Семинар #6: Указатели. Домашнее задание.

Задача 1. Создание указателей

Решения всех подзадач этой части – одна строка. Результат выполнения задания – `.txt` файл, который содержит все эти строки.

1. В следующей программе создаётся переменная `a` типа `int`:

   ```
   int main()
   {
       int a = 1234;
       // Тут нужно написать 1 строку кода
   }
   ```

   Создайте указатель `p` и инициализируйте его адресом переменной `a`.

2. В следующей программе создаётся переменная `a` типа `double`:

   ```
   int main()
   {
       double a = 12.34;
       // Тут нужно написать 1 строку кода
   }
   ```

   Создайте указатель `p` и инициализируйте его адресом переменной `a`.

3. В следующей программе создаётся переменная `a` типа `char`:

   ```
   int main()
   {
       char a = ')';
       // Тут нужно написать 1 строку кода
   }
   ```

   Создайте указатель `p` и инициализируйте его адресом переменной `a`.

4. В следующей программе создаётся массив `array` из элементов типа `int`:

   ```
   int main()
   {
       int array[5] = {10, 20, 30, 40, 50};
       // Тут нужно написать 1 строку кода
   }
   ```

   Создайте указатель `p` и сделайте так, чтобы он указывал на первый элемент массива (индекс `0`).

5. В следующей программе создаётся строка – массив `str` из элементов типа `char`:

   ```
   int main()
   {
       char str[20] = "Sapere Aude";
       // Тут нужно написать 1 строку кода
   }
   ```

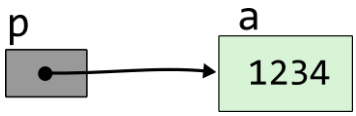
   Создайте указатель `p` и сделайте так, чтобы он указывал на символ `'A'` из строки `str`.

Задача 2. Использование указателей

Решения всех подзадач этой части – одна строка. Результат выполнения задания – `.txt` файл, который содержит все эти строки.

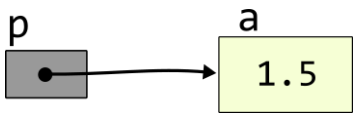
1. В следующей программе была создана переменная `a` и указатель на неё `p`. Удвойте значение переменной `a`, используя только указатель `p`. Нужно использовать указатель `p`, саму переменную `a` использовать нельзя.

```c
#include <stdio.h>
int main()
{
    int a = 1234;
    int* p = &a;
    // Тут нужно написать 1 строку кода

    printf("%i\n", a);
}
```



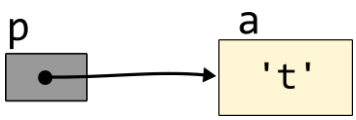
2. В следующей программе была создана переменная `a` типа `float` и указатель на неё `p`. Возведите значение переменной `a` в квадрат, используя только указатель `p`. Нужно использовать только указатель `p`, саму переменную `a` использовать нельзя.

```c
#include <stdio.h>
int main()
{
    float a = 1.5;
    float* p = &a;
    // Тут нужно написать 1 строку кода

    printf("%f\n", a);
}
```



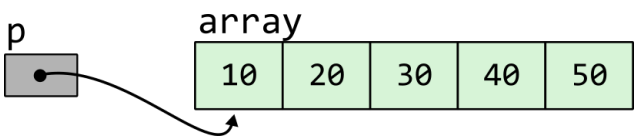
3. В следующей программе была создана переменная `a` типа `char` и указатель на неё `p`. Переведите символ, хранящийся в переменной `a` в верхний регистр, используя только указатель `p`. Нужно использовать только указатель `p`, саму переменную `a` использовать нельзя.

```c
#include <stdio.h>
int main()
{
    char a = 't';
    char* p = &a;
    // Тут нужно написать 1 строку кода

    printf("%c\n", a);
}
```



4. В следующей программе был создан массив `array` переменных типа `int` и указатель `p` на первый элемент массива.

```c
#include <stdio.h>
int main()
{
    int array[5] = {10, 20, 30, 40, 50};
    int* p = &array[0];

    for (int i = 0; i < 5; ++i)
        printf("%i ", array[i]);
}
```



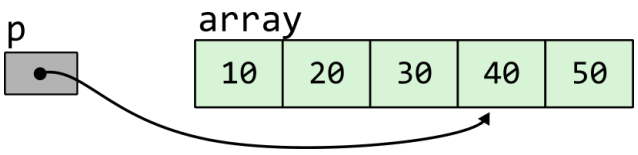
   (a) Добавьте `1` к первому элементу массива (`array[0]`), используя только указатель `p`. Нужно использовать только указатель `p`, сам массив `array` использовать нельзя. Решение – 1 строка.

   (b) Добавьте `1` к четвёртому элементу массива (`array[3]`), используя только указатель `p`. Нужно использовать только указатель `p`, сам массив `array` использовать нельзя. Менять `p` тоже нельзя. Решение – 1 строка.

   (c) Добавьте `1` ко всем элементам массива. Нужно использовать только указатель `p`, сам массив `array` использовать нельзя. Решение – 1 цикл.

5. В следующей программе был создан массив `array` переменных типа `int` и указатель `p` на четвёртый элемент массива (`array[3]`).

```c
#include <stdio.h>
int main()
{
    int array[5] = {10, 20, 30, 40, 50};
    int* p = &array[3];

    for (int i = 0; i < 5; ++i)
        printf("%i ", array[i]);
}
```



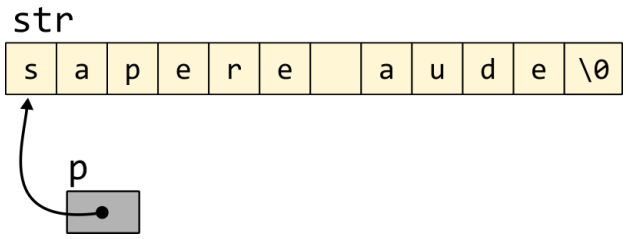
   (a) Добавьте `1` к первому элементу массива (`array[0]`), используя только указатель `p`. Нужно использовать только указатель `p`, сам массив `array` использовать нельзя. Менять `p` тоже нельзя. Решение – 1 строка.

   (b) Добавьте `1` к пятому элементу массива (`array[4]`), используя только указатель `p`. Нужно использовать только указатель `p`, сам массив `array` использовать нельзя. Менять `p` тоже нельзя. Решение – 1 строка.

   (c) Добавьте `1` ко всем элементам массива. Нужно использовать только указатель `p`, сам массив `array` использовать нельзя. Решение – 1 цикл.

6. В следующей программе была создана строка `str` и указатель `p` на первый символ строки.

```c
#include <stdio.h>
int main()
{
    char str[] = "sapere aude";
    char* p = &str[0];

    printf("%s\n", str);
}
```



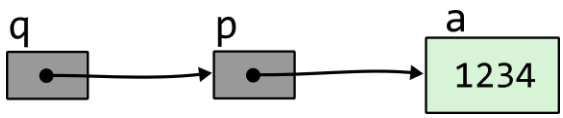
(a) Переведите в верхний регистр первую букву строки, используя только указатель `p`. Нужно использовать только указатель `p`, саму строку `str` использовать нельзя. Решение – 1 строка.

(b) Переведите в верхний регистр первую букву второго слова строки, используя только указатель `p`. Нужно использовать только указатель `p`, саму строку `str` использовать нельзя. Решение – 1 строка.

(c) Переведите в верхний регистр все буквы строки, используя только указатель `p`. Нужно использовать только указатель `p`, саму строку `str` использовать нельзя. Решение – 1 цикл

7. В следующей программе есть переменная `a` типа `int`, указатель `p` на эту переменную и указатель `q` на `p`. Удвойте значение переменной `a`, используя только указатель `q`. Нужно использовать только указатель `q`.

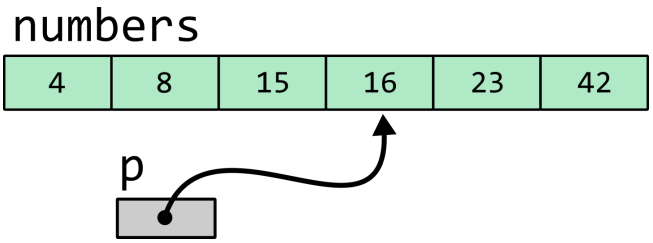
```c
#include <stdio.h>
int main()
{
    int a = 1234;
    int* p = &a;
    int** q = &p;
    // Тут нужно написать 1 строку кода
    printf("%i\n", a);
}
```



Задача 3. Указатель в массиве

Пусть есть массив и указатель на 4-й элемент этого массива:

```
int numbers[6] = {4, 8, 15, 16, 23, 42};
int* p = &numbers[3];
```



Чему равны следующие выражения:

1. `numbers[5]`
2. `*p`
3. `*(p+1)`
4. `*(p-2)`
5. `p[0]`
6. `p[1]`
7. `p[-2]`
8. `*numbers`
9. `*(numbers+5)`
10. `p - numbers`
11. `(short*)p - (short*)numbers`
12. `(char*)p - (char*)numbers`

Решение этой задачи – `.txt` файл со всеми ответами.

Задача 4. Куб по указателю

Напишите функцию `cube`, которая будет принимать на вход указатель, содержащий адрес некоторой переменной типа `float`. Функция должна возводиить в куб переменную, чей адрес хранит входящий указатель. Вызовите эту функцию из `main` и протестируйте её.

Задача 5. Умножение массива на 2

Напишите функцию `void mult2_array(int* p, size_t n)`, которая принимает указатель на первый элемент некоторого массива и число `n`, равное размеру этого массива. Вам нужно, используя этот указатель, увеличить все элементы массива в 2 раза.

Задача 6. Обращение строки

Напишите функцию `void reverse_array(char* s)`, которая принимает указатель на первый элемент некоторой строки обращать эту строку.

Задача 7. Квадратное уравнение

Напишите функцию `int solve_quadratic(double a, double b, double c, double* px1, double* px2)`, которая должна решать квадратное уравнение с коэффициентами `a`, `b` и `c`. Результат функция должна записывать по адресам `px1` и `px2`. Функция должна возвращать:

- `0` - если корней нет. По адресам `px1` и `px2` ничего записывать в этом случае не надо.
- `1` - если есть один корень. Его нужно записать по адресу `px1`.
- `2` - если есть два корня. Их нужно записать по адресам `px1` и `px2`.

  Все сравнения делать с точностью $\epsilon = 10^{-10}$.

Задача 8. Изменить символы:

Напишите функцию `void set_characters(char* begin, char* end, char c)`, которая задаёт символы в строке символом `c`. Начиная с символа, на который указывает `begin` и заканчивая символом на который указывает `end` (но не включая его). Гарантируется, что `end` указывает на символ, находящийся в этой же строке и не левее символа, на который указывает `begin`. Протестируйте функцию с помощью следующего кода:

```
#include <stdio.h>
// Тут нужно написать функции set_characters

int main()
{
    char s[] = "Sapere Aude";
    set_characters(&s[2], &s[8], 'b');
    printf("%s\n", s); // Должно напечатать Sabbbbbbude
    set_characters(s, &s[4], 'a');
    printf("%s\n", s); // Должно напечатать aaaabbbbude
}
```

Задача 9. Печать разных типов:

Напишите функцию `void polyprint(const char* type, void* p)`, которая должна будет печатать то, на что указывает указатель `p`. Тип того, на что указывает `p`, задаётся с помощью первой переменной и может принимать следующие значения:

- Если `type == "Integer"`, то `p` указывает на целое число типа `int`.
- Если `type == "Float"`, то `p` указывает на вещественное число типа `float`.
- Если `type == "Character"`, то `p` указывает на символ (тип `char`).
- Если `type == "Book"`, то `p` указывает на структуру `Book` (определение этой структуры смотрите выше).
- Если `type == "String"`, то `p` указывает на первый символ строки.
- Если `type == "IntegerArray 15"`, то `p` указывает на первый элемент массива размером 15. Элементы этого массива имеют тип `int`. Нужно распечатать все элементы через пробел. Тут нужно использовать функцию `sscanf`, для того чтобы распарсить строку `type`.
- В ином случае функция должна печатать `Error!`

В любом случае, в конце функция должна печатать символ перехода на новую строку. Для сравнения строк нужно пользоваться функцией `strcmp`. Протестируйте функцию с помощью следующего кода:

```c
#include <stdio.h>
struct book
{
    char title[50];
    int pages;
    float price;
};
typedef struct book Book;

// Тут нужно написать функцию polyprint

int main()
{
    int a = 123;
    polyprint("Integer", &a);
    float b = 1.5;
    polyprint("Float", &b);
    char c = 'T';
    polyprint("Character", &c);

    Book d = {"War and Peace", 1200, 900.0};
    polyprint("Book", &d);

    char e[] = "Sapere Aude";
    polyprint("String", e);
    int f[] = {10, 20, 30, 40, 50};
    polyprint("IngerArray 5", f);
}
```