Федеральное государственное бюджетное образовательное учреждение высшего

образования

«Чувашский государственный университет им. И.Н. Ульянова»

Кафедра вычислительной техники

Лабораторная работа № 6

Функция. Перегрузка функций. Шаблоны функций.

Вариант 9

Выполнил: Иванов В.С.

студент группы ИВТ-41-22

Проверил: кандидат технических наук

Обломов Игорь Александрович

Чебоксары, 2024

**Цель работы**: изучить классы в С++, получить практические навыки программирования задач с использованием классов.

**Теория**.

Класс – абстрактный тип данных, определяемый пользователем, и представляющий собой модель реального мира в виде данных и функций для работы с ними. Данные, содержащиеся в классе, называются полями, а функции – методами.

Ключевые слова (спецификаторы) private, public управляют видимостью компонентов класса. Слово private (для класса этот спецификатор по умолчанию) запрещает обращение к полям класса из вне, а public (обязательный спецификатор) - объявляет поля как общедоступные. Все компоненты, описанные после public, часто называют интерфейсом класса. Количество и порядок следования спецификаторов стандартом не оговаривается.

Поля класса имеют следующие специфические особенности:

- могут иметь любой тип, кроме типа объявляемого класса, но могут быть указателями или ссылками на данный класс;

- могут быть объявлены с модификатором const, при этом инициализируются с помощью конструкторов специального вида;

- могут быть объявлены с модификатором static, но никак не auto, register или extern.

Инициализация полей класса при описании не допускается.

Обращение к объектам класса допускается только через компоненты, описанные после спецификатора public.

Непосредственное обращение к скрытому полю запрещено. Сокрытие данных класса защищает данные класса от несанкционированного доступа через операцию присваивания или через функции, не принадлежащие данному классу.

Число полей и методов класса теоретически может быть любым.

***Задание №1-4.***

Определить простой класс, представляющий описание членов семьи, объявив поля для хранения фамилии, имени, отчества, возраста, пола, статуса (учащийся, рабочий, пенсионер). Объявить методы, позволяющие вводить данные для каждого член семьи, а также обеспечить вывод этих данных на экран.

Определить класс из задания №1 в теле произвольной функции, которая последовательно выводит данные о каждом члене семьи. Расширьте определение класса из первого задания, добавив в определение внешнего класса локальный класс, содержащий элементы, описывающие доходы членов семьи.

Дальнейшая модификация класса семейных отношений состоит в переопределении (добавлении) методов, не меняющих значения полей класса в константные. Объявите константные объекты данного класса и испытайте действие константных и не константных методов.

Расширьте класс семейных отношений объявлением 3-4 конструкторов, имеющих различное количество параметров и различные их типы, позволяющих инициализировать объекты различными способами.

class Family {

private:

string Name, Surname,Fathername;

int Age;

string Sex;

string Status;

public:

void Print();

Family(string Surnamev, string Namev, string Fathernamev) {

Surname = Surnamev;

Name = Namev;

Fathername = Fathernamev;

}

Family(string Sexv,string Statusv) {

Sex = Sexv;

Status = Statusv;

}

Family(int Agevv) {

Age = Agevv;

}

void setAge(int Age) {

this->Age;

}

void showage() const {

cout << "Age: " << Age << endl;

}

class Salary {

private:

int salary;

public:

/\*Salary(int salary) {

this->salary;

}\*/

int GetSalary() {

return salary;

}

int SetSalary(int salaryv) {

salary=salaryv;

return salary;

}

};

void Print1() {

Salary amount;

amount.SetSalary(2000);

cout << "Surname: " << Surname << "\t\tName: " << Name << "\t\tFathername: "

<< Fathername << "\nSalary: " ;

int result = amount.GetSalary();

cout << result << endl;

}

void Print2() {

cout << "Age: " << Age << endl;

}

void Print3() {

cout << "Sex: " << Sex << "\nStatus: " << Status << endl;

}

};

***Задание №5.***

Определить собственный класс, моделирующий целочисленные (вещественные) величины. Предусмотреть наличие константных и ссылочных полей.

class Value {

private:

const int constValue;

const int &addValue;

int &refValue;

public:

Value(int constvalue, int addvalue, int refvalue) : constValue(constvalue), addValue(addvalue), refValue(refvalue) {};

void Out() {

cout << "Sum = " << constValue + addValue + refValue << endl;

}

};

***Задание №6-10,16,17,18.***

Определить класс комплексные числа Complex, объектными полями которого являются поля Re и Im типа Float (вещественный тип), объявленный ранее. Определить конструктор со списком инициализации, обеспечивающим инициализацию объявленных полей.

Для класса, оперделенного в задании №6, определить копирующий конструктор, инициализирующий все поля объявляемого объекта.

Для класса, определенного в задании №6, определит внешнюю функцию, возвращающую в качестве результата объект типа Complex. В копирующем конструкторе предусмотреть вывод «Работает копирующий конструктор», который позволит убедится в в каких случаях вызывается данный вид конструктора.

Для класса комплексных чисел добавить конструктор с одним аргументом, инициализирующий действительную или мнимую часть комплексного числа, запрещающий преобразования.

Для класса комплексные числа определить методы, использующие переменную this.

Для класса комплекных чисел переопределить оператор доступа, возвращающий значения действительной и мнимой части комплексного числа.

Для класса комплексных чисел переопределить свои собственные операции new и delete.

Для класса компексные числа определит указатели на поля и методы класса. Показать их использование.

class Float {

float value;

public:

Float(float val) : value(val) {}

Float() {

//value = 3.14;

}

float Get() {

return value;

}

/\*Float(const Float& f) {

Float = f.value;

}\*/

Float(const Float& f) :value(f.value) {};

Float operator+(Float& a) {

return this->value + a.value;

}

/\*Float operator<<(ostream& stream) {

stream << value;

}\*/

friend ostream& operator<<(ostream& b\_1, const Float& b) {

b\_1 << b.value << endl;

return b\_1;

}

};

class Complex {

public:

Float Re;

Float Im;

Complex(float re, float im) : Re(re), Im(im) {};

Complex(float Re) {

this->Re = Re;

}

Complex() {}

Complex\* operator->() {

cout << "Operator" << endl;

return this;

}

Float\* Complex\_18() {

return &Re;

}

Float Complex\_18\_summ() {

return Re + Im;

}

void Out\_1() {

cout << Re.Get() << "/" << Im.Get() << endl;

}

void Out\_2() {

cout << Re.Get() << endl;

}

Complex(const Complex& c) : Re(c.Re), Im(c.Im) {

cout << "Copy constructor" << endl;

}

//Complex(const Complex& c);

Complex& operator = (const Complex& c) {

Re = c.Re;

Im = c.Im;

return \*this;

}

void\* operator new(size\_t size) {

cout << "New operation for class Complex, bytes need: " << size << endl;

void\* storage = malloc(size);

if (NULL == storage) {

cout << "Lack of memory" << endl;

}

return storage;

}

void operator delete(void\* p) {

cout << "Operation delete for class Complex" << endl;

free(p);

}

};

Float\*(Complex::\* Complex\_18\_ptr)() = &Complex::Complex\_18;

Float(Complex::\* Complex\_18\_summptr)() = &Complex::Complex\_18\_summ;

int main(){

Complex ex\_18(9.8, 14.6);

cout << "Sum: " << (ex\_18.\*Complex\_18\_summptr)() << endl;

Complex\* ptr\_18 = &ex\_18;

cout << "Sum: " << (ptr\_18->\*Complex\_18\_summptr)() << endl;

cout << "Re:" << \*(ex\_18.\*Complex\_18\_ptr)() << endl;

cout << "Re:" << \*(ptr\_18->\*Complex\_18\_ptr)() << endl;

return 0;

}

***Задание №11-13.***

Определить два простых класса, объявив составную функцию одного из них, дружественной для другого.

Для задания №11 класс Test1 дополнить дружественными функциями – операциями ввода и вывода.

Переопределите самостоятельно для данного класса унарную префиксную операцию как дружественную и как внешнюю.

class Test\_2;

class Test\_1 {

int test;

public:

int num;

Test\_1(int num) {

this->num = num;

}

void fun(Test\_2);

friend istream& operator>>(istream& a\_1, Test\_1& a) {

cout << "Enter num: " << endl;

a\_1 >> a.num;

return a\_1;

}

friend ostream& operator<<(ostream& b\_1, const Test\_1& b) {

b\_1 << "Num:" << b.num << endl;

return b\_1;

}

/\*friend Test\_1& operator++(Test\_1& tst, int) {

++tst.num;

return tst;

}\*/

};

Test\_1 operator++(Test\_1& tst, int) {

++tst.num;

return tst;

}

class Test\_2 {

public:

friend void Test\_1::fun(Test\_2);

};

***Задание №14.***

Определите собственный класс, содержащий поля-указатели на стандартные типы, требующие выделения места в динамической области памяти. Переопределить операцию присваивания для данного класса.

class Test\_14 {

int\* Ext;

public:

Test\_14() {

Ext = new int;

}

Test\_14(int a) {

Ext = new int(a);

}

~Test\_14() = default;

//delete Ext;

//}

Test\_14& operator=(const Test\_14& Other) {

/\*if (this != &Other) {

delete Ext;

}\*/

Ext = new int;

Ext = Other.Ext;

return \*this;

}

void Out() {

cout << "Element: " << \*Ext << endl;

}

};

***Задание №15.***

Определите класс однонаправленный список, к элементам которого можно обращаться по индексу.

class Ex\_15 {

int size;

int\* p;

public:

explicit Ex\_15(int n = 5);

Ex\_15(const int arr[], int n) :size(n) {

p = new int[size];

for (int i = 0; i < size; i++) {

p[i] = arr[i];

}

}

~Ex\_15() {

delete[]p;

}

int& operator[](int i) {

if (i<1 || i>size) {

cout << "Out of array" << endl;

exit(0);

}

return p[i];

}

void Out() {

for (int i = 0; i < size; i++) {

cout << p[i] << ' ';

}

cout << endl;

}

};

**Вывод**: изучил классы в С++, получил практические навыки программирования задач с использованием классов.