Белорусский государственный технологический университет

Факультет информационных технологий

Кафедра программной инженерии

 Реферат

По дисциплине «Основы алгоритмизации и программирования»

На тему «Отличие NULL, nullptr, nullptr\_t»

Выполнила:

Студент(ка) 1 курса 7 группы

Подшиваленко Диана Игоревна

Проверил:

Белодед Николай Иванович

2024, Минск

**Содержание**

1. Предисловие … 3
2. Что такое NULL … 4
3. Что такое nullptr… 4
4. Что такое nullptr\_t …4
5. Основные отличия NULL и nullptr … 5
6. Примеры … 6
7. Вывод … 8

**Предисловие**

В данном реферате будут рассмотрены основные отличия NULL, nullptr, nullptr\_t а также будут приведены примеры их использования.

**1.** **Что такое NULL**

NULL — это термин, который чаще всего используется в контексте языков программирования, отличных от C++. В этих языках программа может использовать NULL для обозначения нулевого значения или отсутствия какого-либо значения. Однако, в отличие от nullptr, NULL обычно является чем-то сомнительным и часто приводит к ошибкам, так как его значения можно неявно преобразовывать, что может привести к непредсказуемым результатам.

NULL — особое значение, которое может присваиваться переменным, указателям и объектам. Оно означает, что переменная или объект не указывает на какой-либо адрес в памяти.

Когда переменной присваивается NULL, это означает, что она не содержит никакого значения. В отличие от других значений, NULL не является ни числом, ни строкой, ни объектом. NULL ­– это просто эквивалент записи 0 (ноль). Все дело в том, что NULL — это просто макрос.

**2.** **Что такое nullptr**

nullptr — это ключевое слово в языке программирования C++, которое используется для обозначения нулевого указателя. Его можно использовать при работе с указателями, чтобы указать на то, что указатель не ссылается ни на какой объект или функцию. В отличие от NULL, который может использоваться в других языках программирования, nullptr является типовобезопасным и может помочь избежать ошибок при работе с указателями.

nullptr — это более новая разработка, добавленная в С++ 11, и она уже работает аналогично тому, как это реализовано в Java или C#.

Это уже не просто макрос и не просто нолик. Это уже — отдельный тип данных. Мы его присваиваем и, на первый взгляд, разницы никакой нет. Однако для компилятора разница есть, он никогда не перепутает указатель nullptr с целочисленным типом данных. Ключевое слово nullptr является константой rvalue.

**3.** **Что такое nullptr\_t**

nullptr\_t является специальным типом данных в C++, который используется для представления значения nullptr. Он является типом, который может быть использован для перегрузки функций или операторов, связанных с нулевыми указателями.

nullptr\_t представляет тип значения nullptr и может быть использован для создания функций, которые принимают только нулевые указатели в качестве аргументов. Он также может быть использован для перегрузки операторов, чтобы обеспечить специальную обработку нулевых указателей.

nullptr\_t также может быть использован для перегрузки операторов, связанных с нулевыми указателями, таких как операторы сравнения (==, !=) или операторы присваивания (=).

**4. Основные отличия NULL и nullptr**

1) Тип данных:

* NULL – это макрос, который определен как ноль (0). Он не имеет своего собственного типа данных.
* nullptr – это ключевое слово, которое имеет тип данных nullptr\_t. Он был введен в C++11 и предназначен для представления нулевого указателя.

2) Типовая безопасность:

* NULL не является типовобезопасным, поскольку он может быть неявно преобразован в целые числа или другие типы указателей.
* nullptr обеспечивает типовую безопасность, поскольку он неявно преобразуется только в указатели и не может быть преобразован в другие типы.

3) Ясность и читаемость:

* nullptr более явное и понятное представление нулевого указателя, поскольку его название явно указывает на отсутствие указателя.
* NULL может быть менее ясным, особенно для новых программистов, которые могут не знать, что это макрос, раскрывающийся в ноль.

4) C++11 и выше:

* nullptr был добавлен только в стандарт C++11 и выше, поэтому его можно использовать только в этих версиях языка.
* NULL был доступен в более ранних версиях C++ и продолжает поддерживаться для совместимости со старым кодом.

**5.** **Примеры**

#include <iostream>

#include <string>

#include <cmath>

#include <Windows.h>

using namespace std;

void printstring(char\* a) {

if (a == NULL) {

cout << "Строка пустая\n";

}

else {

cout << a << '\n';

}

}

int main() {

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

char\* emptystring = NULL;

char fullstring[] = "Hello\n";

printstring(emptystring);

printstring(fullstring);

}

Результаты выполнения:



#include <iostream>

#include <string>

#include <cmath>

#include <Windows.h>

using namespace std;

void printnum(int \*num) {

if (num == nullptr) {

cout << "Этот указатель пуст\n";

}

else {

cout << \*num << '\n';

}

}

int main() {

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

int\* numptr = nullptr;

printnum(numptr);

int num = 13;

numptr = &num;

printnum(numptr);

}

Результаты выполнения:



#include <iostream>

#include <string>

#include <cmath>

#include <Windows.h>

using namespace std;

void nullptrcheck(nullptr\_t ptr)

{

cout << "Пустой указатель\n";

}

int main() {

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

nullptrcheck(nullptr);

return 0;

}

Результат выполнения:



**Выводы**

1) NULL – это макрос, который был использован в более ранних версиях C++ для представления нулевого указателя. Он обычно определяется как 0. Однако, использование NULL может привести к неоднозначности, так как он может быть определён по-разному в разных компиляторах. Поэтому, рекомендуется использовать nullptr вместо NULL.

2) nullptr – это ключевое слово, введенное в стандарте C++11, которое представляет нулевое значение указателя. Оно имеет свой собственный тип nullptr\_t, который может быть использован для перегрузки функций или шаблонов. Использование nullptr является предпочтительным способом указания нулевого указателя в современном C++.

3) nullptr\_t – это тип значения nullptr. Он используется для перегрузки функций и шаблонов, чтобы различать случаи, когда аргумент равен nullptr. В большинстве случаев, вам не понадобится явно использовать nullptr\_t, так как компилятор будет автоматически преобразовывать nullptr в nullptr\_t, когда это необходимо.