

Last updated: 2019-02-17  
vadimov@i.ua

## Практическое занятие 2 (семестр2).

Задача 2.1. Напишите программу для ввода количества отработанных персоналом часов и размера почасовой оплаты каждого. Затем выведите суммарную зарплату персонала. (Удостоверьтесь в правильности ввода.)

Задача 2.2. Напишите программу для преобразования метров в футы и футов в дюймы. Организуйте ввод числа метров и вывод на экран соответствующего числа футов и дюймов. Повторяйте эту процедуру до тех пор, пока пользователь не введет 0 в качестве числа метров.

Задача 2.3. Ниже приведена программа на языке C. Перепишите ее в соответствии со стилем ввода/вывода C++.

```
/* Преобразуйте эту программу на C в соответствии со стилем программирования C++.
Эта программа подсчитывает наименьшее общее кратное*/
#include <stdio.h>
int main(void) {
    int a, b, d, min;
    printf("Enter two numbers: ");
    scanf("%d%d", &a, &b);
    min = a>b?b:a;
    for (d=2; d<min; d++)
        if (((a%d) == 0) && ((b%d) == 0))
            break ;
    if (d==min) {
        printf("No common denominators\n");
        return 0;
    }
    printf ("The lowest common denominator is %d\n", d);
    return 0;
}
```

Задача 2.4. Создайте класс **card**, который поддерживает каталог библиотечных карточек. Этот класс должен хранить заглавие книги, имя автора и выданное на руки число экземпляров книги. Заглавие и имя автора храните в виде строки символов, а количество экземпляров - в виде целого числа. Используйте открытую функцию-член **store()** для запоминания информации о книгах и открытую функцию-член **show()** для вывода информации на экран. В функцию **main()** включите краткую демонстрацию работы созданного класса.

Задача 2.5. Создайте класс с циклической очередью целых (см. Unit 2 "A more practical example"). Сделайте очередь длиной 100 целых. В функцию **main()** включите краткую демонстрацию ее работы.  
Провести инициализацию с помощью конструктора.

### Подсказка:

```
class q_type {
    int queue[SIZE]; // содержит очередь
    int head, tail; // индекс вершины и хвоста
```

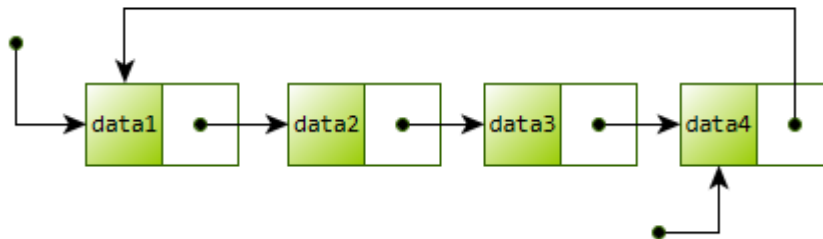
```

public:
    void init();    // инициализация
    void q(int num); // запоминание
    int deq();     // восстановление
};

```

Задача 2.6. Реализуйте на C решение Задачи 2.5. (см. Unit 1 "A singly-linked list").

Пример циклической очереди, построенной при помощи указателей:



Задача 2.7. Объясните, почему работает такой код:

```

#include <cstdlib>
#include <iostream>
using namespace std;
int main (int argc, char **argv){
    cout << argv[argc-1] << endl
    << argv[argc-1] << endl;
    return EXIT_SUCCESS;
}

```

Задача 2.8. Создайте функцию **myroot()**, которая возвращает квадратный корень своего аргумента. Перегрузите **myroot()** тремя способами (см. Unit 2 "Introducing Function Overloading"): чтобы получить квадратный корень целого, длинного целого и числа с плавающей точкой двойной точности. (Для непосредственного подсчета квадратного корня вы можете использовать стандартную библиотечную функцию **sqrt()**).

Задача 2.9. Стандартная библиотека C++ содержит три функции:

```

double atof (const char *s);
int atoi (const char *s);
long atol(const char *s);

```

Эти функции возвращают численное значение, содержащееся в строке, на которую указывает *s*. Заметьте, что **atof()** возвращает **double**, **atoi** возвращает **int** и **atol** возвращает **long**. Почему нельзя перегрузить эти функции? Продемонстрируйте пример кода.

Задача 2.10. Создайте свою функцию **min()**, которая возвращает наименьший из двух численных аргументов, используемых при ее вызове. Перегрузите функцию **min()** так, чтобы она воспринимала в качестве аргументов символы, целые и действительные двойной точности (см. Unit 2 "Introducing Function Overloading").

Задача 2.11. Создайте свою функцию **sleep()**, приостанавливающую работу компьютера на столько секунд, сколько указано в аргументе функции. Перегрузите **sleep()** так, чтобы она могла вызываться или с целым, или со строкой, задающей целое. Например, оба этих вызова должны

заставить компьютер остановиться на 10 секунд:

```
sleep(10);  
sleep("10");
```

Продемонстрируйте работу ваших функций, включив их в короткую программу.

Задача 2.12. Напишите программу, использующую стиль ввода/вывода C++, для ввода двух целых с клавиатуры и затем вывода на экран результата возведения первого в степень второго. (Например, пользователь вводит 2 и 4, тогда результатом будет 24, или 16.)

Задача 2.13. Создайте функцию **rev\_str()** для изменения порядка следования символов строки на обратный. Перегрузите **rev\_str()** так, чтобы она могла вызываться с одним или двумя символьными строками. Если функция вызывается с одной строкой, то операция должна осуществляться с ней. Если она вызывается с двумя строками, то результирующая строка должна оказаться во втором аргументе (см. Unit 2 "Overloaded functions that differ in the number of their arguments"). Например:

```
char s1 [80], s2 [80];  
strcpy (s1, "hello");  
rev_str(s1, s2); // reversed string goes in s2, s1 untouched  
rev_str(s1);    // reversed string is returned in s1
```

Задача 2.14. Создайте класс **stopwatch** для имитации секундомера. Используйте конструктор для начальной установки секундомера в 0. Образуйте две функции-члена **start()** и **stop()** соответственно для запуска и остановки секундомера. Включите в класс и функцию-член **show()** для вывода на экран величины истекшего промежутка времени. Также используйте деструктор для автоматического вывода на экран времени, прошедшего с момента создания объекта класса **stopwatch**, до его удаления. (Для простоты время приведите в секундах.)

Подсказка (нет никакой гарантии, что "подсказка" правильно работает):

```
#include <iostream>  
#include <ctime>  
using namespace std;  
  
class timer {  
    clock_t start;  
public:  
    timer(); // constructor  
    ~timer(); // destructor  
};  
  
timer::timer() {  
    start=clock();  
}  
  
timer::~~timer() {  
    clock_t end;  
    end=clock();  
    cout << "Elapsed time: " << (end-start) / CLOCKS_PER_SEC << "\n";  
}  
  
int main() {  
    timer ob;  
    char c;  
    // delay ...  
    cout << "Press a key followed by ENTER:";
```

```

    cin >> c;
    return 0;
}

```

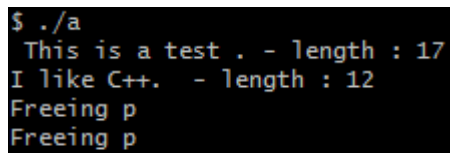
Задача 2.15. Создайте новый вариант разработанного ранее класса **strtype**, в котором используется конструктор с параметром (см . Unit 2 "An Example that Shows the Need for Both a Constructor and a Destructor Function"). В вашей версии класса **strtype** строка получает свое начальное значение с помощью конструктора.

Подсказка:

```

class strtype {
    char *p;
    int len;
public:
    strtype(char *ptr);
    ~strtype();
    void show();
};
...
int main() {
    strtype s1("This is a test."), s2("I like C++.");
    s1.show();
    s2.show();
    return 0;
}

```



```

$ ./a
This is a test . - length : 17
I like C++. - length : 12
Freeing p
Freeing p

```

Задача 2.16. Измените класс **stack** (см . Unit 2 "An Improved Version of the Stack class") так, чтобы память для стека выделялась динамически. При этом длина стека должна задаваться параметром конструктора. (Не забудьте освободить эту память с помощью деструктора!)

Задача 2.17. Создайте класс **time\_and\_date**, которому при его создании передается текущее системное время и дата в виде параметров конструктора. Этот класс должен включать в себя функцию-член, выводящую время и дату на экран.

Подсказка: Для нахождения и вывода на экран этих данных воспользуйтесь стандартной библиотечной функцией времени и даты.

Задача 2.18. Создайте класс **mybox**, конструктору которого передаются три значения типа **double**, представляющие собой длины сторон параллелепипеда. Класс **mybox** должен подсчитывать его объем и хранить результат также в виде значения типа **double**. Включите в класс функцию-член **volume()**, которая будет выводить на экран объем любого объекта типа **mybox**.