* 1. **Конструкторы**

В предыдущем примере метод chislo.vvod() использовался для присвоения начального значения некоторым членам класса, однако для упрощения процесса инициализации объекта предусмотрена специальная функция, которая называется конструктором. Имя конструктора совпадает с именем класса. **Конструктор** предназначен для инициализации объекта и запускается автоматически при создании экземпляра класса (при объявлении переменной типа class).

. Ниже перечислены основные свойства конструкторов.

* Конструктор *не возвращает значение*, даже типа **void**.
* Класс может иметь *несколько конструкторов* с разными параметрами для разных видов инициализации (при этом используется механизм перегрузки).
* Конструктор, вызываемый без параметров, называется *конструктором по умолчанию.*
* *Параметры конструктора* могут иметь любой тип, кроме этого же класса. Можно задавать значения параметров по умолчанию. Их может содержать только один из конструкторов.
* Если программист не указал ни одного конструктора, компилятор создает его *автоматически.* Такой конструктор вызывает конструкторы по умолчанию для полей класса. В случае, когда класс содержит константы или ссылки, при попытке создания объекта класса будет выдана ошибка, поскольку их необходимо инициализировать конкретными значениями, а конструктор по умолчанию этого делать не умеет.
* *Конструкторы не наследуются*.
* Конструкторы нельзя описывать с модификаторами **const, virtual** и **static**.
* Конструкторы глобальных объектов вызываются до вызова функции **main**. Локальные объекты создаются, как только становится активной область их действия. Конструктор запускается и при создании временного объекта (например, при передаче объекта из функции).
* Конструктор вызывается, если в программе встретилась какая-либо из синтаксических конструкций:

**имя\_класса имя\_объекта [(список параметров)];**

// Список параметров не должен быть пустым

**имя\_класса (список параметров);**

// Создается объект без имени (список может быть пустым)

**имя\_класса имя\_объекта = выражение;**

// Создается объект без имени и копируется

Примеры:

complex S (200, 300), V(50), Z;

complex X = complex (1000);

complex Y = 500;

В первом операторе создаются три объекта. Значения не указанных параметров устанавливаются по умолчанию.

Во втором операторе создается безымянный объект со значением параметра **x = 1000** (значение второго параметра устанавливается по умолчанию). Выделяется память под объект **Х**, в которую копируется безымянный объект.

В последнем операторе создается безымянный объект со значением параметра **x = 500** (значение второго параметра устанавливается по умолчанию). Выделяется память под объект **Y**, в которую копируется безымянный объект. Такая форма создания объекта возможна в том случае, если для инициализации объекта допускается задать один параметр.

ЗАДАЧА 3.2. Добавим в созданный в предыдущем примере класс complex конструктор:

#include <iostream>

#include <string>

#include <math.h>

const double PI=3.14159;

using namespace std;

//Объявляем класс complex. Внутри класса указаны, только прототипы методов,

//а сами функции описаны за пределами класса.

class complex

{

double x;

double y;

void show\_complex();

public:

//Прототип конструктора класса.

complex();

double modul();

};

complex::complex()

{

cout<<"Vvedite x\t"; cin>>x;

cout<<"Vvedite y\t"; cin>>y;

}

double complex::modul()

{

return pow(x\*x+y\*y,0.5);

}

void complex::show\_complex()

{

if (y>=0) cout<<x<<"+"<<y<<"i"<<endl;

else cout<<x<<y<<"i"<<endl;

}

int main()

{

//Описываем экземпляр класса, при выполнении программы после создания переменной

//автоматически вызывается констуктор.

complex chislo;

chislo.show\_complex();

cout<<"Modul kompleksnogo chisla="<<chislo.modul();

return 1;

}

Если члены класса, являются массивами (указателями), то в конструкторе логично предусмотреть выделение памяти.

ЗАДАЧА 3.3. С использованием классов решить следующую задачу. Заданы координаты n точек в k-мерном пространстве. Найти точки, расстояние между которыми наибольшее и наименьшее.

Для решения задачи создадим класс prostr.

Члены класса:

• int n – количество точек;

• int k – размерность пространства;

• double \*\*a – матрица, в которой будут храниться координаты точек, a[i][j] - i-я координата точки с номером j.

• double min – минимальное расстояние между точками в k-мерном пространстве;

• double max – максимальное расстояние между точками в k-мерном пространстве;

• int imin, int jmin – точки, расстояние между которыми минимально;

• int imax, int jmax – точки, расстояние между которыми максимально.

Методы класса:

• prostr() – конструктор класса, в котором определяются n – количество точек, k – размерность пространства, выделяется память для матрицы a координат точки и вводятся координаты точек;

• poisk\_max() – функция нахождения точек, расстояние между которыми наибольшее;

• poisk\_min() – функция нахождения точек, расстояние между которыми наименьшее;

• vivod\_result() – функция вывода результатов: значений min, max, imin, jmin, imax, jmax;

• delete\_a() – освобождение памяти выделенной для матрицы a.

В главной программе необходимо будет описать экземпляр класса и

последовательно вызвать методы poisk\_min(), poisk\_max(), vivod\_result(), delete\_a().

Текст программы:

#include <iostream>

#include <math.h>

using namespace std;

class prostr{

//Все члены класса – закрытые.

int n;

int k;

double \*\*a;

double min;

double max;

int imin;

int jmin;

int imax;

int jmax;

//Открытые методы класса.

public:

prostr();

void poisk\_min();

void poisk\_max();

int vivod\_result();

int delete\_a();

};

prostr::prostr()

{

int i,j;

cout<<"VVedite razmernost prostrantva ";

cin>>k;

cout<<"VVedite kolichestvo tochek ";

cin>>n;

a=new double\*[k];

for(i=0;i<k;i++)

a[i]=new double[n];

for(j=0;j<n;j++)

{

cout<<"VVedite koordinati "<<j<<" tochki"<<endl;

for(i=0;i<k;i++)

cin>>a[i][j];

}

}

void prostr::poisk\_max()

{

int i,j,l;

double s;

for(max=0,l=0;l<k;l++)

max+=(a[l][0]-a[l][1])\*(a[l][0]-a[l][1]);

max=pow(max,0.5);

imax=0;jmax=1;

for(i=0;i<n;i++)

for(j=i+1;j<n;j++)

{

for(s=0,l=0;l<k;l++)

s+=(a[l][i]-a[l][j])\*(a[l][i]-a[l][j]);

s=pow(s,0.5);

if (s>max)

{

max=s;

imax=i;

jmax=j;

}

}

}

void prostr::poisk\_min()

{

int i,j,l;

double s;

for(min=0,l=0;l<k;l++)

min+=(a[l][0]-a[l][1])\*(a[l][0]-a[l][1]);

min=pow(min,0.5);

imin=0;jmin=1;

for(i=0;i<k;i++)

for(j=i+1;j<n;j++)

{

for(s=0,l=0;l<k;l++)

s+=(a[l][i]-a[l][j])\*(a[l][i]-a[l][j]);

s=pow(s,0.5);

if (s<min)

{

min=s;

imin=i;

jmin=j;

}

}

}

void prostr::vivod\_result()

{

int i,j;

for(i=0;i<k;cout<<endl,i++)

for (j=0;j<n;j++)

cout<<a[i][j]<<"\t";

cout<<"max="<<max<<"\t nomera "<<imax<<"\t"<<jmax<<endl;

cout<<"min="<<min<<"\t nomera "<<imin<<"\t"<<jmin<<endl;

}

void prostr::delete\_a()

{

delete [] a;

}

void main()

{

prostr x; //Описание переменной - экземпляра класса prostr.

//Вызов метода poisk\_max для поиска максимального расстояния между

//точками в k-мерном пространстве;

x.poisk\_max();

//Вызов метода poisk\_min для поиска минимального расстояния между

//точками в k-мерном пространстве;

x.poisk\_min();

//Вызов метода vivod\_result для вывода результатов

x.vivod\_result();

//Вызов функции delete\_a.

x.delete\_a();

}

Результаты работы программы:

VVedite razmernost prostrantva 2

VVedite kolichestvo tochek 4

VVedite koordinati 0 tochki

1 2

VVedite koordinati 1 tochki

2 3

VVedite koordinati 2 tochki

4 5

VVedite koordinati 3 tochki

-7 -9

1 2 4 -7

2 3 5 -9

max=17.8045 nomera 2 3

min=1.41421 nomera 0 1

* 1. **Перегрузка конструктора**

Также как и любые другие функции, конструкторы могут перегружаться. Перепишем предыдущий пример, добавив в него перегружаемый конструктор, в который можно передавать значения n и k. В этом примере экземпляр класса можно описывать, например, так prostr x; в этом случае конструктор вызывается без параметров, или так prostr x (3,5); в этом случае вызывается перегружаемый конструктор с параметрами.

#include <iostream>

#include <math.h>

using namespace std;

class prostr{

int n;

int k;

double \*\*a;

double min;

double max;

int imin;

int jmin;

int imax;

int jmax;

public:

prostr(int,int);

prostr();

void poisk\_min();

void poisk\_max();

void vivod\_result();

void delete\_a();

};

//Текст второго конструктора

//Нельзя в качестве формальных параметров конструктора использовать переменные n и k,

//потому что это имена членов класса. Если в качестве формальных параметров указать n и k,

//то внутри конструктора будут использоваться локальные переменные n и k, но при этом члены //класса prost n и k будут не определены.

prostr::prostr(int k1, int n1)

//Входными параметрами являются размерность пространства n1 и количество точек в //пространстве k1.

{

int i,j;

//Присваиваем членам класса n и k значения входных параметров конструктора

k=k1;

n=n1;

a=new double\*[k];

for(i=0;i<k;i++)

a[i]=new double[n];

for(j=0;j<n;j++)

{

cout<<"VVedite koordinati "<<j<<"tochki"<<endl;

for(i=0;i<k;i++)

cin>>a[i][j];

}}

void main()

{

//Можно вызывать конструктор с параметрами

prostr x(3,5);

//или без prostr x; в этом случае будет вызываться

//тот же конструктор, что и в предыдущем примере.

x.poisk\_max();

x.poisk\_min();

x.vivod\_result();

x.delete\_a();

}