===============================================================ПРОГА 3 СЕМ===============================================================

4 лабы

1 лаба наследование

2 лаба визуальная среда

3 и 4 лаба кусок синтаксического анализатора

1. Специальные функции - элементы класса

К специальным функциям-элементам класса кроме конструктора и деструктора относятся конструктор копии операция присваивания, конструкторы преобразования. Кроме того к таким функциям принято относить функция, обеспечивающие преобразование объектов класса в другие типы.

Конструктор копии - это конструктор специального вида, который имеет ровно один параметр - ссылку на объект этого же класса. Если конструктор копии не создавался, то генерируется конструктор копии по умолчанию, создающий буквальную копию объекта.

class Salieva

{

int a, b;

public:

Salieva(int aa, int bb) : a(aa), b(bb)

{

}

Salieva (Salieva & proval)

{

a = proval.a\*5;

b = proval.b-6;

}

Salieva (int starosta)

{

a = starosta + 5;

b = starosta - 6;

}

operator int ()

{

return 2\*a+3\*b-11;

}

Salieva & operator=(const Salieva &);

};

Salieva & Salieva::operator=(const Salieva & elkin)

{

a = elkin.a;

b = elkin.b-666;

return \* this;

}

void main(void)

{

Salieva diskretka (25, 25), frezer(30, 30);

Salieva evgenich = diskretka; // конструктор копии

frezer = diskretka; // операция присваивания

Salieva myagkov = 54;

int zaraza;

zaraza = (int)myagkov;

}

Операция присваивания - это функция с именем operator=, которая принимает в качестве своего единственного параметра ссылку на объект данного класса. Эта функция вызывается тогда, когда одному уже существующему объекту класса присваивается значение другого существующего объекта класса. Если операцию присваивания не определить явно, то выполняется автоматическое присваивание, создающее точную копию объекта.

Среди конструкторов класса отдельную группу составляют конструкторы преобразования, в которую относят все конструкторы с одним параметром.

При необходимости автор класса может запретить вызов конструктора преобразования с использованием знака =. Для такого запрета в объявлении конструктора должно быть добавлено ключевое слово explicit.

В классе можно определять элементы функции, которые будут обеспечивать явное преобразование типа данного класса в другие типы.

Заголовки таких функций имеют вид operator имя\_типа () {}. Такие функции не могут иметь параметров и для них не определяется тип возвращаемого значения, так как подразумевается тип, стоящий за ключевым словом operator имени функции.

proverka ploho = 7;

2. Понятие о дружественности.

Язык C++ позволяет разрешить доступ к элементам класса, объявленным с метками private или protected, функциям других классов или даже функциям, не относящимся ни к каким классам. Это делается при помощи ключевого слова friend.

class proverka

{

Int a, b;

friend class rabota;

friend void otchislenie::idb2204(int);

friend void print (const proverka&);

public: ~proverka ();

}

Строки, начинающиеся с ключевого слова friend, можно размещать в теле класса не глядя на метки доступа. Дружественность не взаимна (если класс А объявил класс В другом, то это не означает что А является другом для В). Дружественность не передается по наследству (если класс А объявил В другом, то это не означает, что наследники А автоматически считают В другом, а также, что А считает друзьями наследников В).

void print (const proverka & diskretka)

{cout << diskretka.a << “#” << diskretka.b; }

3. Наследование классов

Язык C++ позволяет классу наследовать элементы данных и методы из одного или нескольких других классов. Новый класс называется производным классом. Класс, элементы которого наследуются, называется базовым классом. В свою очередь производный класс может служить базовым для другого класса.

За счет наследования реализуется один из базовых принципов ООП – инкапсуляция. Смысл инкапсуляции состоит в том, что некоторые свойства и поведение объектов объединяются в одно целое, причем это целое закрыто для внешней обработки.

Синтаксис объявления класса-наследника имеет следующий вид:

class имя1 : ключ\_доступа имя2

{ };

имя1 – наследник,

имя2 – базовый класс

Ключ доступа, стоящий в заголовке определения класса-наследника может быть public private или protected. По умолчанию принимается private при использовании слова class и public при использовании ключевого слова struct.

Этот ключ доступа влияет на уровень доступа к элементам базового класса через объекты производного класса. Доступ определяется в соответствии с более жестким ключом доступа из пары: ключ доступа в заголовке класса наследника и ключ доступа в теле базового класса.

При наследовании не происходит автоматического наследования конструкторов.

НЕРАБОЧИЙ ПРИМЕР:

class prolet: public proverka

{

char\* stroka;

public: prolet (char\*, int, int);

friend void print (const proverka &); - friend необязательно, т.к. print не обращается к эл. проверки

void print1 ()

{

proverka zaval (a,b) – НЕРАБОТАЕТ, т.к. a и b -

print (zaval);

print(“%s”, stroka);

}

};

prolet::prolet(char \*str, int a1, int b1):proverka (a1,b1)

{strcpy (stroka, str);}

void main ()

{

prolet poezdka (“Psel von”, 25, 25);

poezdka.print1();

}

Этот пример будет работать только в случае, если в базовом классе элементы данных будут объявлены как protected, а не как private.

С++допускает так называемое сложное наследование, то есть одновременное наследование от двух и более базовых классов.

class leonid: public salieva, protected elkin

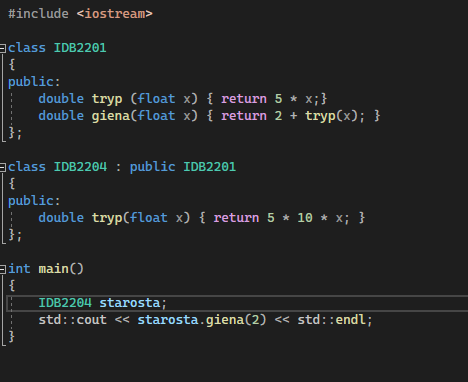
{ };

При этом в иерархии классов может оказаться так, что класс косвенно унаследует несколько элементов данных от одного и того же базового класса.

4. Виртуальные функции

Среди особенностей ООП выделяют полиморфизм – способность различной реакции на внешне абсолютно одинаковый запрос в зависимости от того, от кого этот запрос исходит. Механизм виртуальных функций C++ как раз является примером проявления полиморфизма.

Пример:



При отсутствии в определении функций ключевого слова virtual в данном примере полиморфной реакции не наблюдается. Если же добавить это ключевое слово перед типом извращаемого значения в базовом классе(также можно прописать это слово и во всех наследниках), то полиморфная реакция будет наблюдаться. Виртуальность будет сохраняться до тех пор, пока в цепочке наследования не изменяется список параметров. При этом виртуальные функции могут замещать и невиртуальные функции, но такая особенность не приветствуется разработчиками. Более того, принято об я блядь чистые виртуальные функции, у которых нет тела.

virtual double tryp (float) = 0;

Такие функции создаются в базовом классе, чтобы затем переопределяться в классах, производных от него. Класс, об ебля ющий одну или несколько чисто виртуальных функций, называется абстрактным базовым классом. У такого класса нельзя создать представителя, а нужен он только в качестве базового для дальнейшего порождения классов, которое будет реализовывать его чисто виртуальные функции.

5. Динамические распри деления памяти в C++.

В C++ для управления динамическим распределением памяти используются операции new и delete для простых переменных, а также new [] и delete [] для массивов. На ряду с этими механизмами сохранена возможность вызова стандартных функций языка си – malloc(), calloc(), free(), и т.д.

Int \* idb2203;

Idb2203 = new int;

……………

delete idb2203

idb2203 = new int[25];

…………

delete [] idb2203;

При динамическом распределении памяти можно сразу и инициализировать переменные.

int \* elkin = new int (25);

cout << \*elkin << endl;

Если операция new вызывается для представителя какого-либо класса, то одновременно вызывается и конструктор.

proverka \* dvoechnik;

dvoechnik = new proverka (25, 25);

Для переменных, для которых память выебляется подобным образом =, конструктор на занимается выеблением памяти для объектов, а деструктор ее не высвобождает. Этим занимаются new и delete.

Класс может определять свои собственные операции new и delete.

void \* proverka :: operator new [] (…) {}

6. Стандартная схема проекта

Заказчик – бизнес-требования (пишутся в терминах решаемой проблемы, часто их пишет не заказчик, а бизнес-аналитк).

Аналитик – техническое задание (структуры хранения данных, алгоритмы их обработки, описание интерфейсов).

=========================================================================

В случае, когда речь идёт о разработке новой подсистемы, ТЗ попадает к архитектору, который осуществляет выбор программного обеспечения для реализации(среды).

После выбора среды реализации (сред) задача попадает программисту ( группе программистов). После завершения работы над программой и элементарной проверки, программа попадает к тестировщику.

Как правило, тестировщик запускает тест, составленный аналитиком, если программа на тестовых примерах работает некорректно, то она возвращается в разработку.

После завершения тестирования программа документируется (инструкция пользования, инструкция администратора и т.д.).

Затем наступает этап внедрения (сначала тестовая эксплуатация, затем полноценная).

Порядок работы, который описан выше, может уменьшаться, часть этапов при некоторых стратегиях разработки может отсутствовать. Полная схема иногда носит названия каскадной разработки.

Наравне с каскадной разработкой используются разные варианты гибких методологий ( обобщенное название agile).

Среди этих методологий особо выделяется SCRUM.

7 понятие о восходящем подходе.

Как правило проектирование программного продукта производится сверху-вниз, а вот программирование часто выполняется снизу вверх. При таком подходе вначале изготавливаются модули самого низкого уровня, затем они тестируются и происходит переход на уровень вверх.

Основная трудность при восходящем подходе состоит в том, что каждый модуль может правильно работать при выполнении тестов рассчитанных на проверку данного модуля, но не работать при тестах на взаимосвязь модулей (при включении а общий программный проект). Это может быть связано с изменением ТЗ в ходе работы над проектом, с разной интерпретацией разными программистами тех или иных частей ТЗ с особенностью реальных данных и т.д.

Эти трудности обычно концентрируются на последней фазе работы над программным проектом в условиях острой нехватки времени.

При возникновении подобных ситуаций применяются следующие решения:

- вносятся изменения в уже написанных модули

- перепрограммируется большая часть модулей

-вносятся изменения в проект, чтобы сделать его соответствующим готовой программе

Д.з.

Найти известнейшую картинку с качелями как хотел заказчик, как хотел аналитик, так сделали в тз