|  |  |
| --- | --- |
| ДИСЦИПЛИНА | Технологии индустриального программирования |
| ИНСТИТУТ | ИПТИП |
| КАФЕДРА | Индустриального программирования |
| ВИД УЧЕБНОГО МАТЕРИАЛА | Методические указания к практическим занятиям по дисциплине |
| ПРЕПОДАВАТЕЛЬ | Александров Алексей Сергеевич |
| СЕМЕСТР | 3 семестр, 2024/2025 уч. год |

# Практическое занятие №11

## Адреса и указатели

**Указатели**

Указатель — это переменная, в которой хранится адрес памяти объекта. Объявление указателя выполняется следующим образом:

тип\_данных\* название\_указателя;

Например:

int \*p;

Указатель может хранить адрес переменной или быть пустым (NULL).

Задать указателю адрес переменной, на которую он ссылается, можно с помощью оператора & указываемой переменной:

int a = 10;

int \*p = &a; // указатель p хранит адрес переменной a

Если попытаться вывести значение переменной p на экран, то будет выведено следующее:

int a = 10;

int \*p = &a;

std::cout << "pointer: " << p << std::endl;

На экран будет выведено следующее:

pointer: 0xe1f99ff7cc

Это адрес переменной a, т.к. указатель сам по себе не хранит значение, а ссылается на другую переменную.

Однако, если выполнить **разыменование** указателя, то будет выведено значение переменной a, на которую он ссылается:

std::cout << "pointer value: " << \*p << std::endl;

Будет выведено следующее:

pointer value: 10

Таким образом, если указатель передаётся в функцию, как параметр, он обладает теми же свойствами, что и обычный параметр:

void func(int \*p)

{

int \*a = new int(10);

p = a;

}

int val = 25; // значение

int \*p\_val = &val; // указатель на переменную val

func(p\_val);

std::cout << "pointer value: " << \*p\_val << std::endl;

В результате выполнения данного кода значение не изменится, и на экран выведется число 25, т.к. указатель передаётся как аргумент функции, а значит просто копируется, и все изменения внутри функции void func(int \*p) остаются тоже внутри неё.

Однако, указатель хранит в себе адрес переменной, а следовательно, несмотря на то что мы не можем изменить сам указатель, мы можем обратиться по адресу, который хранится внутри и изменить значение переменной, на которую этот указатель ссылается. Аналогично тому, что мы не можем изменить индекс элемента массива, но можем изменить значение, ­которое хранится под данным индексом.

Таким образом, сделав разыменование указателя, можно обратиться напрямую к переменной, на которую он ссылается:

void func(int \*p)

{

int \*a = new int(10);

**\*p = \*a; // делаем рызыменование**

}­

int val = 25; // значение

int \*p\_val = &val; // указатель на переменную val

func(p\_val);

std::cout << "pointer value: " << \*p\_val << std::endl;

std::cout << "pointer value: " << val << std::endl;

В результате выполнения данного кода на экран будет выведено значение 10 на обоих строках. Это происходит из-за того, что внутри функции func() было произведено разыменование указателя \*p, а следовательно работа происходила напрямую через адрес переменной val, в следствии чего все изменения происходили с этой переменной.