**Теория. Указатели. 12.2.2021 г.**

Объявление указателя.

Синтаксически язык C++ принимает объявление указателя, когда звёздочка находится рядом с типом данных, с идентификатором или даже посередине. Обратите внимание, эта звёздочка не является оператором разыменования. Это всего лишь часть синтаксиса объявления указателя.

|  |  |
| --- | --- |
| 2  3  4  5  6  7 | int \*iPtr; // указатель на значение типа int  double \*dPtr; // указатель на значение типа double    int\* iPtr3; // корректный синтаксис (допустимый, но не желательный)  int \* iPtr4; // корректный синтаксис (не делайте так)    int \*iPtr5, \*iPtr6; // объявляем два указателя для переменных типа int |

Операции адреса, разыменования.

**Присваивание**

Указателю можно присвоить либо адрес объекта того же типа, либо значение другого указателя.

Присвоение указателю адреса уже рассматривалось в прошлой теме. Для получения адреса объекта используется операция **&**:

**Нулевые указатели**

Нулевой указатель (null pointer) - это указатель, который не указывает ни на какой объект. Если мы не хотим, чтобы указатель указывал на какой-то конкретный адрес, то можно присвоить ему условное нулевое значение. Для создания нулевого указателя можно применять различные способы:

|  |  |
| --- | --- |
| 1  2  3 | int \*p1 = nullptr;  int \*p2 = NULL;  int \*p3 = 0; |

### Разыменование указателя

Операция разыменования указателя представляет выражение в виде \*имя\_указателя. Эта операция позволяет получить объект по адресу, который хранится в указателе.

### Адрес указателя

Указатель хранит адрес переменной, и по этому адресу мы можем получить значение этой переменной. Но кроме того, указатель, как и любая переменная, сам имеет адрес, по которому он располагается в памяти. Этот адрес можно получить также через операцию **&**:

### Операции сравнения

К указателям могут применяться операции сравнения **>**, **>=**, **<**, **<=**,**==**, **!=**. Операции сравнения применяются только к указателям одного типа и к значениям **NULL** и **nullptr**. Для сравнения используются номера адресов:

**Приведение типов**

Иногда требуется присвоить указателю одного типа значение указателя другого типа. В этом случае следует выполнить операцию приведения типов с помощью операции (тип\_указателя \*):

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | #include <iostream>    int main()  {      char c = 'N';      char \*pc = &c;      int \*pd = (int \*)pc;      void \*pv = (void\*)pc; |

Инициализация указателей. Операции с указателями.

Особенности операций инкремента и декремента (постфиксных, префиксных) с указателями.

Указатели и функции – передача аргументов функции по указателю, возвращение указателя из функции.

void func(int \*a)

{

\*a = 5;

}

int main()

{

int a;

func(&a);

cout << a;

}

Передача указателя функции по значению, ссылке. Нулевые указатели. Указатели на тип **void.** Указатели и спецификатор **const**.

const int value = 7; // value - это константа

int \*ptr = &value; // ошибка компиляции: невозможно конвертировать const int\* в int\*

\*ptr = 8; // изменяем значение value на 8

const int value = 7;

const int \*ptr = &value; // здесь всё ок: ptr - это неконстантный указатель, который указывает на "const int"

\*ptr = 8; // нельзя, мы не можем изменить константное значение

|  |  |
| --- | --- |
| 2  3 | int value = 7;  const int \*ptr = &value; // ptr указывает на "const int"  value = 8; // переменная value уже не константа, если к ней получают доступ через неконстантный идентификатор |

Но не следующее:

|  |  |  |  |
| --- | --- | --- | --- |
| 1  2  3 | int value = 7;  const int \*ptr = &value; // ptr указывает на "const int"  \*ptr = 8; // ptr обрабатывает value как константу, поэтому изменение значения переменной value через ptr не допускается Константные указатели Мы также можем сделать указатель константным. **Константный указатель**— это указатель, значение которого не может быть изменено после инициализации. Для объявления константного указателя используется **ключевое слово const между звёздочкой и именем указателя**   |  |  | | --- | --- | | 1  2 | int value = 7;  int \*const ptr = &value; | |

**Задачи.**

Сделать надо **одну** задачу (**или** 1-ую, или 2-ую)**!!!**

Динамический массив не использовать!

1. Составить функцию, которая для двух целых чисел, переданных функции по указателю, определяет число с максимальной суммой цифр и возвращает его адрес в **main**().

В **main**() ввести два целых числа и вывести на экран число с максимальной суммой цифр.

1. **\***Составить функцию, которая для двух целых чисел, переданных функции по указателю, определяет число, у которого количество максимальных цифр больше, и возвращает его адрес в **main**(). Поиск количества максимальных цифр в числе следует выполнить за один просмотр числа.

В **main**():

* ввести **массив** из **n** целых чисел;
* для каждой пары элементов массива определить элемент, у которого количество максимальных цифр больше, и вывести его **на экран**;
* дополнительно – можно в памяти получить массив результатов.

Например:

**исходный массив** – 3213 33213 543 5543 65919 6789

**результат –**  33213 5543 65919