43. Понятие указателя.

Указатель – специализированный объект предназначенный для хранения адресов областей памяти.

Когда компилятор обрабатывает определение переменной то в памяти выделяется область в соответствии с типом переменной и записывается в эту область указанное значение. Программист может определять собственные переменные для хранения адресов областей памяти(указатели).

Указатели бывают трех типов:

1. Указатель на объект.

тип\*имя. (тип любой кроме ссылки и битового поля)(\*относится к имени)

Указатель на объект содержит адрес области памяти в которой хранятся данные определенного типа.

1. Указатель на функцию.

Тип(\*имя)(типы параметров)

Указатель на функцию хранит адрес в сегменте кода в которой находится исполняемый код функции.

Нужен для того чтобы передавать функцию в качестве параметра в другую функцию и косвенного вызова функции.(не через имя а через адрес)

1. Указатель на void.

Используется когда не определен тип объекта адрес которого хранится в указателе.(когда в разные моменты времени указатель хранит данные разного типа)

Указателю на void может быть присвоен указатель любого другого типа, но перед выполнением действий над областью памяти на которую указывает указатель необходимо явно привести его тип.

Указатель может быть переменной и константой а также указывать на переменную и константу.

Указатель на целую константу: const int\*a=&ci

Указатель константа на целое: int \*const a=&I;

Указатель константа на константу : const int\*const a=&ci;

44. Инициализация указателей.

Всегда следует стремиться к инициализации указателей. Существует 6 способов проинициализировать указатель.

1. С помощью оператора получения адреса int b=5, int \*a=&b; или int \*a(&b);
2. C помощью уже проинициализированного указателя int \*a=b;
3. С помощью присвоения адреса int\*a=(\*char) 00B80000
4. С помощью имени массива или функции(тк они трактуются как адрес) int a[20], int\*b=a;
5. Присвоение пустого значения int\*a=nullptr(NULL);
6. Выделение для указателя динамической памяти и присваивание ее адреса.

45. Операции над указателями.

1. Разыменовывание.
2. Присваивание.
3. Арифметические операции(инкремент декремент, сложение с константами, вычитание)
4. Сравнение.
5. Приведения типов.

Разыменовывание – это операция получения значения объекта, адрес которого хранит указатель.

Если указатель указывает на переменную то значение можно поменять.

Int b=5;

Int \*p(&b); выведет 6

(\*p)++;

Арифметические операции применимы только к указателям одного типа.

Инкремент увеличивает значение адреса на размер типа указателя.

Также работают и с остальными константами. Складывать указатели нельзя.

Вычитание работает и с константами и с двумя указателями. При вычитании из адреса отнимается адрес другого указателя и делится на размер типа данных.

int a=123, b=456, c=789

int \*pa=&a; \*pb=&b, \*pc=&c;

cout<<pa-pb; -1(-8)

cout<<pa-pc; -2(-16)

Приведение типов.

На одну и ту же область памяти могут ссылаться указатели разного типа и при применении к ним операции разыменовывания результат будет разный.

Int a=123;

Int\*i=&a;

Char\*c=(char\*)&a;

Float\*f=(float\*)&a;

Cout<<\*I; 123

Cout<<\*c; E;

///

При использовании в выражении указателей разных типов необходимо провести явное преобразование

46. Понятие ссылки. Правила работы с ссылками.

Ссылка -- это синоним к объекту который указан при инициализации ссылки.

Тип&имя ссылки=имя объекта.

Правила работы с ссылками.

1. Переменная ссылка должна быть явно проинициализирована если только она не является параметром функции, не определяется как extern, и не ссылается на поле класса.
2. После инициализции ссылке не может быть присвоено другое значение.
3. Операция над ссылкой меняет и объект на который она ссылается.
4. Не существует ссылок на ссылки, указателей ссылок, массивов ссылок.
5. Ссылка не занимает доп пространства в памяти.

47. Динамические переменные.

Динамические перемененые – участки динамической памяти.

Динамическая память – это память выделяемая для работы программы за вычетом сегманта памяти и стэка в котором хранятся все локальные переменные подпрограмм и тело программы.

Доступ к динамическим переменным производится с помощью указателей.

Для работы с динамической памятью есть два способа.

1. Унаследованный из С, с использованием семейства функций malloc.
2. С помощью операций new, delete.

Создание динамической переменной

Int \*a=new int[инициализатор];

Удаление с помощью delete \_переменная\_

48. Указатели и массивы.

Имя массива трактуется как его адрес.

Имя массива==&имя массива==&имя массива[0].

К нему применимы все свойства адресной арифметики.

Int a[3];

a[0]=\*a=\*(a+0)

a[1]=\*(a+1)

//

a[i]=\*(a+i);

int a[]={1,2,3,4,5}

int size=sizeof(a)/sizeof(a[0]) –колво элементов в массиве.

49. Одномерные динамические массивы.

Если до начала работы не известно сколько будет элементов в массиве рекомендуется использовать динамические массивы. Адрес для них выделяется с помощью операции new, адрес начала массива будет хранится в указателе.

Int\*a=new int[n];

Проинициализировать динамический массив нельзя. Если в какой то момент времени массив перестает быть нужным используется операция удаления delete[]a.

Замечание.

Область действия и время жизни указателя подчиняется обычным правилам. Переменная при выходе из блока теряется. Если это был указатель с адресом динамической памяти то она перестает быть доступной и не освобождается для дальнйшей работы. Это называется утечкой памяти.(ошибка)

50.