А. А. Воропаев, М. А. Аксёненко, О. Э. Марчук

СОЗДАНИЕ УНИВЕРСАЛЬНОГО «ТИПА ДАННЫХ» ДЛЯ ИСПОЛЬЗОВАНИЯ В ЯЗЫКЕ С++

Введение

В ходе выполнения лабораторных работ во втором семестре, мы познакомились с функционалом выделения памяти. Заинтересовавшись этим мы решили углубиться в данном направлении и решили создать свой класс схожий по принципу работы с классами в Python. В языке Python у каждого значения есть тип, но нет необходимости явно указывать типы переменных. Основываясь на первом присвоении значения переменной, Python определяет её тип и в дальнейшем отслеживает его самостоятельно. Классы в С++ — это абстракция описывающая методы, свойства, ещё не существующих объектов. Объекты — конкретное представление абстракции, имеющее свои свойства и методы. Созданные объекты на основе одного класса называются экземплярами этого класса. Эти объекты могут иметь различное поведение, свойства, но все равно будут являться объектами одного класса. Методы класса — это его функции. Свойства класса — его переменные. Конструктор класса — это специальная функция, которая автоматически вызывается сразу после создания объекта этого класса. Он не имеет типа возвращаемого значения и должен называться также, как класс, в котором он находится. Деструктор класса вызывается при уничтожении объекта. Деструктор не имеет входных параметров. Структура - это совокупность переменных, объединенных одним именем, предоставляющая общепринятый способ совместного хранения информации. Объявление структуры приводит к образованию шаблона, используемого для создания объектов структуры.

I. Принцип и пример работы

В С++ существует особый тип указателей – указатели типа void или пустые указатели. Эти указатели используются в том случае, когда тип переменной не известен. Так как void не имеет типа, то к нему не применима операция разадресации (взятие содержимого) и адресная арифметика, так как неизвестно представление данных. Тем не менее, если мы работаем с указателем типа void, то нам доступны операции сравнения. В зависимости от присваиваемых данных вызывается определённый оператор присваивания, далее входные данные и их тип записываются в класс. Также в классе каждый оператор(+-=/ и тд.) по отдельности перегружается для каждого из типов. Для контроля типа данных используется спецификация списка имён(enum). А как хранилище данных используется void*.

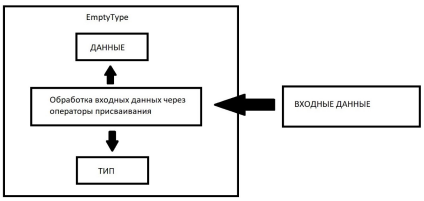

Рис. 1 – Принцип работы оператора присваивания

Использовать наш класс можно как обычную переменную без привязки к типу данных.

```
int main()
{
    EmptyType array_;
    array_ = { 12, 23, 22, 1 };
    EmptyType i;
    for (i = 0; i < 4; i++) {
        cout << " array[" << i << "]: " << array_[i] << endl;
    }
    return 0;
}
```

```
Консоль отладки Microsoft Visual Studio
array[0]: 12
array[1]: 23
array[2]: 22
array[3]: 1
```

Рис. 2 – Пример использования

II. Заключение

Таким образом был создан пользовательский класс, который позволяет работать с данными без привязки для пользователя к типу данных. На самом деле, создать полностью идеальный класс очень трудоёмкий процесс. В этой работе мы хотели показать, как видим реализацию данного класса через знания полученные в процессе обучения во втором семестре.

Список литературы

1. Роберт Лафоре. Объектно-ориентированное программирование в С++.
2. Изучаем Python. Том 1. Марк Лутц

Воропаев Андрей Александрович, студент 1 курса ФИТиУ, igrakkaunt@gmail.com.
Аксёненко Максим Александрович, студент 1 курса ФИТиУ, aksenenko2208@gmail.com.
Марчук Олег Эдуардович, студент 1 курса ФИТиУ, oleg_mar@list.ru.
Научный руководитель: Кривоносова Татьяна Михайловна, доцент кафедры вычислительных методов и программирования Белорусского государственного университета, krivonosova@bsuir.by.