Семинар #9: Память. 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`.

6. В следующей программе создаётся структура `Book` из семинара про структуры:

```
struct book {
    char title[50];
    int pages;
    float price;
};
typedef struct book Book;

int main() {
    Book b = {"Fahrenheit 451", 400, 700.0};
    // Тут нужно написать 1 строку кода
}
```

   (a) Создайте указатель `pb` и сделайте так, чтобы он указывал на структуру `b`.

   (b) Создайте указатель `pprice` и сделайте так, чтобы он указывал на поле `price` структуры `b`.

   (c) Создайте указатель `pc` и сделайте так, чтобы он указывал символ `'t'` поля `title` структуры `b`.

7. В следующей программе создаётся переменная `a` типа `float` и `p` указатель, который хранит её адрес:

```
int main() {
    float a = 1.2;
    float* p = &a;
    // Тут нужно написать 1 строку кода
}
```

   Создайте указатель `pp` и сделайте так, чтобы он указывал на указатель `p`.

8. В следующей программе создаётся структура `Book` из семинара про структуры и указатель на неё:

```
struct book {
    char title[50];
    int pages;
    float price;
};
typedef struct book Book;

int main() {
    Book b = {"Fahrenheit 451", 400, 700.0};
    Book pb = &b;
    // Тут нужно написать 1 строку кода
}
```

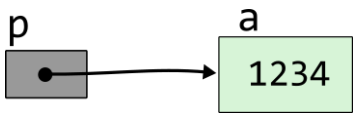
   Создайте указатель `ppb` и сделайте так, чтобы он указывал на указатель `pb`.

Задача 2: Использование указателей

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

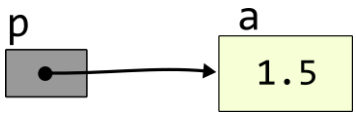
1. В следующей программе была создана переменная `a` и указатель на неё `p`. Удвойте значение переменной `a`, используя только указатель `p`. Нужно использовать указатель `p`, саму переменную `a` использовать нельзя.

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

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



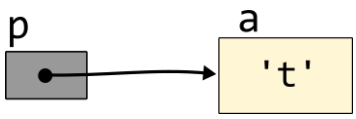
2. В следующей программе была создана переменная `a` типа `float` и указатель на неё `p`. Возведите значение переменной `a` в квадрат, используя только указатель `p`. Нужно использовать только указатель `p`, саму переменную `a` использовать нельзя.

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

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



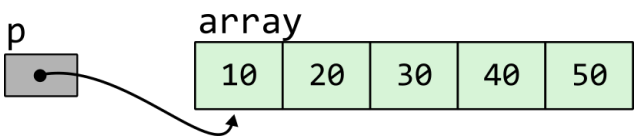
3. В следующей программе была создана переменная `a` типа `char` и указатель на неё `p`. Переведите символ, хранящийся в переменной `a` в верхний регистр, используя только указатель `p`. Нужно использовать только указатель `p`, саму переменную `a` использовать нельзя.

```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];
    // Тут нужно написать 1 строку кода

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



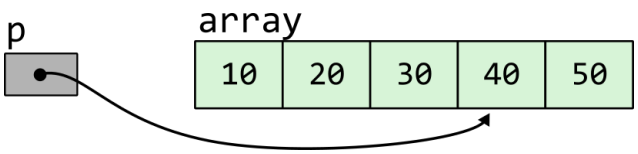
(a) Добавьте `1` к первому элементу массива (`array[0]`), используя только указатель `p`. Нужно использовать только указатель `p`, сам массив `array` использовать нельзя. Решение – 1 строка.

(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];
    // Тут нужно написать 1 строку кода

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



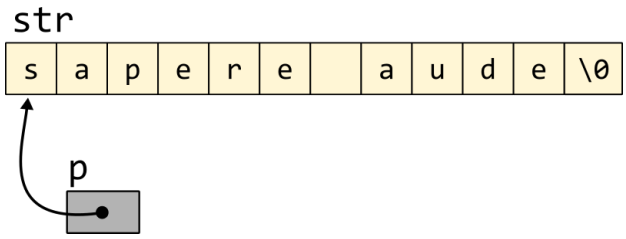
(a) Добавьте `1` к первому элементу массива (`array[0]`), используя только указатель `p`. Нужно использовать только указатель `p`, сам массив `array` использовать нельзя. Менять `p` тоже нельзя. Решение – 1 строка.

(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.

```
#include <stdio.h>
int main() {
    int str[] = "sapere aude";
    int* p = &str[0];
    // Тут нужно написать 1 строку кода

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



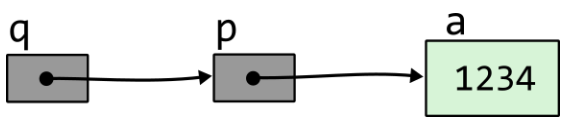
(a) Переведите в верхний регистр первую букву строки, используя только указатель `p`. Нужно использовать только указатель `p`, саму строку `str` использовать нельзя. Решение – 1 строка.

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

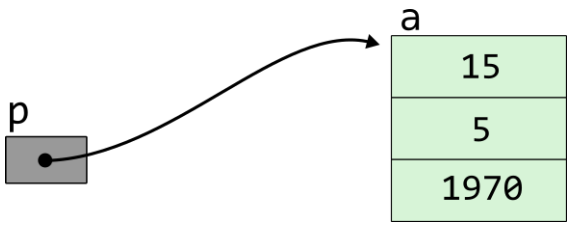
7. В следующей программе была создана переменная `a` типа `int`, указатель `p` на эту переменную и указатель `q` на указатель `p`. Удвойте значение переменной `a`, используя только указатель `q`. Нужно использовать только указатель `q`. Использовать переменную `a` и указатель `p` нельзя.

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



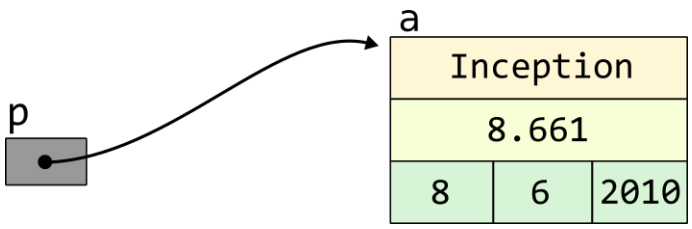
8. В следующей программе была создана структура `a` типа `Date` и указатель на эту структуру. Добавьте `1` к значению поля `year`, используя только указатель `p`.

```
#include <stdio.h>
struct date {
    int day, month, year;
};
typedef struct date Date;
int main() {
    Date a = {15, 5, 1970};
    Date* p = &a;
    // Тут нужно написать 1 строку кода
    printf("%d %d %d\n",
           a.day, a.month, a.year);
}
```



9. В следующей программе была создана структура `a` типа `Movie` из семинара про структуры. Также создан указатель `p` на эту структуру.

```c
#include <stdio.h>
struct date {
    int day, month, year;
};
typedef struct date Date;

struct movie {
    char title[50];
    float rating;
    Date release_date;
};
typedef struct movie Movie;

int main() {
    Movie a = {"Inception", 8.661, {8, 6, 2010}};
    Movie* p = &a;
    // Тут нужно написать 1 строку кода
}
```

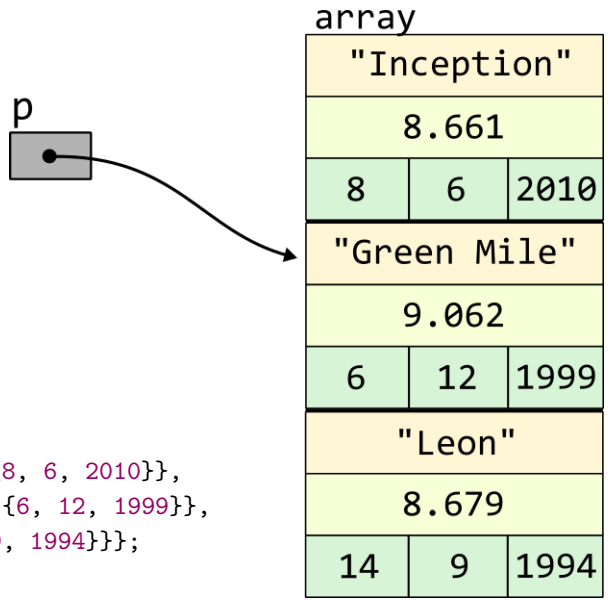


(a) Увеличьте на 1 значение поля `rating`, используя только указатель `p`.

(b) Увеличьте на 1 значение поля месяца выхода фильма, используя только указатель `p`.

10. В следующей программе был создан массив `array` из структур типа `Movie` и указатель `p`, который указывает на второй элемент массива (`array[1]`).

```c
#include <stdio.h>
struct date {
    int day, month, year;
};
typedef struct date Date;

struct movie {
    char title[50];
    float rating;
    struct date release_date;
};
typedef struct movie Movie;

int main() {
    Movie array[3] = {{"Inception", 8.661, {8, 6, 2010}},
                      {"Green Mile", 9.062, {6, 12, 1999}},
                      {"Leon", 8.679, {14, 9, 1994}}};
    Movie* p = &array[1];
}
```



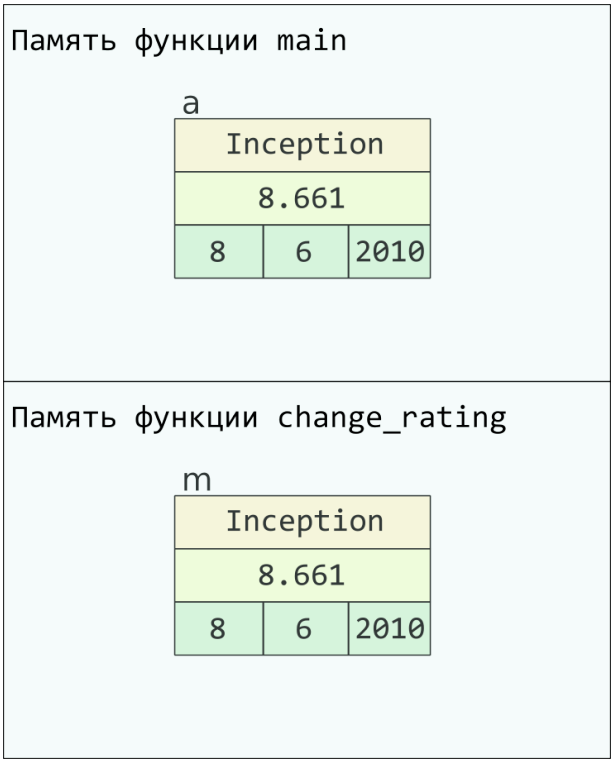
(a) Увеличьте на 1 значение рейтинга фильма `Inception`, используя только указатель `p`. При этом менять `p` нельзя, он должен указывать на `array[1]`.

(b) Удвойте значение года выхода фильма `Leon`, используя только указатель `p`. При этом менять `p` нельзя, он должен указывать на `array[1]`.

Задача 3: Передача в функцию по указателю

Передача в функцию по значению

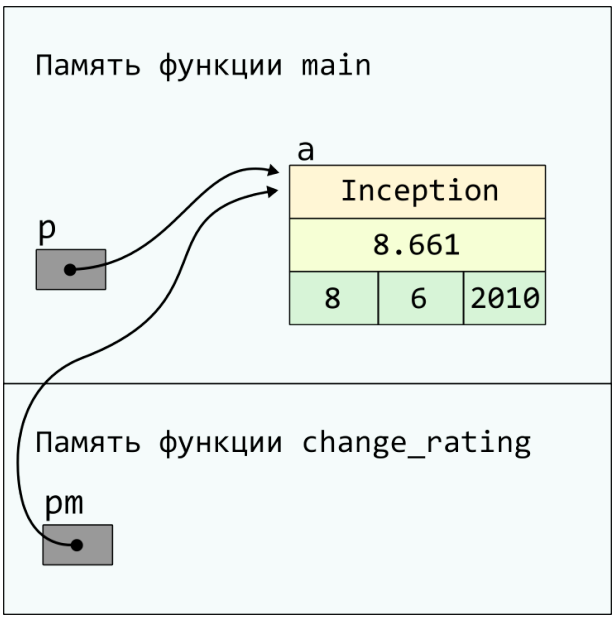
```c
#include <stdio.h>

struct movie {
    char title[50];
    float rating;
    struct date release_date;
};
typedef struct movie Movie;

void change_rating(Movie m) {
    m.rating += 1;
}
int main() {
    Movie a = {"Inception", 8.661,
                {8, 6, 2010}};
    change_rating(a);
}
```



Всё, что передаётся в функцию, копируется. Поэтому функция `change_rating` будет менять поле `rating` у копии структуры `a`, а изначальная структура не изменится.

Передача в функцию по указателю:

```c
#include <stdio.h>
struct movie {
    char title[50];
    float rating;
    struct date release_date;
};
typedef struct movie Movie;

void change_rating(Movie* pm) {
    pm->rating += 1;
}
int main() {
    Movie a = {"Inception", 8.661,
                {8, 6, 2010}};
    Movie* p = &a;
    change_rating(&a);
}
```



Всё, что передаётся в функцию, копируется. Но теперь туда копируется указатель, который содержит адрес стурктуры `a`. Используя этот указатель, мы можем изменить изначальную структуру. Более того, так как указатель занимает меньше памяти, его копирования в функцию происходит быстрее, чем копирование всей структуры.

Подзадачи:

1. Напишите функцию `void inc(int* p)`, которая будет принимать на вход указатель на переменную типа `int` и увеличивать на 1 эту переменную. Протестируйте функцию, вызвав её из функции `main` с помощью следующего кода:

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

int main() {
    int a = 20;
    inc(&a);
    printf("%i\n", a); // Должно напечатать 21
}
```

2. Напишите функцию `void cube(float* p)`, которая будет принимать на вход указатель на переменную типа `float` и возводиить в куб эту переменную. Протестируйте функцию, вызвав её из функции `main` с помощью следующего кода:

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

int main() {
    float a = 1.5;
    cube(&a);
    printf("%f\n", a); // Должно напечатать 3.375
}
```

3. Напишите функцию `void to_upper(char* p)`, которая будет принимать на вход указатель на переменную типа `char` и, если символ, на который указывает этот указатель, является строчной буквой, функция должна делать эту букву заглавной. Протестируйте функцию, вызвав её из функции `main` с помощью следующего кода:

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

int main() {
    char a = 't';
    to_upper(&a);
    printf("%c\n", a); // Должно напечатать T
}
```

4. Напишите функцию `void inc_array(int* p, int size)`, которая будет принимать на вход указатель на первый элемент массива и целое число `size` – размер массива. Функция должна увеличивать на 1 все элементы массива. Протестируйте функцию, вызвав её из функции `main` с помощью следующего кода:

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

int main() {
    int a[5] = {10, 20, 30, 40, 50};
    inc_array(a, 5);

    for (int i = 0; i < 5; ++i)
        printf("%i ", a[i]);
    // Цикл должен напечатать 11 21 31 41 51
}
```

5. Напишите функцию `void string_to_upper(char* p)`, которая будет принимать на вход указатель на первый символ строки и переводить эту строку в верхний регистр.

6. Напишите функцию `void trim_after_first_space(char* p)`, которая будет принимать на вход указатель на первый символ строки и укорачивать строку до первого пробела. Протестируйте функции, вызвав её их из функции `main` с помощью следующего кода:

```
#include <stdio.h>
// Тут вам нужно написать функции string_to_upper и trim_after_first_space
int main() {
    char a[] = "Sapere Aude";
    string_to_upper(a);
    printf("%s\n", a); // Должно напечатать SAPERE AUDE
    trim_after_first_space(a);
    printf("%s\n", a); // Должно напечатать SAPERE
}
```

7. Напишите функцию `void increase_rating(Movie* p)`, которая будет принимать указатель типа `Movie*` и увеличивать рейтинг фильма, на которой указывает `p`, на 1.

8. Напишите функцию `void change_year_of_movies(Movie* p, int size)`, которая принимает на вход указатель на первый элемент массива структур типа `Movie` и размер этого массива. Функция должна увеличивать год выхода всех фильмов на 1. Протестируйте функции, вызвав их из функции `main` с помощью следующего кода:

```
#include <stdio.h>
struct date {
    int day, month, year;
};
typedef struct date Date;
struct movie {
    char title[50];
    float rating;
    struct date release_date;
};
typedef struct movie Movie;

void print_date(const Date* pd) {
    printf("%02d.%02d.%04d", pd->day, pd->month, pd->year);
}
void print_movie(const Movie* pm) {
    printf("Title: %s\nRating: %.2f\nDate: ", pm->title, pm->rating);
    print_date(&pm->release_date);
    printf("\n");
}
// Тут вам нужно написать функции increase_rating и change_year_of_movies

int main() {
    Movie a[3] = {{"Inception", 8.661, {8, 6, 2010}},
                  {"Green Mile", 9.062, {6, 12, 1999}},
                  {"Leon", 8.679, {14, 9, 1994}}};
    increase_rating()
    change_year_of_movies(a, 3);
    for (int i = 0; i < 3; ++i)
        print_movie(&a[i]);
}
```

Задача 4:

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 == "Date"`, то `p` указывает на структуру `Date` (определение этой структуры смотрите выше).
- Если `type == "Movie"`, то `p` указывает на структуру `Movie` (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 date {
    int day, month, year;
};
typedef struct date Date;
struct movie {
    char title[50];
    float rating;
    struct date release_date;
};
typedef struct movie Movie;
// 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);

    Date d = {15, 5, 1970};
    polyprint("Date", &d);
    Movie e = {"Inception", 8.661, {8, 6, 2010}};
    polyprint("Movie", &e);

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

Задача 5:

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 6:

Как выглядит память, инициализируемая при создании следующих переменных (в системе с порядком байт Little Endian):

- `int a = 0x11223344;`
- `int b = 65535;`
- `int c = -1;`
- `int array[3] = {10, 2000, 65535};`
- `char str[8] = ''Hello'';`
- `float x = 1.0f;`
- ```
  struct data {
      char str[5];
      int number;
  };
  struct data c = {"Cat", 100000};
  ```

Память представить в виде последовательности 2-значных шестнадцатеричных чисел. Например число $123456 = 1e240_{16}$ будет храниться в памяти как `40 E2 01 00`.

Подсказка: Чтобы проверить, как будет выглядеть память, можно создать указатель типа `char*` на эту память и распечатать каждый байт в виде шестнадцатеричного числа.
  ```