Семинар #4: Часть 1: Указатели. Домашнее задание.

Задача 1. Одна строка

Решения всех подзадач этой задачи – одна строка. Вам нужно, используя любой текстовый редактор, создать файл в формате `.txt` и записать в него ответы на все подзадачи. После этого, файл нужно поместить в ваш репозиторий на github.

1. В следующей программе создайте указатель `p` и инициализируйте его адресом переменной `a`:

```
int main()
{
    int a = 10;
    // Тут нужно написать 1 строку кода
}
```

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

```
int main()
{
    float a = 1.1;
    // Тут нужно написать 1 строку кода
}
```

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

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

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

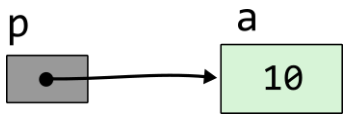
```
int main()
{
    char str[20] = "Sapere Aude";
    // Тут нужно написать 1 строку кода
}
```

5. Создайте указатель `q` и инициализируйте его адресом переменной `p`:

```
int main()
{
    int a = 10;
    int* p = &a;
    // Тут нужно написать 1 строку кода
}
```

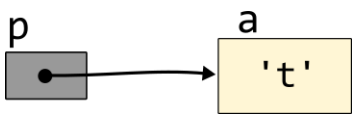
6. В следующей программе удвойте значение переменной `a`, используя только указатель `p`. Можно использовать только указатель `p`. Переменную `a` использовать нельзя.

```c
#include <stdio.h>
int main()
{
    int a = 10;
    int* p = &a;
    // Тут нужно написать 1 строку кода

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



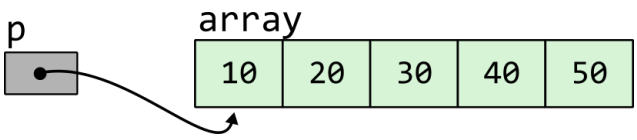
7. Переведите символ, хранящийся в переменной `a` в верхний регистр, используя только указатель `p`. Можно использовать только указатель `p`. Переменную `a` использовать нельзя.

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

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



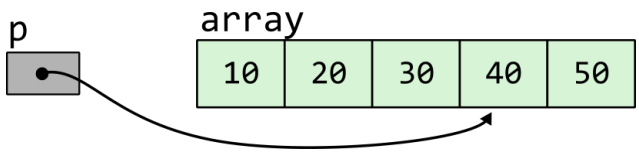
8. В следующей программе добавьте `1` к четвёртому элементу массива, используя только указатель `p` на первый элемент. Можно использовать только указатель `p`. Массив `a` использовать нельзя.

```c
#include <stdio.h>
int main()
{
    int a[5] = {10, 20, 30, 40, 50};
    int* p = &a[0];
    // Тут нужно написать 1 строку кода

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



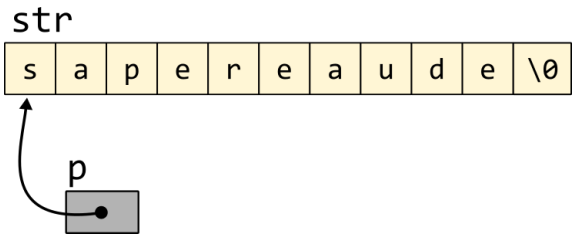
9. В следующей программе используйте указатель на четвёртый элемент массива(p), чтобы добавить 1 к первому элементу массива. Можно использовать только указатель p. Массив a использовать нельзя.

```c
#include <stdio.h>
int main()
{
    int a[5] = {10, 20, 30, 40, 50};
    int* p = &a[3];
    // Тут нужно написать 1 строку кода

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



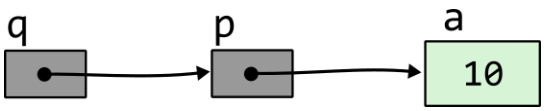
10. В следующей программе переведите в верхний регистр все буквы строки указатель. Можно использовать только указатель p. Строку str использовать нельзя. Решение – 1 цикл (несколько строк кода).

```c
#include <stdio.h>
int main()
{
    char str[] = "sapereaude";
    char* p = &str[0];
    // Тут нужно написать код

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



11. Удвойте значение переменной a, используя указатель q. Можно использовать только указатель q. Использовать переменные a и p нельзя.

```c
#include <stdio.h>
int main()
{
    int a = 10;
    int* p = &a;
    int** q = &p;
    // Тут нужно написать 1 строку кода

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

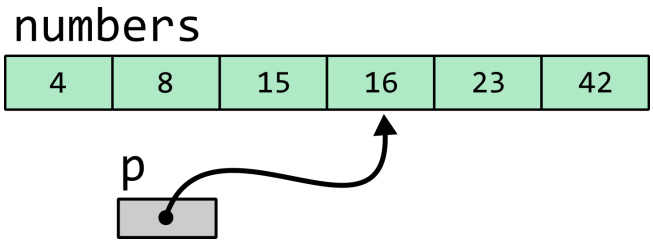


Задача 2. Указатель в массиве

Решения всех подзадач этой задачи – одно число. Вам нужно, используя любой текстовый редактор, создать файл в формате `.txt` и записать в него ответы на все подзадачи. После этого, файл нужно поместить в ваш репозиторий на github.

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`

Задача 3. Куб по указателю

Напишите функцию `cube`, которая будет принимать на вход адрес некоторой переменной типа `float`.

```
void cube(float* px)
```

Функция должна возводиить в куб переменную, чей адрес хранит входящий указатель. Вызовите эту функцию из `main` и протестируйте её.

Задача 4. Умножение массива на 2

Напишите функцию

```
void mult2(int* p, size_t n)
```

которая будет принимать указатель на первый элемент некоторого массива `p` и целое число `n`, равное размеру этого массива. Функция должна увеличивать все элементы массива в 2 раза. Решите эту задачу в двух вариантах:

a. К указателю можно применять только оператор сложения `+` и оператор разыменования `*`.

b. К указателю можно применять только оператор квадратные скобки `[]`.

Задача 5. 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 `eps = 1e-10`.

Задача 6. Обмен

Напишите функцию

```
int exchange(int* pa, int b)
```

которая будет присваивать переменной по адресу `pa` значение `b` и возврщает старое значение переменной, на которую указывает `pa`. Протестируйте функцию с помощью следующего кода:

```
#include <stdio.h>
// Тут нужно написать функцию exchange

int main()
{
    int a = 10;
    printf("%i\n", exchange(&a, 20));  // Напечатает 10
    printf("%i\n", a);                 // Напечатает 20
}
```

Задача 7. Максимум по адресу

Напишите функцию

```
int* max(int* pa, int* pb)
```

которая будет принимать два адреса на числа и находить максимум из этих чисел. Функция должна возвращать адрес наибольшего числа. Протестируйте функцию с помощью следующего кода:

```
#include <stdio.h>
// Тут нужно написать функцию max

int main()
{
    int a = 10;
    int b = 30;
    int c = 20;
    *max(max(&a, &b), &c) += 1;
    printf("%i %i %i\n", a, b, c);  // Напечатает 10 31 20
}
```

Задача 8. Изменить символы

Напишите функцию

```
void set_characters(char* begin, char* end, char c)
```

которая будет задавать 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`.

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

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

Задача 9. Используемые символы

Напишите функцию

```
void used_chars(const char* str, char* used)
```

Которая бы принимала указатель на первый символ строки `str` и указатель на первый символ другой строки `used`. Функция должна находить все буквы в строке `str` и записывать их в `used` в алфавитном порядке. Символы, не являющиеся буквами, игнорируются. Если буква встречается в `str` (заглавная или строчная), в `used` добавляется соответствующая строчная буква. Считайте, что в строке `used` достаточно места.

```
#include <stdio.h>
#include <string.h>
// Тут нужно написать функцию used_chars

int main()
{
    char s[50] = "Sapere Aude";
    char u[30];
    used_chars(s, u);
    printf("%s\n", u);  // Напечатает "adeprsu"

    strcpy(s, "123!$@");
    used_chars(s, u);
    printf("%s\n", u);  // Ничего не напечатает (только перенос строки)

    strcpy(s, "The Quick Brown Fox Jumps Over The Lazy Dog!");
    used_chars(s, u);
    printf("%s\n", u);  // Напечатает "abcdefghijklmnopqrstuvwxyz"
}
```

Задача 10. Максимальная строка

Напишите функцию

```
char* strmax(char** strings, size_t n)
```

которая будет принимать на вход массив строк (на самом деле указатель на первый элемент массива указателей типа `char*`) и размер этого массива. Функция дожна будет находить максимальную строку и вовращать указатель на неё. Для сравнения строк используйте функцию `strcmp`. Протестируйте функцию с помощью следующего кода:

```
#include <stdio.h>
#include <string.h>
// Тут нужно написать функцию strmax

int main()
{
    char a[] = "Cat";
    char b[] = "Mouse";
    char c[] = "Wolf";
    char d[] = "Kangaroo";
    char e[] = "Elephant";

    char* animals[5] = {&a[0], &b[0], &c[0], &d[0], &e[0]};
    char* x = strmax(animals, 5);
    printf("%s\n", x);  // Напечатает Wolf
}
```

Необязательные задачи (не входят в ДЗ, никак не учитываются)

Задача 1. Обращенная копия строки

Напишите функцию

```
void reverse_copy(const char* source, char* destination, size_t n)
```

которая принимает указатель на первый элемент некоторой строки (`source`), а также указатель на первый элемент другой строки (`destination`). Функция должна записывать в `destination` обращённую копию строки `source`. Также, на вход функции передаётся `n` – вместимость строки `destination`. Если длина `source` будет больше или равна `n`, то вы должны будете скопировать только `n - 1` символов и поставить нулевой символ в конце.

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

int main()
{
    char s[10] = "Elephant";
    char d[10];
    reverse_copy(s, d, 10);
    printf("%s\n", d); // Должно напечатать tnahpelE

    reverse_copy(s, d, 5);
    printf("%s\n", s); // Должно напечатать tnah
}
```

Задача 2. 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 == "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>

// 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);

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