Балтийский государственный технический университет «ВОЕНМЕХ»

им. Д. Ф. Устинова

Кафедра О7

«Информационные системы и программная инженерия»

Практическое задание № 4

по дисциплине «Информационные технологии и программирование»

на тему «Указатели»

Выполнил:

Студент Карпинская Полина Анатольевна

Группа Е721б

Преподаватель: Удовиченко Андрей Сергеевич

Санкт-Петербург2023

Цель работы – познакомиться с адресацией памяти, научиться правильно использовать указатели различных типов.

Скрипт practical\_task\_4.c:

Ответы на контрольные вопросы:

1. Указатель - это переменная, которая содержит адрес ячейки памяти, где хранится значение другой переменной или объекта.

2. Объем памяти, занимаемый указателем, зависит от архитектуры компьютера. Обычно это 4 или 8 байт.

3. Значением переменной-указателя является адрес ячейки памяти, на которую она указывает.

4. Указатель можно проинициализировать при объявлении, присвоив ему адрес существующей переменной или объекта, или позже, с помощью оператора присваивания, после того, как он был объявлен.

5. NULL - это особое значение указателя, которое указывает на отсутствие валидного адреса. Он используется, чтобы показать, что указатель не ссылается на какой-либо объект.

6. Указатель на void - это указатель, который может указывать на объект любого типа данных. Он используется, когда тип объекта неизвестен или может измениться.

7. При работе с указателями можно выполнять операции присваивания, разыменования, арифметические операции (сложение, вычитание, инкремент, декремент) и сравнение.

8. Унарная операция & возвращает адрес переменной. Унарная операция \* разыменовывает указатель и возвращает значение, на которое он указывает.

9. Совместимость типов указателей означает, что указатель одного типа может быть присвоен переменной другого типа, если оба типа совместимы и не нарушают правил типизации языка.

10. Да, можно получить адрес указателя с помощью операции &.

11. Да, указателю можно присвоить его же адрес.

12. Указатель на void не имеет типа данных и неизвестно, какой размер имеет объект, на который он указывает. Поэтому операция разыменования не может быть применена к указателю на void.

13. Операции инкремента и декремента при применении к указателям изменяют адрес, на который указывает указатель. Инкремент увеличивает адрес на количество байт, соответствующее типу данных, на который указывает указатель. Декремент уменьшает адрес на такое же количество байт.

14. Результат операции вычитания, примененной к указателям одного типа, будет равен разнице их адресов в единицах размера типа данных.

15. Для вывода адреса на экран с помощью функции printf() необходимо использовать спецификатор типа %p.

16. (float \*) a - это приведение указателя a к типу указателя на float. (float) \* a - это разыменование указателя a и получение значения float из ячейки памяти, на которую указывает a.

17. \*а++ сначала разыменовывает указатель а и возвращает его значение, затем увеличивает адрес на количество байт, соответствующее типу, на который указывает а. (\*а)++ сначала разыменовывает указатель а, увеличивает значение, на которое он указывает, на 1, а затем присваивает его обратно ячейке памяти.

18. Указатель на начало массива можно описать с помощью следующего синтаксиса: int \*ptr;

19. Указатель на указатель можно описать с помощью следующего синтаксиса: int \*\*pptr;

20. Операция разыменования может повторно использоваться, когда необходимо получить доступ к значению, на которое указывает указатель. Например, при работе с динамической памятью, когда нужно получить значение переменной, на которую указывает указатель на эту переменную. Также операция разыменования может повторно использоваться, когда нужно применить другую операцию к значению, на которое указывает указатель, например, передать значение в другую функцию или изменить значение данной переменной.