

```
List() { front = back =0; }
bool empty() { return front==0; }
void push_back(const T& elem) {
        if(back==0) front = back = new Item<T>(elem);
        else back = (back->get_next() = new Item<T>(elem));  }
T& pop_back() {
        if(back==0) ERR("List::pop_back: list is empty");
        rab=back->get_node(); Item<T> *p=front;
        if(front==back) front = back = 0;
        else { while(p->get_next()!=back) p=p->get_next();
        back=p; p=p->get_next(); back->get_next()=0;}
        delete p; return rab; }
bool insert_after(const T& k, const T& after) {
        Item<T> *c; find(c,after); if(c==0) return 0;
        c->get_next()=new Item<T>(k,c->get_next()); return 1; }
```

```
bool remove(const T& k) { Item<T> *b,*c; b=find(c,k);
   if(c==NULL) return 0;
   if(b==NULL) { front=front->get_next(); delete (c); }
   else { b->get_next()=c->get_next(); delete(c); }
   return 1;}
void push_front(const T& elem) {
   front = new Item<T>(elem,front);
   if(back==0) back = front; }
T& pop_front() {
   if(front==0) ERR("List::pop_front: list is empty");
   Item<T> *p=front; rab = p->get_node();
   front=p->get_next(); delete p;
   if(front==0) back=0; return rab; }
```

```
void sort() { Item<T> *D=0;
   while(front!=0) {
      Item<T> *p=front; rab = front->get_node();
      while(p=p->get_next())if(p->get_node()>rab)
                              rab=p->get_node();
      D = new Item<T>(rab,D); remove(rab); }
   front = D; back = front;
   while(back->get_next()!=0) back=back->get_next(); }
void revers() { Item<T> *D=0;
   while(front!=0) D = new Item<T>(pop_front(),D);
   front = D; back=front;
   while(back->get_next()!=0) back=back->get_next(); }
```

```
T& operator[](int i) {
   Item<T>* p=front;
   while(i-- && p) p=p->get_next();
   if(p) return p->get_node();
   ERR("LIST::operator[]: end of List appear");}
friend ostream& operator <<
                (ostream& out, const List<T>& a) {
   Item<T>* p=a.front;
   while(p) { out << p->get_node() << ' ';
              p=p->get_next(); }
   return out; }                              };
```

```
#include <iostream>
using namespace std;
#include "My_string"
#include “SQL.h"

int main() {
My_string x;
Stack<My_string> a;
Queue<My_string> b;
List<My_string> c;

while(cin >> x) { a.push(x); b.put(x); c.push_back(x); }
while(!b.empty()) cout << b.get() << endl; cout << "\n";
while(!a.empty()) cout << a.pop() << endl;
```

```
for(int i=3; i>0; i--) { x=c[i]; c.remove(x); }
c.revers(); cout << c;  cout << endl;
c.sort();   cout << c; cout << endl;
c.push_front(x); c.insert_after(x,x);
cout << c;  cout << endl;

    return 0;
}
```

Пример 2: вычисление площади прямоугольной фигуры, заданной произвольным перечислением координат вершин

```
#include <iostream>
#include <fstream>
#include “SQL.h"      /* Файл с описанием
   шаблонов Стека, Очереди и Списка  */
using namespace std;
```

```
class point {
      int x,y;
public:
      point(int a=0,int b=0) { x=a; y=b; }
      friend ostream& operator<< ( ostream& a, const point& b) {
            return a << '(' << b.x << ',' << b.y << ')'; }
      friend istream& operator>> ( istream& a, point& b) {
            return a >> b.x >> b.y; }
      bool operator> (const point& b) {
          if (x==b.x) return y>b.y; else return x>b.x; }
      bool operator!= (const point& b) { return x!=b.x || y!=b.y; }
      bool operator== (const point& b) { return x==b.x && y==b.y; }
      point operator! () { return point(y,x); }
      int operator[](int i) { return i==1?x:y; }
};
```

```
int main() {
List<point> Sx,Sy;
point a,b; int i,k,s=0;
ifstream in("test_sq.dat");
    while(in>>a) { Sx.push_front(a); Sy.push_front(!a); }
    Sx.sort(); Sy.sort(); a=b=Sx[0];
    do {    for(i=k=0;Sx[i][1]<a[1];i++);
            while(a[1]==Sx[i][1] && Sx[i][2]<a[2]) i++,k++;
            if(k%2) i--; else i++; a=!Sx[i]; s-=a[1]*a[2];
            for(i=k=0;Sy[i][1]<a[1];i++);
            while(a[1]==Sy[i][1] && Sy[i][2]<a[2]) i++,k++;
            if(k%2) i--; else i++; a=!Sy[i]; s+=a[1]*a[2];
    } while(a!=b);      cout << "S= " << s << endl;
    return 0; }
```