**19.04.2024**

При выносе шаблонных функций в отдельные файлы, на место вызова должен быть подключен не только прототип функций но и ее реализация. В противном случае возникает ошибка на этапе компоновки LNK2019: Unresolved external symbol. То есть если функция не шаблонная, ее реализация всегда подключается на место вызова.

Шаблонную функцию так же, как и любую другую функцию можно перегрузить.

Шаблонная функция обеспечивает идентичное поведение над всеми типами данных, но если над каким то определенным типом данных нужно обеспечить особое поведение, то шаблонная функция перегружается для этого типа данных.

N= a^n = 2^32

1 byte = 8 bit(binary digit )

**Указатели**

Часто возникает необходимость вместо значения переменной узнать адрес этой переменной. Для работы с адресами в языке C++ есть указатели. Указатель(pointer)- переменная, которая содержит адрес другой переменной. Адрес это число длиной 4 байта означающее порядковый номер байта в оперативной памяти. Адреса всегда записываются в 16ричной системе (Hexadecimal).

Для процессора память представляет собой массив байт. Как и в любом другом массиве каждый элемент имеет свой порядковый номер(индекс). Этот номер является адресом байта в оперативной памяти

Один 16ричный разряд вбирает в себя 4 двоичных разряда 16^1=2^4. За счет этого благодаря 16ричной СС запись двоичного числа сокращается в 4 раза и один байт всегда можно записать двухразрядным 16ричным числом

Объявление указателей

Int a = 2;

Int\* pa = &a;

Для понимания указателей нужно достаточно понимать два оператора: & и \*

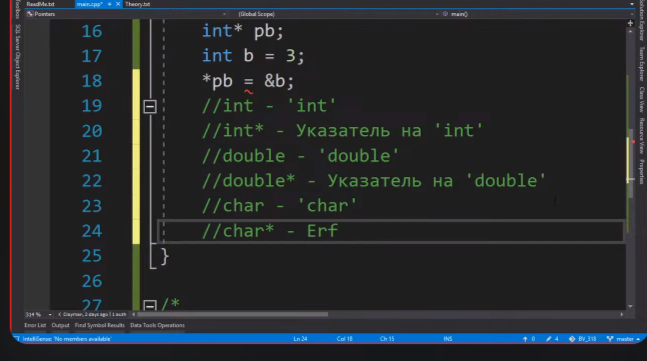
& - оператор взятия адреса(address of operator)унарный оператор который возвращает адрес своего операнда . У этого оператора есть только префиксная форма записи. cout<<&a Взятие адреса переменной a прямо при выводе

Cout<< pa вывод адреса переменной a хранящегося в указателе pa

Cout<< \*pa вывод значения переменной a

\* - оператор разыменования (Dereference operator)- унарный оператор, который возвращает значение по адресу, у этого оператора есть только префиксная форма записи. только при объявлении указателя \* показывает что объявляемая переменная- указатель. Во всех остальных случаях \* разыменовывает указатель и возвращает значение по адресу

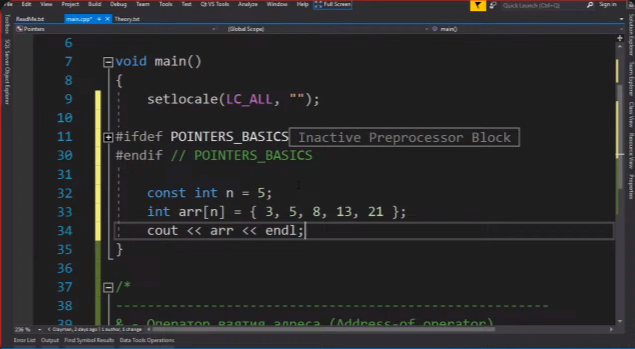
Cout << \*pa разыменование указателя pa и вывод значения переменной a



Уточнить алгоритм работы #ifdef

Указатели и массивы

Имя массива является указателем на массив, поскольку содержит адрес нулевого элемента массива. Это легко проверить следующим образом:

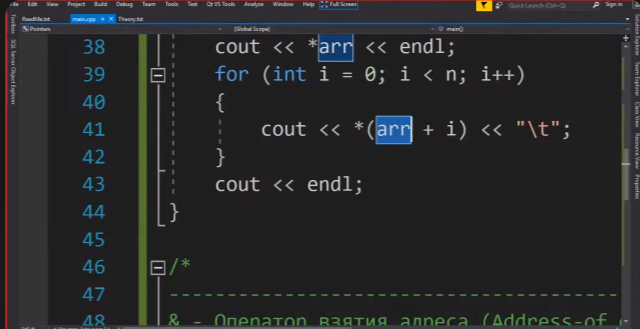


**Арифметика указателей**

К указателям применимы следующие арифметические операции

+,-,++,--

Результат операции зависит от размера целевого типа. Если к указателю на char прибавить 1 адрес изменится на один байт, на short - 2байта, int 4 байта,double 8байт и и .тд.



**Передача параметров в функцию**

Всего существует три способа передать параметры в функцию

1. По значению(by value);
2. По указателю(by pointer);
3. По ссылке(by reference);

Всё время до этого мы передавали параметры в функцию по значению, при этом значения передаваемых переменных копируются в функцию и если функция их меняет переданные в нее переменные, то эти изменения отображаются только на копии переменных, а оригиналы переданных переменных остаются неизменными

Для того что б функция имела возможность изменять значения переданных в неё переменных их нужно передать по указателю либо по ссылке

Передача параметров по указателю

Для того что бы передать параметры в функцию по указателю, принимаемые параметры функции должны быть указателями. Эти указатели нужно разыменовывать в теле функции. Именно разыменование и открывает доступ к переданным переменным, кроме того при вызове такой функции нужно взять адреса передаваемых в неё переменных. И тогда при вызове функции в нее скопируются не значения, а адреса передаваемых переменных.

**24.04.2024**

Система контроля версий - найти в плэйлисте.

**Передача параметров по ссылке.**

Ссылка (References ) – это переменная которая содержит адрес другой переменной. В отличие от указателя ссылку не нужно разыменовывать, компилятор сам это делает, кроме того не нужно брать адрес переменной, которой мы хотим сохранить в ссылке компилятор и это сам делает. Но компилятор скрывает от программиста адрес в хранящейся ссылке. Его невозможно узнать и тем более изменить. Именно поэтому ссылку можно проинициализировать только при объявлении.

Объявление ссылок.

Ссылка как константный указатель. Ссылка – это упрощенный вариант указателя. Ссылки появились в языке C++,т е в языке C ссылок не было. Ссылки, как правило, используются для передачи параметров функции. Для того чтобы передать параметры в функцию по ссылке принимаемые параметры функции должны быть с ссылками. В остальном передача параметров по ссылке синтаксически ничем не отличается от передачи параметров по значению, но имеет такой же эффект как передача параметров по указателю.

Заключение.

Важно понимать, что в функцию в любом случае что - то копируется - либо значение либо адрес переменной.

**Динамические массивы**

Динамическим называется массив, размер которого может быть задан переменным значением на этапе выполнения программы. Предварительно это переменное значение может быть вычислено или введено пользователем с клавиатуры. Для того, чтобы объявить динамический массив нужно объявить указатель и выделить память оператором new. Следующим образом int n = 5; int\* arr = new int[n];

Оператор new выделяет непрерывный блок памяти для n элементов заданного типа, в данном случае int(будет выделено 20байт).

Обращение к элементам динамического массива.

К элементам динамического массива можно обращаться так же, как и к элементам статического массива – через арифметику указателей и оператор разыменования или же через оператор индексирования, т е через квадратные скобки.

Передача динамического массива в функцию.

Динамический массив передается в функцию точно так же как и статический. Какой бы массив мы не передавали в функцию, в неё скопируется указатель на массив.

Удаление динамического массива.

После того как массив больше не нужен, его обязательно нужно удалить из памяти. Дело в том, что оператор new запрашивает память у операционной системы на временное пользование и после того, как эта память больше не нужна, ее обязательно нужно вернуть операционной системе. Для этого используется оператор delete[] arr;

Оператор delete должен быть вызван столько раз, сколько был вызван оператор new!

Если только выделять память и не удалять ее, рано или поздно вся свободная память в операционной системе закончится, что приведет к аварийному завершению нашей программы, а иногда и к полному зависанию системы.

**26.04.2024**

**Ошибка на этапе выполнения Debug assertion failed возникает**

1. Оператором delete удаляется область стега (область статической памяти)
2. Оператор delete второй раз обрабатывает по одному и тому же адресу

Вся память в пределах функций является статической и называется stack, динамическая память выделенная оператором new называется heap

**29.04.2024**

Stack- это модель памяти, из которой последний записанный элемент считывается первым. Stack обеспечивает всего две операции push(вставить) и pop(вытащить)

Push- помещает элемент на вершину стэка

Pop- снимает элемент с вершины стэкая

Для того чтобы прочитать дно стэка из него нужно выгрузить все эл-ты

**15.05.2024г**

**ПОВТОРНОЕ ИСПОЛЬЗОВАНИЕ КОДА**