Лабораторная работа № 2
Наследование классов

Рассмотрим пример создания и использования иерархии классов с использованием механизма наследования.

```
//Программа “база данных по учету студентов”, использующая механизм
//наследования
#include <iostream.h>
#include <string.h>
class Subject  //класс, описывающий свойства некоторого субъекта  {
protected:    char name[20];   //имя субъекта
   int age;              //возраст
   char adress[30];  //адрес
  public:
   void Read();       //функция ввода информации о субъекте с клавиатуры
   void Write();      //функция вывода информации о субъекте на экран
};
class Student:public Subject  //класс, описывающий свойства студента
{ char  group[7];   //название группы, в которой учится студент
  char  numb[8];    //номер его зачетной книжки
  int   balls[10];    //оценки, полученные на экзамене
  static int n;        //количество экзаменов в сессию
  protected:
```

```
 float rait;          //рейтинг студента (среднее по баллам, полученным на экзаменах)
 public:
  void Exam();      //функция ввода баллов по предметам
  void CalcRait();  //функция вычисления рейтинга студента
  void ReadSt();    // функция ввода информации о студенте с клавиатуры
  void WriteSt();   // функция вывода информации о студенте на экран
};
        class DayStud:public Student   //класс, описывающий свойства студента дневной
  // формы обучения
{int stip;              //стипендия студента
 public:
 void CalcStip();     //функция вычисления стипендии студента
 void WriteSt();      //переопределенная функция вывода информации о студенте
};

// Определение методов классов
void Subject::Read()
{ cout<<" Введите информацию\n  Имя";
  cin>>name;
  cout<<"\n Возраст";
  cin>> age;
  cout<<"\nАдрес";
```

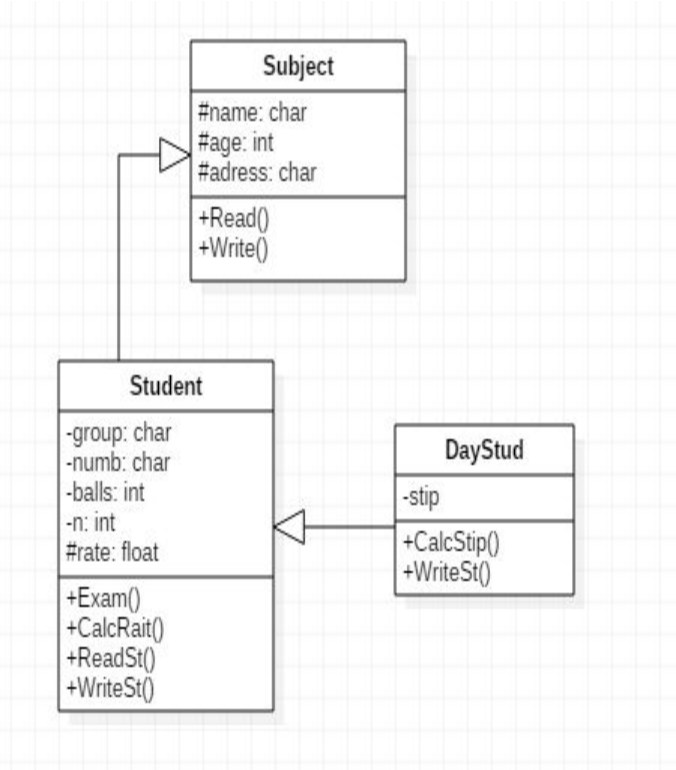
```
void Subject::Write() { cout<<" Имя  "<<name<<"  Возраст  "<<age<<"  Адрес
"<<adress;} int Student::n=4; void Student::ReadSt() { Read();   cout<<"\nНомер зач.
книжки";   cin>>numb;
  cout<<"\nГруппа";
  cin>>group;
}
void Student::WriteSt() { Write();   cout<<"Номер зач.книжки  "<<numb<<"Группа
"<<group<<" Рейтинг "<<rait<<"\n";
}
void Student::CalcRait() { rait=0;   for(int i=0;i<n;i++)    rait+=balls[i];    rait/=n; } void
Student::Exam()
{ for (int j=0;j<n;j++)
   { cout<<"\nПредмет  N"<<j+1;
     cin>>balls[j];
 }}
void DayStud::CalcStip()
{if (rait>=90)   stip=300;
 else  if (rait>=76)
   stip=200;
  else   stip=0;
}
void DayStud::WriteSt()
{ Student::WriteSt();
  cout<<"Стипендия"<<stip;
```

```
//пример использования определенных выше классов
main()
{ const int m=10;    //будем работать с 10-ю студентами
  int i;
  DayStud gr[m];
  for(i=0;i<m;i++)
   gr[i].ReadSt();    //вводим информацию о каждом студенте
  for(i=0;i<m;i++)
   { cout<<"Экзамены"<<i+1<<" студента";
     gr[i].Exam();    //проводим экзамены (вводим информацию о баллах,
полученных
      //каждым студентом на экзаменах
   }
  for(i=0;i<m;i++)
   gr[i].CalcRait();   //вычисляем рейтинг каждого студента
  for(i=0;i<m;i++)
   gr[i].CalcStip();   //вычисляем стипендию каждого из студентов
  for(i=0;i<m;i++)
   gr[i].WriteSt();    //выводим информацию о каждом студенте на экран
}
```

В приведенном выше примере определены три класса. Класс Subject является корнем всей системы классов, в нем объединены свойства и методы, описывающие «субъекта», то есть свойства, присущие каждому человеку: имя, возраст, адрес, и методы, позволяющие обрабатывать эту информацию (вводить с клавиатуры, выводить на экран).

От класса Subject порожден класс Student, в котором определены свойства, присущие каждому студенту: номер зачетки, название группы, в которой учится студент, оценки, полученные им на экзаменах, рейтинг студента, вычисленный по результатам сессии. В классе также определен ряд методов, позволяющих изменять перечисленные свойства: вводить с клавиатуры, рассчитывать, выводить на экран. При этом, некоторые методы класса Student вызывают методы, унаследованные от родительского класса Subject. Так, например, для ввода информации о студенте в классе определена функция ReadSt, в которой непосредственно вводятся с клавиатуры лишь те компонентные данные, которые определены в классе Student. Для ввода значения компонент, унаследованных от Subject (очевидно, что для каждого студента необходимо хранить имя, возраст, адрес) вызывается унаследованный метод Read.

Третий класс называется DayStud является конкретизацией класса Student в плане описания свойств студента дневного отделения. В частности, для студента дневного отделения определено компонентное данное stip (стипендия), значение которого вычисляется в компонентной

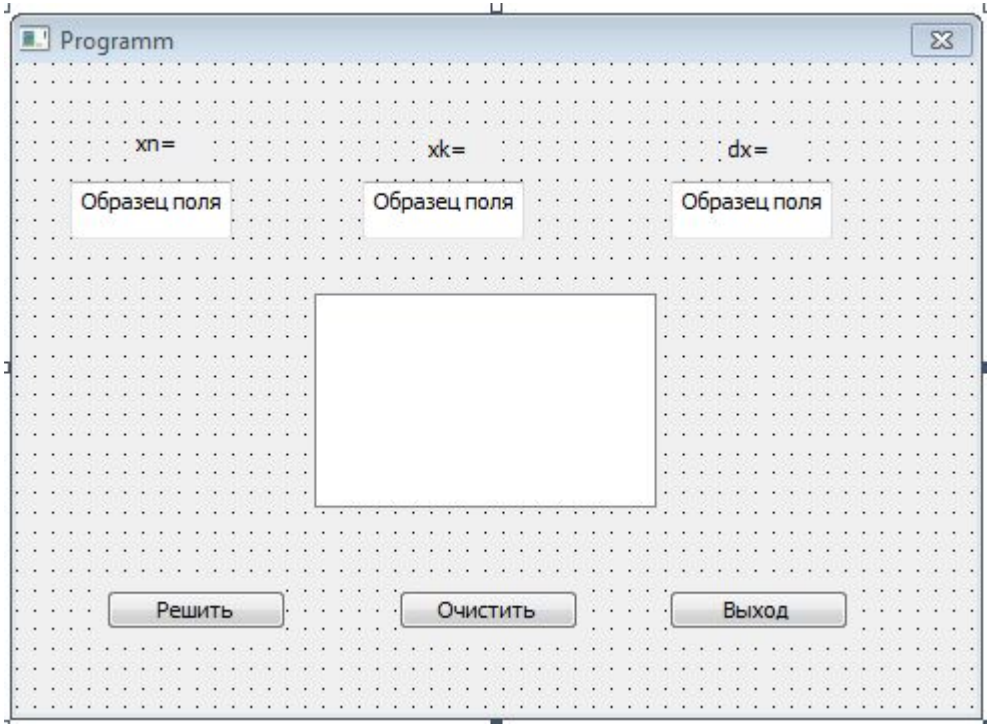

Subject
#name: char
#age: int
#adress: char
+Read()
+Write()
Student
-group: char
-numb: char
-balls: int
-n: int
#rate: float
+Exam()
+CalcRait()
+ReadSt()
+WriteSt()
DayStud
-stip
+CalcStip()
+WriteSt()

Visual C++. MFC. Программирование циклических процессов

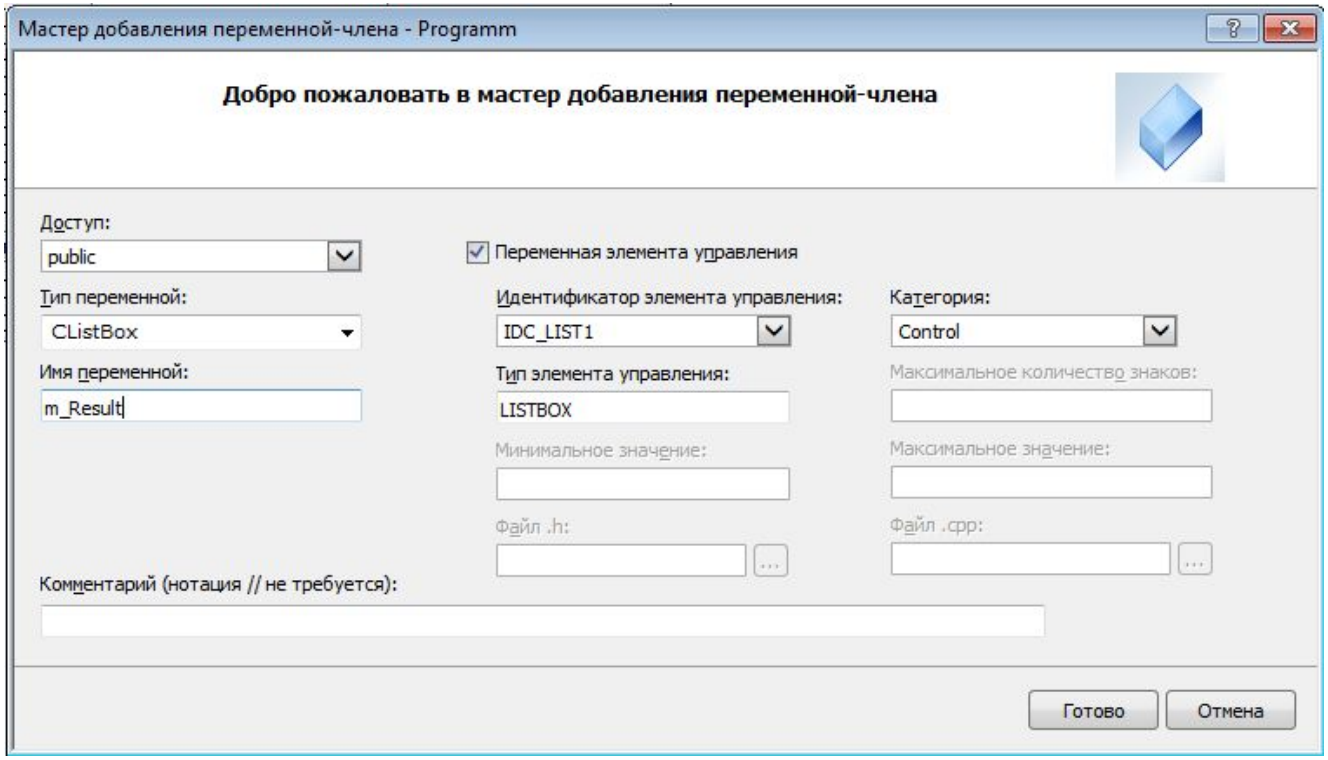
Переменная x меняется от xn до xk с шагом dx. Вывести таблицу значений x и $y=e^{\sin(x)}$, вычислить сумму и произведение y.

Создадим диалоговое окно (мое имеет название ListBProg) и разместим на нем следующие компоненты:

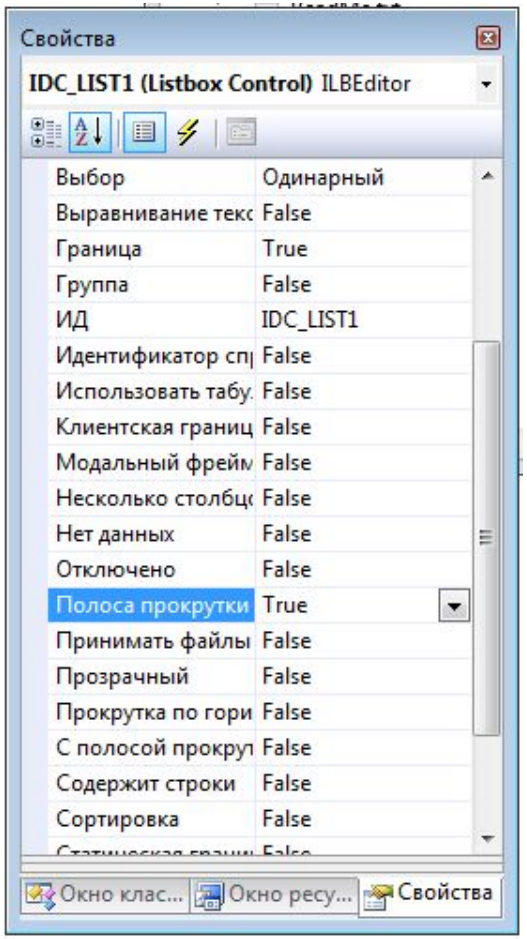
- 3 метки (*Static Text);*
- 3 поля ввода (*Edit Control);*
- 1 список (*List Box);*
- 3 кнопки (*Button).*



Добавим для полей ввода (*Edit Control) переменные m_edit_xn, m_edit_xk,m_edit_dx, которые будут возвращать значение float. В функцию кнопки «Выход» впишем строку OnOK() для того чтобы программа завершала свою работу по нажатию соответствующей клавиши.* Изменим ID кнопок «Решить» и «Очистить» на ID_Solve и ID_Clear, а также добавим переменную m_Result для компонента List Box



Установите в свойствах List Box значение «Сортировка» в false, так как иначе содержимое будет сортироваться по алфавиту, а «С полосой прокрутки» в true.



Зададим начальные значения полей ввода. Для этого откроем файл *Dlg.cpp и в функцию OnInitDialog перед оператором return впишем

```
    // TODO: добавьте дополнительную инициализацию
    m_edit_xn=-5;
    m_edit_xk=4;
    m_edit_dx=3;
    UpdateData(FALSE);
    return TRUE;  // возврат значения TRUE, если фокус не передан элементу управления
}
```

Щелкаем по кнопке «Решить» и вводим следующий текст.

```
void CProgrammDlg::OnBnClickedSolve()
{
    // TODO: добавьте свой код обработчика уведомлений
    int i; float x, xn, xk, dx, y, s, p;
CString S;
UpdateData(TRUE);
xn=m_edit_xn;
xk=m_edit_xk;
dx=m_edit_dx;
for (s=0, p=1, x=xn, i=0; x<=xk; x+=dx, i++)
{
y=exp(sin(x)); s+=y; p*=y;
S.Format (_T("x=%g y=%g"),x,y);
m_Result.AddString(S);
S.Format(_T("s=%g p=%g"), s, p);
m_Result.InsertString(0,S);
}
}
```

А для кнопки «Очистить» определите такой код:

```
void CProgrammDlg::OnBnClickedClear()
{
    // TODO: добавьте свой код обработчика уведомлений
    int j, n;
    n=m_Result.GetCount();
    for (j=0; j<n; j++)
    m_Result.DeleteString(0);
}
```