ГУАП

guap.ru

Информатика. Информационные технологии

План лекции

- Язык программирования C++
  - Указатели
  - Адресная арифметика
  - Ссылки
  - Ошибка сегментации
- Объектно-ориентированное программирование
  - Структуры
  - Классы

Указатели

- Указатель (pointer) – переменная которая содержит адрес другой переменной
- указатели используются для косвенной адресации, так как указатель напрямую не связан с данными в памяти
- Указатель можно объявить на любой из встроенных типов данных (int, char, long, short …)
- Указатель можно объявить на любой объект (класс или структуру)

```
int myarray[]={5, 6, 7, 8, 9, 10};
int myVariable=myarray[3];
```

```
int myarray[]={5, 6, 7, 8, 9, 10};
int* ptr = array;
int myVariable = ptr[3];
```

Указатель содержит адрес начала массива и таким образом указывает на массив

Разыменование указателей

- Разыменование (dereferencing) – получение получение содержимого памяти, на который ссылается указатель

```
int x =20;
int* ptrx = &x;
//…
int z = *ptrx;
```

Присвоение переменной x значение 20

Инициализация указателя ptrx с указанием ее значения как адреса переменной x

Разыменование указателя ptrx с целью извлечения значения переменной x и записи в переменную z

& - оператор взятия адреса

Математические операции над адресами памяти - Адресная арифметика

- Указатели могут учувствовать в математических операциях
- сложение, вычитание, инкремент (i++), декремент (i--)
- В зависимости от типа данных операции над ними происходят несколько по разному:
  - размер данных типа int 4 байта => при прибавлении единицы к адресу переменной типа int будет отображаться содержимое ячейки памяти смещенное на 4 байта
  - Увеличение на адреса единицу означает, что мы хотим перейти к следующему объекту в памяти, который находится за текущим и на который указывает указатель. А уменьшение на единицу означает переход назад к предыдущему объекту в памяти.

Адресная арифметика

- Для указателя типа **short** эти операции изменяли бы адрес на 2, а для указателя типа **char** на 1.
- При вычитании или сложении будет выполняться операция при которой учитывается размер числа в байтах
  - например при прибавлении 2 к числу типа char будет производиться операция по смещению указателя на 2*1байт, а при смещении указателя типа double на 2*8бит
- вычитание адресов даст информации о размере занятой памяти между двумя адресами.

Тип данных	Размер в байтах
char	1
unsigned char	1
short	2
unsigned short	2
long	4
unsigned long	4
int	4
unsigned int	4
float	4
double	8
bool	1

Ссылки

- Ссылка (reference) – это специальный тип указателя, который позволяет работать с указателем как с обычным объектом
- Ссылка объявляется при помощи специального оператора ссылки, & (тот же оператор, что и для взятия адреса)

ссылка

```
int x = 5;int y = 6;
int &myref = x;
```

указатель

```
int x = 5;int y = 6;
int *mypointer;
mypointer = &x;
mypointer = &y;
*mypointer = 10;
```

Отличия ссылок и указателей

- **Указатель может быть переназначен любое количество раз**, в то время как **ссылка после привязки не может быть перемещена на другую ячейку памяти**.
- **Указатели могут указывать "в никуда"** (быть равными NULL), в то время как ссылка всегда указывает на определенный объект. GCC может без выдачи предупреждений обработать код наподобие int &x = *(int*)0;, однако поведение подобного кода может быть непредсказуемым.
- Вы **не можете получить адрес ссылки**, как можете это делать с указателями.
- **Не существует арифметики ссылок**, в то время как существует арифметика указателей. Однако есть возможность получить адрес объекта, указанного по ссылке, и применить к этому адресу арифметику указателей (например &obj + 5).

Ошибки сегментации

- Пример ошибки сегментации:

```
int * adr = (int *) 0;
*adr = 1;
```

- Причины возникновения ошибок сегментации:
  - недостаток памяти
  - указатель нулевой,
  - указатель указывает на произвольный участок
  - указатель указывает на удалённый участок памяти.

Зачем используются ссылки и указатели

- работа с динамической памятью
- расширение возможностей функций: Функция всегда возвращает только одно значение, но объект измененный по ссылке может рассматриваться как дополнительно возвращаемая величина.
- ключевое слово const перед переменной запрещает
- Размещение объектов (структур и классов) в динамической памяти

Динамическое и локальное размещение данных в памяти

- Локальное распределение данных (local allocation) – память для переменной или объекта выделяется в программном стеке.
- Стек (stack) – область оперативной памяти резервируема] программой при запуске.
- Динамическое распределение (dynamic allocation) память выделяется в “куче”
- Куча “heap” ‘это вся виртуальная память компьютера (оперативная память и место на жестком диске (файле подкачки))

Особенности использования программного стека и кучи.

- Особенности использования программного стека:
  - Память в стеке выделяется очень быстро
  - Стек имеет фиксированный размер, который не меняется в процессе воронения программы
  - Переполнение стека вызывает непредвиденные ошибки и часто приводит к зависанию программы
  - указатели располагаются в стеке
- Особенности использования программного динамической памяти:
  - Возможно использовать объекты большого объема
  - динамическое выделение памяти, возможность ее очитки

Операторы new и delete

- new – оператор используется для размещения переменной в куче

```
char buff[80];
char* bigBuff = new char[4096];
```

- delete – оператор используется для очистки памяти, выделенной

```
SomeObject* myObject = new SomeObject;
// использование myObject
delete myObject; // удаление переменной
```

Каждому **new** должен соответствовать свой **delete**
Если не освободить всю память, распределенную с помощью оператора new, программа вызовет утечку памяти в системе

Объекты в C++: Структуры

- Структура – набор взаимосвязанных данных, объединенных в единое целое и именуемое одним ключевым словом.
- Для использования структуры ее необходимо сначала объявить, а затем создать экземпляр структуры.
- структура содержит данные разных типов

```
struct mailListRecord {
        char firstName[20];
        char lastName[20];
        char address[50];
        char city[20];
        int zip;
};
```

Экземпляр структуры

```
mailListRecord record;
srtcpy(record.firstName,”Ivan”);
srtcpy(record.lastName, “Ivanov”);
strcpy(record.address,”Moscow, Kremlin”);
strcpy(record.city,”Moscow”);
record.zip=101000;
};
```

Объекты в C++: Классы в C++

- **Классы** — это некоторые описания, схемы, чертежи по которым создаются объекты.
- **Классы в C++** — это абстракция описывающая методы, свойства, ещё не существующих объектов.

```
// объявление классов в C++
class /*имя класса*/
{
 private:
 // список свойств и методов для использования внутри класса /
 public:
 // список методов доступных другим функциям и объектам программы /
 protected:
 /*список средств, доступных при наследовании*/
};
```

уровни доступа

private – закрытый

public – открытый

protected - защищенный

Уровни доступа к членам класса

- Уровни доступа к членам класса определяют способ работы пользователя с классом
- **private** – закрытый – здесь содержится информация, которая скрыта от пользователя данного класса, и таким образом защищена от модификации.
- **public** – открытый – доступ к этим членам класса является открытым и они доступны для пользователя
- **protected** – защищенный – к этим членам класса пользователь обратиться не может, однако они могут быть доступны для классов, которые являются производными данного

По умолчанию члены класса имеют уровень доступа private. Если не использовать модификаторы доступа, то все данные и функции будут закрытыми

Конструкторы и деструкторы

- Конструкторы - функция которая вызывается при создании экземпляра класса
- Конструктор вызывается для инициализации переменных членов класса, выделения памяти и выполнения необходимых действий перед началом использования класса
- Имя конструктора должно совпадать с именем класса
- Класс может иметь несколько конструкторов

- Деструкторы – специализированная функция, которая вызывается перед уничтожением экземпляра класса
- Деструктор занимается освобождением памяти
- Класс может содержать не более одного деструктора
- Название функции деструктора такое же как и у конструктора, но оно содержит знак ~

Пример класса

```
class Reminder // имя класса
{
private: // спецификатор доступа private
            int day, // день
            int month, // месяц
            int year; // год
public: // спецификатор доступа public
            Reminder() // конструктор класса
            ~Reminder() // деструктор класса
            int SetDay   = date_day; // инициализация день
            int SetMonth = date_month; // инициализация месяц
            int SetYear  = date_year; // инициализация год
}; // конец объявления класса
```

Наследование (inheritance)

- Наследование – создание нового класса путем добавления новых членов к уже существующему классу
- Класс, к которому добавили новые члены, называется базовым классом (base class), а вновь созданный класс – производным классом (derived class)

```
class Airplane {
        public:
                Airplane();
                ~Airplane();
                int Speed();
                int Altitude();
        protected:
                void TakeOff();
                void Land();
        private:
                int speed;
                int altitude;
                int status;
};
```

Наследование

```
class MilitaryPlane : Airplane {
        public:
                Militaryplane();
                ~ Militaryplane();
        protected:
                void TakeOff();
                void Land();
        private:
                Mission theMission;
};
```

Пример наследования