

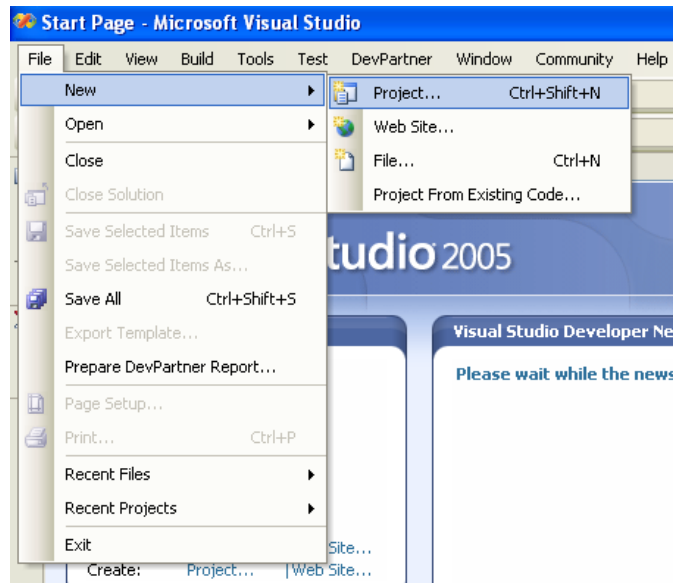
Лабораторная работа №1

Абстрактные классы. Полиморфизм.

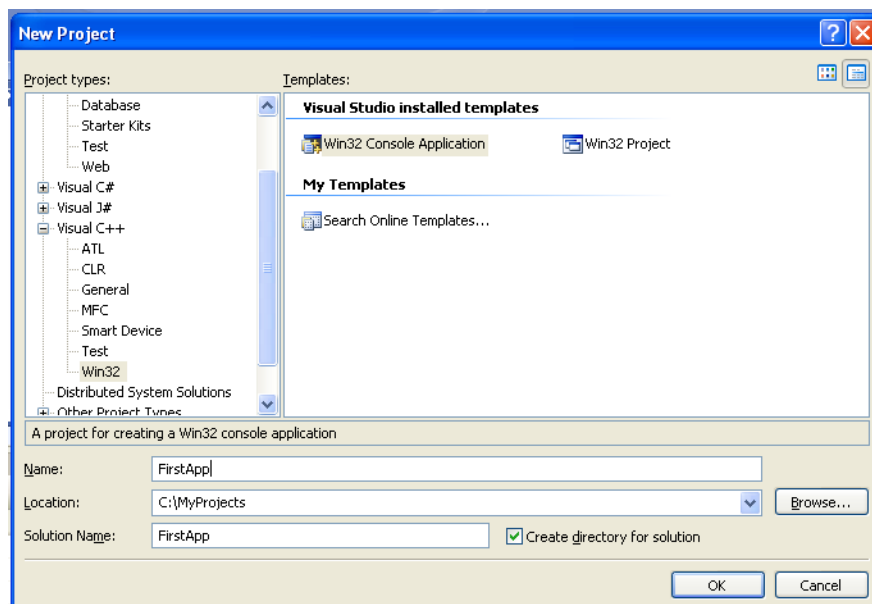
Цель работы: освоить основные принципы полиморфизма и позднего связывания в C++.

Ход работы:

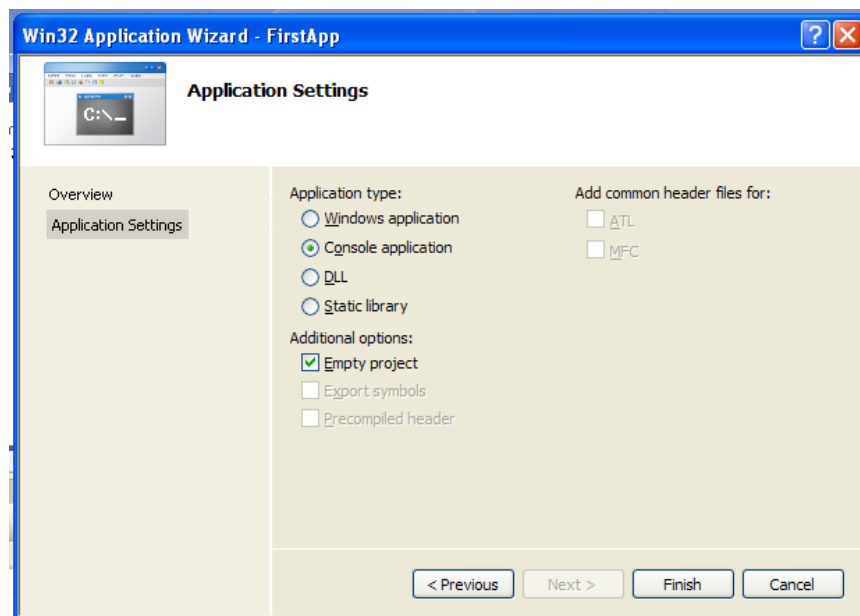
1. Создаем новый проект.



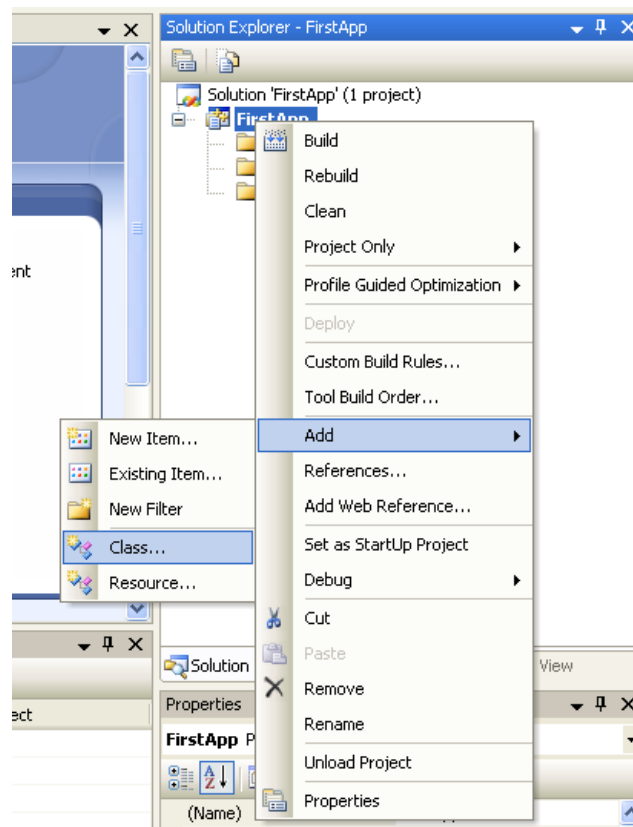
2. Выбираем тип проекта «Win32», консольное приложение «Win32 Console Application». Назначаем имя проекту.

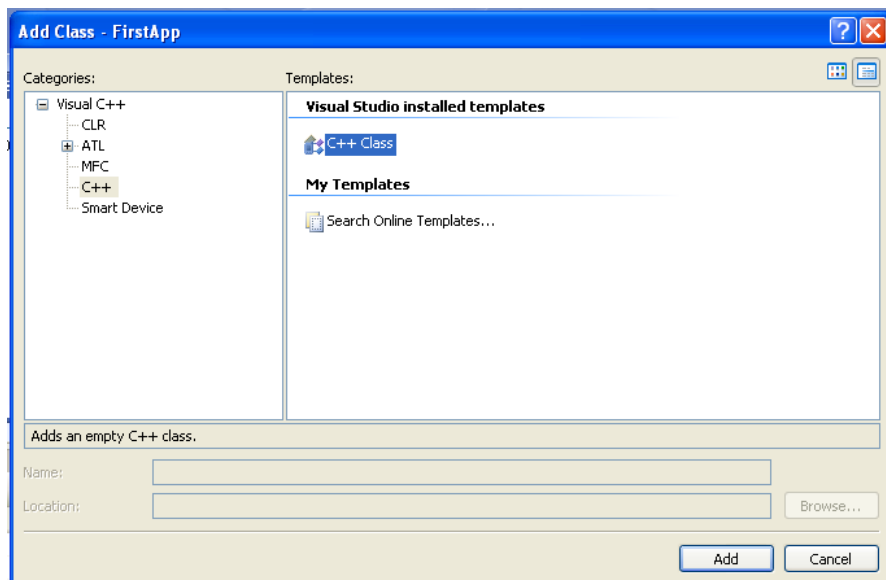


3. На вкладке «Application Settings» выбираем тип «Console Application». Проект должен быть пустым. За это отвечает галочка «Empty project».

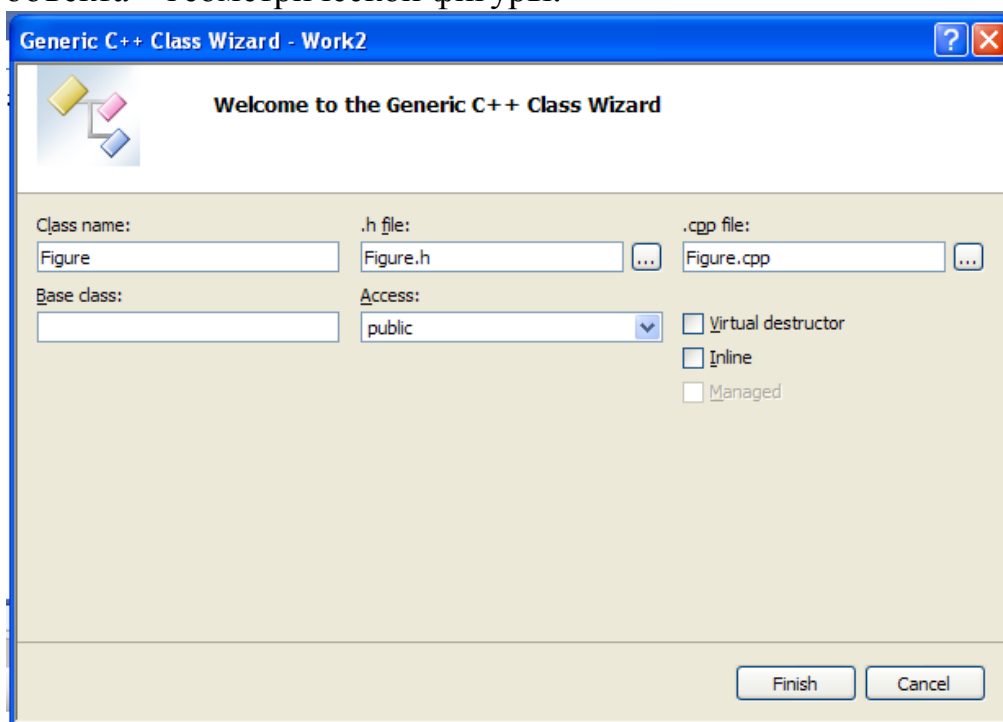


4. В окне «Solution Explorer» добавляем к проекту новый класс.





5. Добавьте в проект новый класс, отвечающий за общую структуру объекта – геометрической фигуры.



6. Кроме автоматически созданных конструктора и деструктора добавьте к классу еще 2 метода:

GetSquare – возвращающий площадь фигуры;

PrintName – печатающий на экране название фигуры.

Описание класса должно выглядеть примерно следующим образом:

```
class Figure
{
public:
    Figure(void);
    ~Figure(void);
    double GetSquare();
    void PrintName();
};
```

7. Реализуйте методы класса следующим образом:

```
Figure::Figure(void)
{
    cout << "Figure Constructor called!" << endl;
}

Figure::~Figure(void)
{
    cout << "Figure Destructor called!" << endl;
}

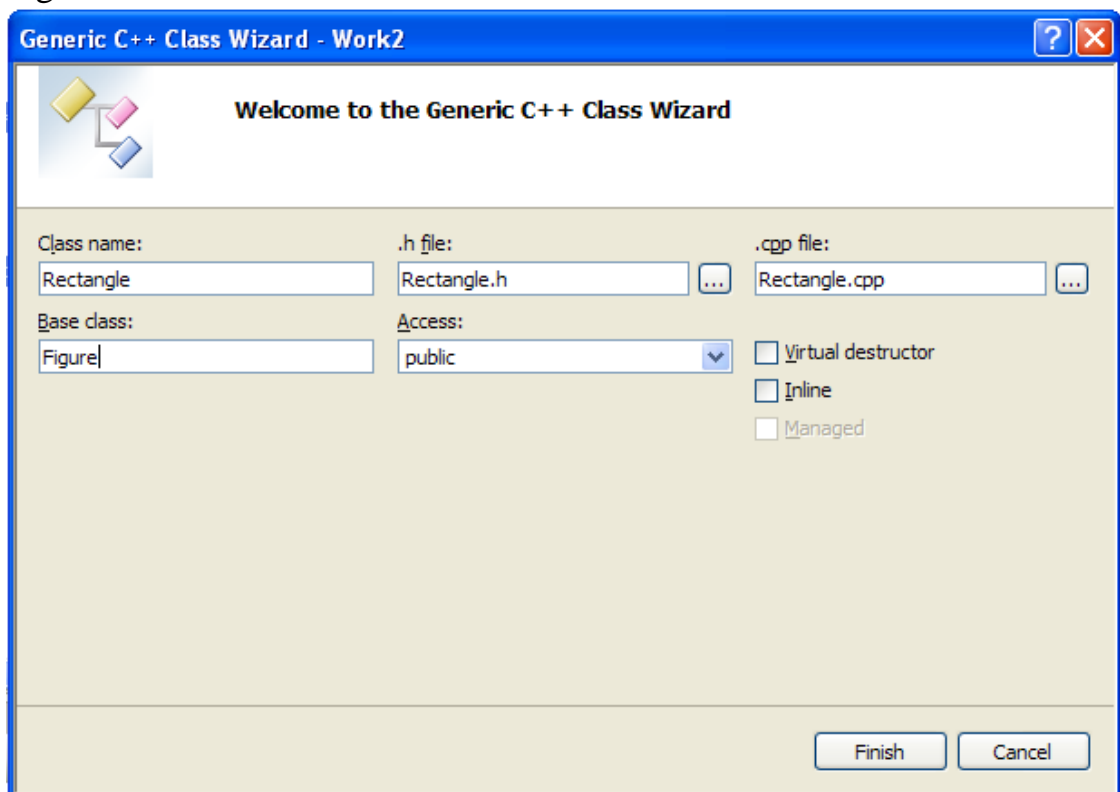
double Figure::GetSquare()
{
    cout << "Common Figure can't have a square!" << endl;
    return 0;
}

void Figure::PrintName()
{
    cout << "Abstract Figure" << endl;
}
```

8. Так как в классе Figure используется оператор cout, не забудьте добавить в файл с реализацией класса подключение библиотеки iostream:

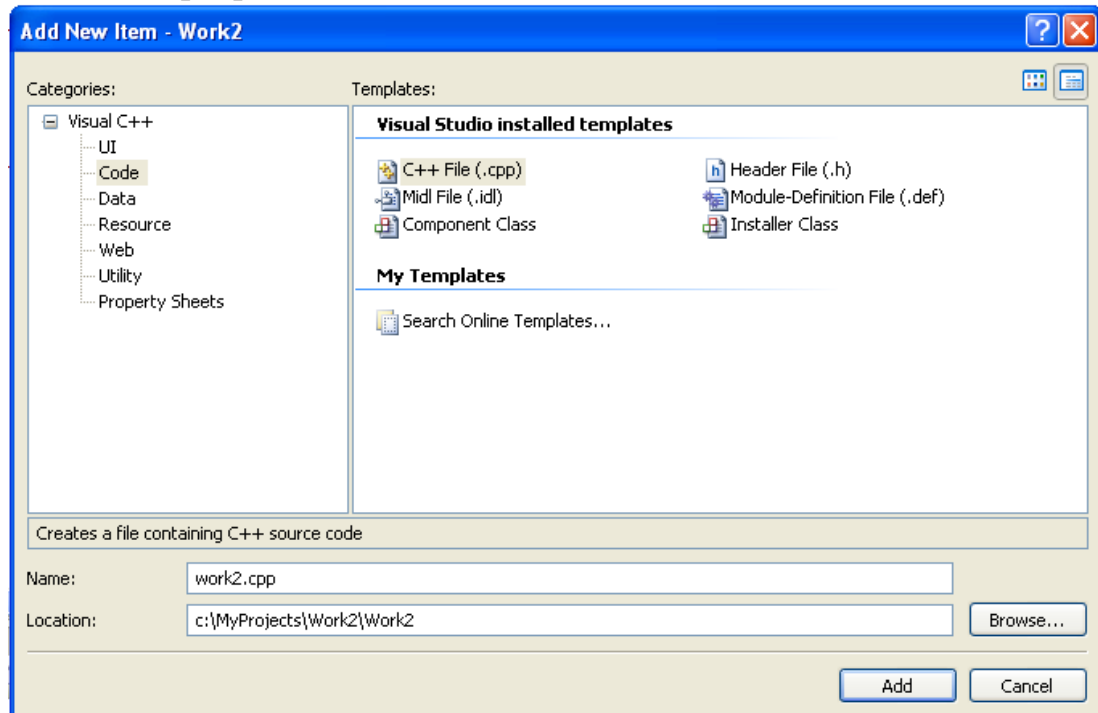
```
#include <iostream>
using namespace std;
```

9. По аналогии добавьте класс Rectangle, унаследованный от класса Figure.



В этом классе будут переопределены все методы таким образом, чтобы они сообщали об операциях с классом Rectangle, а не Figure. Что касается площади, то необходимо добавить в класс еще два поля:

- double a, b. Эти поля будут отвечать за длину и ширину прямоугольника. Метод GetSquare должен рассчитывать и печатать площадь.
10. Добавьте к проекту новый файл, в котором будет располагаться основная программа:



11. Создайте основную программу, в которой будут динамически создаваться объекты класса Figure и Rectangle. Далее будут вызываться

различные методы, а в конце эти объекты будут уничтожаться.

```
#include <conio.h>

#include "Figure.h"
#include "Rectangle.h"

void main ()
{
    Figure* f;
    Rectangle* r;

    f = new Figure();
    r = new Rectangle();

    r->a = 10;
    r->b = 20;

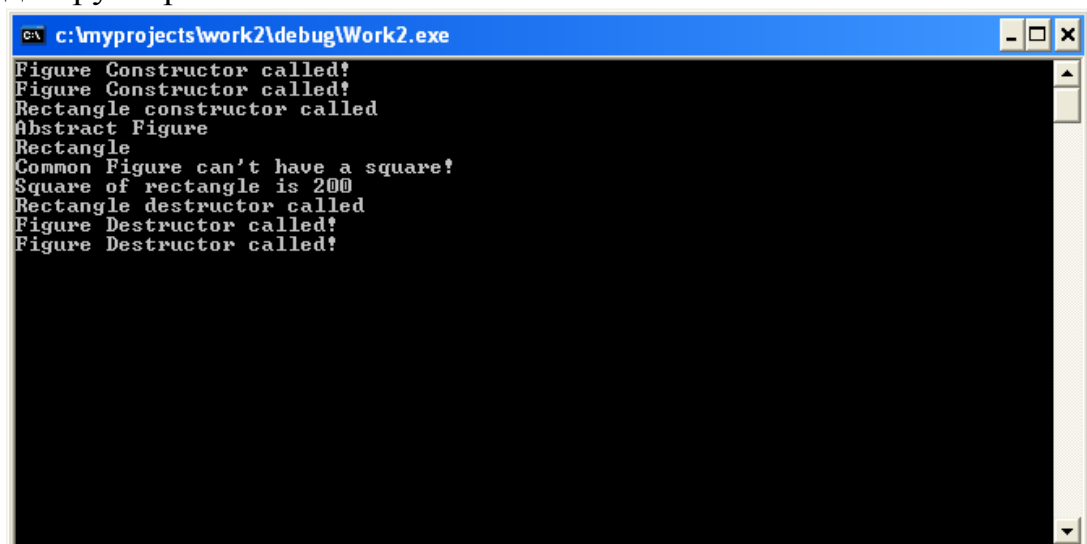
    f->PrintName();
    r->PrintName();

    f->GetSquare();
    r->GetSquare();

    delete r;
    delete f;

    getch();
}
```

12. Обратите внимание на порядок вызова методов, конструкторов и деструкторов:



```
c:\myprojects\work2\debug\Work2.exe
Figure Constructor called!
Figure Constructor called!
Rectangle constructor called
Abstract Figure
Rectangle
Common Figure can't have a square!
Square of rectangle is 200
Rectangle destructor called
Figure Destructor called!
Figure Destructor called!
```

13. Измените основную программу так, чтобы можно было понаблюдать явления, происходящие после преобразования дочернего класса

Rectangle к классу Figure.

```
#include <conio.h>

#include "Figure.h"
#include "Rectangle.h"

void main ()
{
    Figure* f;
    Figure* f2;
    Rectangle* r;

    f = new Figure();
    r = new Rectangle();

    r->a = 10;
    r->b = 20;

    f2 = (Figure*)r;

    f->PrintName();
    f2->PrintName();

    f->GetSquare();
    f2->GetSquare();

    delete f;
    delete f2;

    getch();
}
```

14. Пронаблюдайте за результатом. Что изменилось. Свои наблюдения и причину изменений изложите в отчете.

15. Добавьте в описание всех методов класса Figure ключевое слово virtual, кроме методов PrintName и конструктора:

```
class Figure
{
public:
    Figure(void);
    virtual ~Figure(void);
    virtual double GetSquare();
    void PrintName();
};
```

16. Пронаблюдайте за результатом. Что изменилось. Свои наблюдения и причину изменений изложите в отчете.

17. Приведите класс Figure в полный порядок, добавив ключевое слово virtual к методу PrintName.

18. Реализуйте класс очереди из предыдущей работы с одним отличием – элементом очереди теперь будет не int, а указатель на объект типа

Figure:

```
class Queue
{
public:
    Queue(void);
    ~Queue(void);

    Figure* f;
    void Put(Figure*);
    Figure* Get();
};
```

19. Реализуйте основную программу так, чтобы в очередь ставились несколько объектов типа Figure вне зависимости от того, какими фигурами они являются (пока доступен только класс Rectangle):

```
void main ()
{
    Figure *f1, *f2, *f3;
    Rectangle *r1, *r2, *r3;
    r1 = new Rectangle();
    r2 = new Rectangle();
    r3 = new Rectangle();
    r1->a = 1; r1->b = 2;
    r2->a = 5; r2->b = 7;
    r3->a = 15; r3->b = 3;

    f1 = (Figure*)r1; f2 = (Figure*)r2; f3 = (Figure*)r3;

    Queue q;
    q.Put(f1); q.Put(f3); q.Put(f2);
    q.ShowSquares();

    delete f1; delete f2; delete f3;
    getch();
}
```

20. Метод ShowSquares нужно добавить к описанию очереди. Этот метод выводит названия и площади всех фигур в очереди.

Индивидуальное задание

1. Реализуйте класс Circle для работы с кругом. В главной программе поставьте в очередь несколько кругов и прямоугольников.
2. Реализуйте класс Triangle для работы с треугольником. В главной программе поставьте в очередь несколько треугольников и прямоугольников.
3. Реализуйте класс Trapezia для работы с трапецией. В главной программе поставьте в очередь несколько трапеций и прямоугольников.
4. Реализуйте класс Romб для работы с ромбом. В главной программе поставьте в очередь несколько ромбов и прямоугольников.

5. Выполните задание №1, реализовав во всех классах вместо нахождения площади – нахождение периметра фигуры (для круга – длины окружности).
6. Выполните задание №2, реализовав во всех классах вместо нахождения площади – нахождение периметра фигуры (для круга – длины окружности).
7. Выполните задание №3, реализовав во всех классах вместо нахождения площади – нахождение периметра фигуры (для круга – длины окружности).
8. Выполните задание №4, реализовав во всех классах вместо нахождения площади – нахождение периметра фигуры (для круга – длины окружности).