

## Вопросы по лекции 7

Лектор: Артамонов Юрий Николаевич

Университет "Дубна"  
филиал Котельники

## Вопрос 1

Заполните пропуски в предложениях

- Вопрос: Указатель - это переменная, которая в качестве значения содержит \_\_\_\_\_ другой переменной.

## Вопрос 1

Заполните пропуски в предложениях

- Вопрос: Указатель - это переменная, которая в качестве значения содержит \_\_\_\_\_ другой переменной.
- Ответ: адрес

Заполните пропуски в предложениях

- Вопрос: Указатель - это переменная, которая в качестве значения содержит \_\_\_\_\_ другой переменной.
- Ответ: адрес
- Вопрос: Только три величины могут использоваться для инициализации указателя \_\_\_\_\_, \_\_\_\_\_ или \_\_\_\_\_.

Заполните пропуски в предложениях

- Вопрос: Указатель - это переменная, которая в качестве значения содержит \_\_\_\_\_ другой переменной.
- Ответ: адрес
- Вопрос: Только три величины могут использоваться для инициализации указателя \_\_\_\_\_, \_\_\_\_\_ или \_\_\_\_\_.
- Ответ: 0, NULL, адрес

# Вопрос 1

Заполните пропуски в предложениях

- Вопрос: Указатель - это переменная, которая в качестве значения содержит \_\_\_\_\_ другой переменной.
- Ответ: адрес
- Вопрос: Только три величины могут использоваться для инициализации указателя \_\_\_\_\_, \_\_\_\_\_ или \_\_\_\_\_.
- Ответ: 0, NULL, адрес
- Вопрос: Единственное число, которое может быть присвоено указателю - это \_\_\_\_\_.

# Вопрос 1

Заполните пропуски в предложениях

- Вопрос: Указатель - это переменная, которая в качестве значения содержит \_\_\_\_\_ другой переменной.
- Ответ: адрес
- Вопрос: Только три величины могут использоваться для инициализации указателя \_\_\_\_\_, \_\_\_\_\_ или \_\_\_\_\_.
- Ответ: 0, NULL, адрес
- Вопрос: Единственное число, которое может быть присвоено указателю - это \_\_\_\_\_.
- Ответ: нуль.

## Вопрос 2

Являются ли следующие утверждения верными? Если утверждение неверно, объясните, почему.

- Вопрос: операция взятия адреса & может применяться только к константам, выражениям и переменным, объявленным с модификатором register.



## Вопрос 2

Являются ли следующие утверждения верными? Если утверждение неверно, объясните, почему.

- Вопрос: операция взятия адреса & может применяться только к константам, выражениям и переменным, объявленным с модификатором register.
- Ответ: неверно, модификатор register рекомендует поместить соответствующий объект в регистр, а у него нет адреса в памяти.

## Вопрос 2

Являются ли следующие утверждения верными? Если утверждение неверно, объясните, почему.

- Вопрос: операция взятия адреса `&` может применяться только к константам, выражениям и переменным, объявленным с модификатором `register`.
- Ответ: неверно, модификатор `register` рекомендует поместить соответствующий объект в регистр, а у него нет адреса в памяти.
- Вопрос: указатель на `void` может быть разыменован.

## Вопрос 2

Являются ли следующие утверждения верными? Если утверждение неверно, объясните, почему.

- Вопрос: операция взятия адреса `&` может применяться только к константам, выражениям и переменным, объявленным с модификатором `register`.
- Ответ: неверно, модификатор `register` рекомендует поместить соответствующий объект в регистр, а у него нет адреса в памяти.
- Вопрос: указатель на `void` может быть разыменован.
- Ответ: неверно, указатель на `void` как раз нельзя разыменовывать, поскольку неизвестно, сколько последовательных байт памяти брать, чтобы представить объект.

## Вопрос 2

Являются ли следующие утверждения верными? Если утверждение неверно, объясните, почему.

- Вопрос: операция взятия адреса `&` может применяться только к константам, выражениям и переменным, объявленным с модификатором `register`.
- Ответ: неверно, модификатор `register` рекомендует поместить соответствующий объект в регистр, а у него нет адреса в памяти.
- Вопрос: указатель на `void` может быть разыменован.
- Ответ: неверно, указатель на `void` как раз нельзя разыменовывать, поскольку неизвестно, сколько последовательных байт памяти брать, чтобы представить объект.
- Вопрос: указатели на разные типы данных не могут быть присвоены друг другу без использования операции приведения типов.

## Вопрос 2

Являются ли следующие утверждения верными? Если утверждение неверно, объясните, почему.

- Вопрос: операция взятия адреса `&` может применяться только к константам, выражениям и переменным, объявленным с модификатором `register`.
- Ответ: неверно, модификатор `register` рекомендует поместить соответствующий объект в регистр, а у него нет адреса в памяти.
- Вопрос: указатель на `void` может быть разыменован.
- Ответ: неверно, указатель на `void` как раз нельзя разыменовывать, поскольку неизвестно, сколько последовательных байт памяти брать, чтобы представить объект.
- Вопрос: указатели на разные типы данных не могут быть присвоены друг другу без использования операции приведения типов.
- Ответ: неверно, указатель на `void` можно присвоить указателям других типов, и указателю на `void` можно присваивать значения указателей других типов.

Выполните задания.

- Вопрос: Объявите массив `numbers` типа `float` из 10 элементов и присвойте элементам значения 0.0, 1.1, ... 9.9. Используйте символическую константу `SIZE`, равную 10.

Выполните задания.

- Вопрос: Объявите массив `numbers` типа `float` из 10 элементов и присвойте элементам значения 0.0, 1.1, ... 9.9. Используйте символическую константу `SIZE`, равную 10.
- Ответ:

```
float numbers[SIZE] = {0.0, 1.1, 2.2, 3.3, 4.4, 5.5, 6.6,  
                        7.7, 8.8, 9.9};
```

Выполните задания.

- Вопрос: Объявите массив `numbers` типа `float` из 10 элементов и присвойте элементам значения 0.0, 1.1, ... 9.9. Используйте символическую константу `SIZE`, равную 10.

- Ответ:

```
float numbers[SIZE] = {0.0, 1.1, 2.2, 3.3, 4.4, 5.5, 6.6,
                        7.7, 8.8, 9.9};
```

- Вопрос: Объявите указатель `pPtr`, ссылающийся на тип `float`.



Выполните задания.

- Вопрос: Объявите массив `numbers` типа `float` из 10 элементов и присвойте элементам значения 0.0, 1.1, ... 9.9. Используйте символическую константу `SIZE`, равную 10.

- Ответ:

```
float numbers[SIZE] = {0.0, 1.1, 2.2, 3.3, 4.4, 5.5, 6.6,
                        7.7, 8.8, 9.9};
```

- Вопрос: Объявите указатель `nPtr`, ссылающийся на тип `float`.

- Ответ:

```
float *nPtr;
```

## Вопрос 3 (продолжение)

- Вопрос: выведите элементы массива `numbers`, используя нотацию имя массива/индекс. Используйте цикл `for` и переменную цикла `i`. Выведите каждый элемент с точностью одного знака после запятой.

## Вопрос 3 (продолжение)

- Вопрос: выведите элементы массива numbers, используя нотацию имя массива/индекс. Используйте цикл for и переменную цикла i. Выведите каждый элемент с точностью одного знака после запятой.
- Ответ:

```
int i;  
for (i = 0; i < SIZE; i++) printf("numbers[ %d]= % .1f", i, numbers[i]);
```

## Вопрос 3 (продолжение)

- Вопрос: выведите элементы массива `numbers`, используя нотацию имя массива/индекс. Используйте цикл `for` и переменную цикла `i`. Выведите каждый элемент с точностью одного знака после запятой.

- Ответ:

```
int i;  
for (i = 0; i < SIZE; i++) printf("numbers[ %d]= %0.1f", i, numbers[i]);
```

- Вопрос: выведите элементы массива `numbers` при помощи обращения к элементам массива по методу указатель/смещение, где в качестве указателя используйте имя массива.

## Вопрос 3 (продолжение)

- Вопрос: выведите элементы массива `numbers`, используя нотацию имя массива/индекс. Используйте цикл `for` и переменную цикла `i`. Выведите каждый элемент с точностью одного знака после запятой.

- Ответ:

```
int i;  
for (i = 0; i < SIZE; i++) printf("numbers[ %d]= % .1f", i, numbers[i]);
```

- Вопрос: выведите элементы массива `numbers` при помощи обращения к элементам массива по методу указатель/смещение, где в качестве указателя используйте имя массива.

- Ответ:

```
int i;  
for (i = 0; i < SIZE; i++) printf("numbers[ %d]= % .1f", i, *(numbers + i));
```

## Вопрос 3 (продолжение)

- Вопрос: выведите элементы массива `numbers`, используя индексацию указателя `nPtr`.

## Вопрос 3 (продолжение)

- Вопрос: выведите элементы массива numbers, используя индексацию указателя nPtr.
- Ответ:

```
int i;  
nPtr = a;  
for (i = 0; i < SIZE; i++) printf("numbers[ %d] = % .1f", i, nPtr[i]);
```

## Вопрос 3 (продолжение)

- Вопрос: выведите элементы массива numbers, используя индексацию указателя nPtr.

- Ответ:

```
int i;  
nPtr = a;  
for (i = 0; i < SIZE; i++) printf("numbers[ %d] = % .1f", i, nPtr[i]);
```

- Вопрос: выведите элементы массива numbers при помощи обращения к элементам массива по методу указатель/смещение, где в качестве указателя используйте nPtr.



## Вопрос 3 (продолжение)

- Вопрос: выведите элементы массива numbers, используя индексацию указателя nPtr.

- Ответ:

```
int i;  
nPtr = a;  
for (i = 0; i < SIZE; i++) printf("numbers[ %d] = % .1f", i, nPtr[i]);
```

- Вопрос: выведите элементы массива numbers при помощи обращения к элементам массива по методу указатель/смещение, где в качестве указателя используйте nPtr.

- Ответ:

```
int i;  
for (i = 0; i < SIZE; i++) printf("numbers[ %d] = % .1f", i, *(nPtr+i));  
;
```

## Вопрос 3 (продолжение)

- Вопрос: Сошлитесь на четвертый элемент массива, используя все четыре способа доступа к элементам массива: имя массива/индекс, имя массива/смещение, указатель/индекс с указателем `pPtr` и указатель/смещение с указателем `pPtr`.

## Вопрос 3 (продолжение)

- Вопрос: Сошлитесь на четвертый элемент массива, используя все четыре способа доступа к элементам массива: имя массива/индекс, имя массива/смещение, указатель/индекс с указателем nPtr и указатель/смещение с указателем nPtr.
- Ответ:

```
printf("% .1f", numbers[3]);  
printf("% .1f", *(numbers+3));  
printf("% .1f", nPtr[3]);  
printf("% .1f", *(nPtr+3));
```

## Вопрос 3 (продолжение)

- Вопрос: Сошлитесь на четвертый элемент массива, используя все четыре способа доступа к элементам массива: имя массива/индекс, имя массива/смещение, указатель/индекс с указателем nPtr и указатель/смещение с указателем nPtr.

- Ответ:

```
printf("% .1f", numbers[3]);  
printf("% .1f", *(numbers+3));  
printf("% .1f", nPtr[3]);  
printf("% .1f", *(nPtr+3));
```

- Вопрос: Если предположить, что nPtr указывает на начало массива numbers, то на какой адрес ссылается nPtr+8. Что за значение находится по указанному адресу (предполагаем, что числа с точкой обычной точности занимают 4 байта, и что начальный адрес массива равен 1002500).

## Вопрос 3 (продолжение)

- Вопрос: Сошлитесь на четвертый элемент массива, используя все четыре способа доступа к элементам массива: имя массива/индекс, имя массива/смещение, указатель/индекс с указателем nPtr и указатель/смещение с указателем nPtr.

- Ответ:

```
printf("% .1f", numbers[3]);  
printf("% .1f", *(numbers+3));  
printf("% .1f", nPtr[3]);  
printf("% .1f", *(nPtr+3));
```

- Вопрос: Если предположить, что nPtr указывает на начало массива numbers, то на какой адрес ссылается nPtr+8. Что за значение находится по указанному адресу (предполагаем, что числа с точкой обычной точности занимают 4 байта, и что начальный адрес массива равен 1002500).
- Ответ:  $1002500 + 8 * 4 = 1002532$ . По этому адресу располагается число 8.8.

## Вопрос 3 (продолжение)

- Вопрос: Предположим, что `nPtr` ссылается на `numbers[5]`, на какой адрес ссылается `nPtr-4`? Что за значение находится по указанному адресу?

## Вопрос 3 (продолжение)

- Вопрос: Предположим, что `nPtr` ссылается на `numbers[5]`, на какой адрес ссылается `nPtr-4`? Что за значение находится по указанному адресу?
- Ответ: Адрес `numbers[5]` - это  $1002500 + 5 * 4 = 1002520$ , соответственно `nPtr-4` ссылается на  $1002520 - 4 * 4 = 1002504$ . Там хранится число 1.1.

## Вопрос 4

Выполните каждую из следующих задач при помощи одиночного оператора. Считайте объявленными переменные с плавающей точкой `number1` и `number2`, причем `number1` присвоено значение 7.3.

- Вопрос: Объявите указатель `fPtr` на тип `float`.



## Вопрос 4

Выполните каждую из следующих задач при помощи одиночного оператора. Считайте объявленными переменные с плавающей точкой `number1` и `number2`, причем `number1` присвоено значение 7.3.

- Вопрос: Объявите указатель `fPtr` на тип `float`.
- Ответ: `float *fPtr;`

## Вопрос 4

Выполните каждую из следующих задач при помощи одиночного оператора. Считайте объявленными переменные с плавающей точкой `number1` и `number2`, причем `number1` присвоено значение 7.3.

- Вопрос: Объявите указатель `fPtr` на тип `float`.
- Ответ: `float *fPtr;`
- Вопрос: Присвойте адрес переменной `number1` указателю `fPtr`.

## Вопрос 4

Выполните каждую из следующих задач при помощи одиночного оператора. Считайте объявленными переменные с плавающей точкой `number1` и `number2`, причем `number1` присвоено значение 7.3.

- Вопрос: Объявите указатель `fPtr` на тип `float`.
- Ответ: `float *fPtr;`
- Вопрос: Присвойте адрес переменной `number1` указателю `fPtr`.
- Ответ:

```
fPtr = &number1;
```

## Вопрос 4

Выполните каждую из следующих задач при помощи одиночного оператора. Считайте объявленными переменные с плавающей точкой `number1` и `number2`, причем `number1` присвоено значение 7.3.

- Вопрос: Объявите указатель `fPtr` на тип `float`.
- Ответ: `float *fPtr;`
- Вопрос: Присвойте адрес переменной `number1` указателю `fPtr`.
- Ответ:

```
fPtr = &number1;
```

- Вопрос: Выведите значение величины, на которую ссылается `fPtr`.

## Вопрос 4

Выполните каждую из следующих задач при помощи одиночного оператора. Считайте объявленными переменные с плавающей точкой `number1` и `number2`, причем `number1` присвоено значение 7.3.

- Вопрос: Объявите указатель `fPtr` на тип `float`.
- Ответ: `float *fPtr;`
- Вопрос: Присвойте адрес переменной `number1` указателю `fPtr`.
- Ответ:

```
fPtr = &number1;
```

- Вопрос: Выведите значение величины, на которую ссылается `fPtr`.
- Ответ:

```
printf(" %f", *fPtr);
```

## Вопрос 4 (продолжение)

- Вопрос: Присвойте значение величины, на которую указывает fPtr, переменной number2

## Вопрос 4 (продолжение)

- Вопрос: Присвойте значение величины, на которую указывает fPtr, переменной number2
- Ответ: `number2 = *fPtr;`

## Вопрос 4 (продолжение)

- Вопрос: Присвойте значение величины, на которую указывает fPtr, переменной number2
- Ответ: `number2 = *fPtr;`
- Вопрос: Выведите адрес number1.



## Вопрос 4 (продолжение)

- Вопрос: Присвойте значение величины, на которую указывает fPtr, переменной number2
- Ответ: `number2 = *fPtr;`
- Вопрос: Выведите адрес number1.
- Ответ:

```
printf(" %p%p", fPtr, &number1);
```

Выполните следующие задания.

- Напишите заголовок функции `exchange`, которая имеет два параметра указателя на переменные с плавающей точкой `x` и `y` и не возвращает значения.

Выполните следующие задания.

- Напишите заголовок функции `exchange`, которая имеет два параметра указателя на переменные с плавающей точкой `x` и `y` и не возвращает значения.
- Ответ: `void exchange(float *x, float *y)`

Выполните следующие задания.

- Напишите заголовок функции `exchange`, которая имеет два параметра указателя на переменные с плавающей точкой `x` и `y` и не возвращает значения.
- Ответ: `void exchange(float *x, float *y)`
- Напишите прототип функции из предыдущего задания

Выполните следующие задания.

- Напишите заголовок функции `exchange`, которая имеет два параметра указателя на переменные с плавающей точкой `x` и `y` и не возвращает значения.
- Ответ: `void exchange(float *x, float *y)`
- Напишите прототип функции из предыдущего задания
- Ответ:

```
void exchange(float *, float *);
```

Выполните следующие задания.

- Напишите заголовок функции `evaluate`, которая возвращает целое число и имеет в качестве параметров целое число `x` и указатель на функцию `poly`. Функция `poly`, в свою очередь, имеет целочисленный параметр и возвращает целое число.

## Вопрос 5 (продолжение)

Выполните следующие задания.

- Напишите заголовок функции `evaluate`, которая возвращает целое число и имеет в качестве параметров целое число `x` и указатель на функцию `poly`. Функция `poly`, в свою очередь, имеет целочисленный параметр и возвращает целое число.
- Ответ:

```
int evaluate(int x, int (*poly)(int))
```

## Вопрос 5 (продолжение)

Выполните следующие задания.

- Напишите заголовок функции `evaluate`, которая возвращает целое число и имеет в качестве параметров целое число `x` и указатель на функцию `poly`. Функция `poly`, в свою очередь, имеет целочисленный параметр и возвращает целое число.

- Ответ:

```
int evaluate(int x, int (*poly)(int))
```

- Напишите прототип функции из предыдущего задания



## Вопрос 5 (продолжение)

Выполните следующие задания.

- Напишите заголовок функции `evaluate`, которая возвращает целое число и имеет в качестве параметров целое число `x` и указатель на функцию `poly`. Функция `poly`, в свою очередь, имеет целочисленный параметр и возвращает целое число.

- Ответ:

```
int evaluate(int x, int (*poly)(int))
```

- Напишите прототип функции из предыдущего задания

- Ответ:

```
int evaluate(int x, int (*)(int));
```

## Вопрос 6

Найдите ошибку в фрагменте программы, предполагая, что изначально имеем:

```
int *zPtr;  
int *aPtr = NULL;  
void *sPtr = NULL;  
int number, i;  
int z[5] = {1,2,3,4,5};  
  
sPtr = z;
```

• `++zPtr;`

## Вопрос 6

Найдите ошибку в фрагменте программы, предполагая, что изначально имеем:

```
int *zPtr;  
int *aPtr = NULL;  
void *sPtr = NULL;  
int number, i;  
int z[5] = {1,2,3,4,5};  
  
sPtr = z;
```



```
++zPtr;
```

- Ответ: Указатель zPtr не инициализирован.

## Вопрос 6

Найдите ошибку в фрагменте программы, предполагая, что изначально имеем:

```
int *zPtr;  
int *aPtr = NULL;  
void *sPtr = NULL;  
int number, i;  
int z[5] = {1,2,3,4,5};  
  
sPtr = z;
```



```
++zPtr;
```



● Ответ: Указатель zPtr не инициализирован.



```
number = zPtr;
```

## Вопрос 6

Найдите ошибку в фрагменте программы, предполагая, что изначально имеем:

```
int *zPtr;  
int *aPtr = NULL;  
void *sPtr = NULL;  
int number, i;  
int z[5] = {1,2,3,4,5};  
  
sPtr = z;
```



```
++zPtr;
```



Ответ: Указатель zPtr не инициализирован.




```
number = zPtr;
```



Ответ: Указатель не был разыменован. Правильно `number = *zPtr;`

## Вопрос 6 (продолжение)



```
number = *zPtr[2];
```

## Вопрос 6 (продолжение)

- ```
number = *zPtr[2];
```

- Ответ: `zPtr[2]` это не указатель, его нельзя разыменовывать.

## Вопрос 6 (продолжение)

- ```
number = *zPtr[2];
```

- Ответ: zPtr[2] это не указатель, его нельзя разыменовать.

- ```
for (i=0; i<=5; i++)  
    printf(" %f", sPtr[i]);
```



## Вопрос 6 (продолжение)

- ```
number = *zPtr[2];
```

- Ответ: zPtr[2] это не указатель, его нельзя разыменовывать.

- ```
for (i=0; i<=5; i++)  
    printf(" %f", sPtr[i]);
```

- Ответ: Выход за пределы массива, массив из целых чисел.

## Вопрос 6 (продолжение)

```
number = *zPtr[2];
```

- Ответ: zPtr[2] это не указатель, его нельзя разыменовывать.

```
for (i=0; i<=5; i++)  
    printf(" %f", sPtr[i]);
```

- Ответ: Выход за пределы массива, массив из целых чисел.

```
number = *sPtr;
```

## Вопрос 6 (продолжение)

- ```
number = *zPtr[2];
```

- Ответ: zPtr[2] это не указатель, его нельзя разыменовывать.

- ```
for (i=0; i<=5; i++)  
    printf(" %f", sPtr[i]);
```

- Ответ: Выход за пределы массива, массив из целых чисел.

- ```
number = *sPtr;
```

- Ответ: Попытка разыменовывать указатель типа void

## Вопрос 6 (продолжение)

```
number = *zPtr[2];
```

- Ответ: zPtr[2] это не указатель, его нельзя разыменовывать.

```
for (i=0; i<=5; i++)  
    printf(" %f", sPtr[i]);
```

- Ответ: Выход за пределы массива, массив из целых чисел.

```
number = *sPtr;
```

- Ответ: Попытка разыменовывать указатель типа void

```
++z;
```

## Вопрос 6 (продолжение)

- ```
number = *zPtr[2];
```

- Ответ: zPtr[2] это не указатель, его нельзя разыменовывать.

```
for (i=0; i<=5; i++)  
    printf(" %f", sPtr[i]);
```

- Ответ: Выход за пределы массива, массив из целых чисел.

```
number = *sPtr;
```

- Ответ: Попытка разыменовывать указатель типа void

```
++z;
```

- Ответ: Массив - это указатель, который нельзя изменять.