МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСІТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТІТУТ»

КАФЕДРА МЕДИЧНОЇ КІБЕРНЕТИКИ ТА ТЕЛЕМЕДИЦИНИ

**Лабораторна робота № 8**

з дисципліни «Алгоритмізація та програмування»

на тему: “Функції користувача”

Варіант 16

**Виконав:**

Студент групи БС-32

Хоміцевич Микита

**Перевірила:**

викл.

Матвійчук А.О.

Київ-2014

**Задание**

****

**□ Лабораторна робота виконана без зауважень**

**□ Лабораторна робота має зауваження:**

**□ присутні зауваження до блок-схем:**

**□ виконані не за стандартом**

**□ не відповідають коду**

**□ присутній код**

**□ інші зауваження:**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**□присутні зауваження до коду:**

**□задача завдання вирішена хибно**

**□ код програми не компілюється**

**□ використано глобальні змінні**

**□ недостатня декомпозиція на функціонуванні користувача**

**□ статичні змінні при роботі з масивами**

**□ оформлення коду**

**□присутні зайві символи «{»та «}»**

**□інші зауваження:**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**□невірні відповіді на запитання:**

**□№1 □№2 □№3**

**□маються інші зауваження: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Программный код**

#include <iostream>

#include <conio.h>

#include <locale.h>

using namespace std;

void MAIN() {

setlocale(LC\_ALL, "RUS");

int x, sum = 0;

cout << "Введите целое число x: ";

cin >> x;

int x1 = x;

while (x > 0) {

if (x % 2 == 0) { x1 >>= 1; if (x1 % 2 == 0) { ++sum; x >>= 1; } else { x >>= 1; } }

else { x >>= 1; x1 >>= 1; }

}

cout << "Количество пар нулевых битов в числе х: " << sum << endl;

}

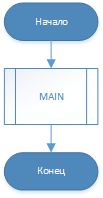
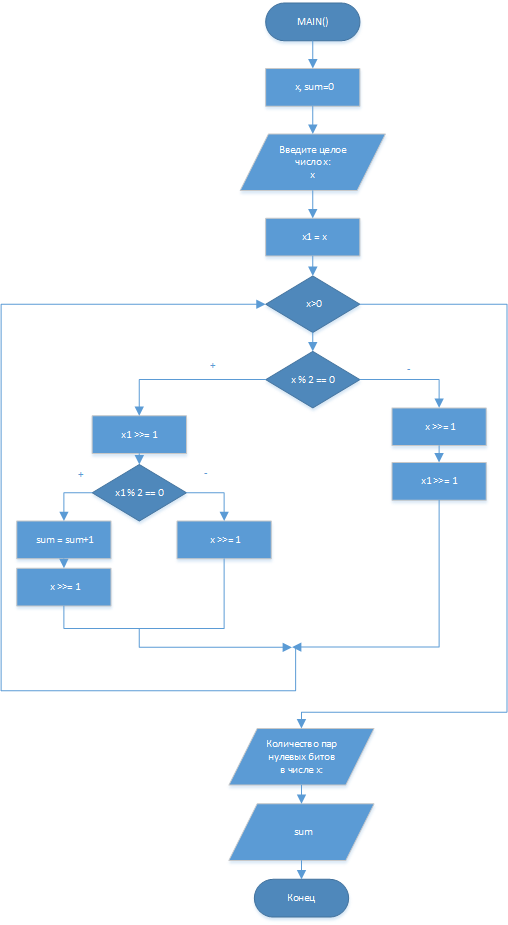
int main(void) {

MAIN();

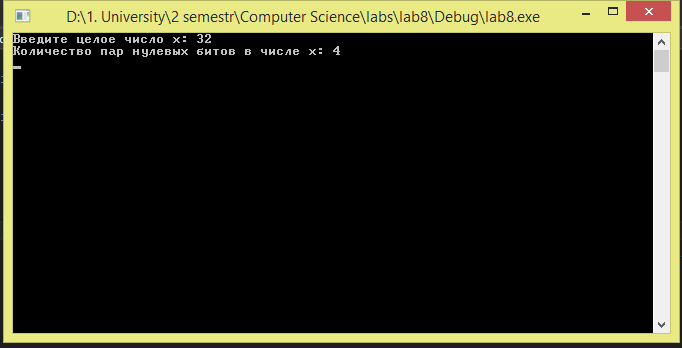
\_getch();

}

**Блок-схема**

** **

**Скриншот работы программы**

****

**Ответы на контрольные вопросы**

1. Что такое прототип (сигнатура) функции, зачем он нужен?

Прототип функции (сигнатура) – это [объявление](http://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D1%8F%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D0%B5_(%D0%B8%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0)) [функции](http://ru.wikipedia.org/wiki/%D0%A4%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D1%8F_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)), что не содержит тела функции, но указывает имя функции, [арность](http://ru.wikipedia.org/wiki/%D0%90%D1%80%D0%BD%D0%BE%D1%81%D1%82%D1%8C), а так же [типы](http://ru.wikipedia.org/wiki/%D0%A2%D0%B8%D0%BF_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85) аргументов и возвращаемый тип данных.

В основном сигнатуры используются программистами для проектирования функции «на будущее» - как описание интерфейса.

Пример:

int function(int n);

1. Чем отличается определение функции отобъявления ее прототипа?

Когда мы создаем прототип функции, мы не описываем саму ф-ию, т.е. не создаем тело функции, где идет описание работы данной функции. Так же в прототипе функции мы може не указывать имена переменных, которые будут использоваться, а можем лишь указать тип используемых параметров.

1. Может ли в одной ф-ии встречаться несколько операторов return?

Да. Может. К примеру, когда мы используем рекурсивную ф-ию и через условие возвращаем разные значения.

1. Сколько значений может вернуть ф-ия?

Одно.

1. Как следует организовывать ф-ии, если нужно вернуть значения нескольких переменных в точку вызова этой ф-ии?

Через стандартый способ возврата значения из функции в языке Си никак, но есть возможность использовать (в языке С++) передачу аргументов в ф-ию через ссылки. Тогда мы можем негласно вернуть сколько угодно значений.

Пример:

#include <iostream>

void lal(int &a, int &b) {

a \*= 100; b \*= 200; }

int main() {

int a, b;

std::cin >> a >> b; //a = 12, b = 20

lal(a, b);

std::cout << a << " " << b << std::endl; // вернет a = 1200, b = 4000;

}

1. Что такое формальные параметры функции?

Это параметры, имена которых используются в теле функции.

1. Что такое фактические параметры функции?

Это параметры, которые передаются в функцию при ее вызове.

1. Какие особенности передачи параметров в функцию в языке С++?

Если, к примеру, мы передаем в функцию какие-то фактические параметры в виде обычной переменной, то далее, изменяя в функции уже данный фактический параметр мы при выходе из функции не поменяем значение фактического параметра.

А если передадим указатель на данную переменную (или ссылку), то данное значение можем изменять в функции, и соответсвенно, менять вне функции

1. Какие есть способы вызова функции?
2. Вызов функции через написание имя функции и в скобках указание параметров ф-ии
3. Рекурсивно
4. Через указатель на функцию
5. Вызов ф-ии в отдельном потоке
6. Значения какого типа возвращает ф-ия, если тип не указан?

Т.к. Си (и с++) язык со статической типизацией, то при вызове ф-ии обязательно должен быть указан тип возвращаемого значения. Но если, к примеру, мы вызываем функцию через указатель, то можно возпользоваться безтиповым (универсальным) значением “**void\***”. Тогда тип возращаемого значения будет определен по типу указателя.