**Лабораторная работа № 6.**

**Битовые (поразрядные) операции.**

1. Постановка задачи:

Объясните, как работает программа.

Запустите программу и объясните результат.

Вопросы заданы в комментариях к коду. Ответы оформите либо в

комментариях к коду, либо отдельным блоком после кода

#include<stdio.h>

#include<conio.h>

void main()

{

int i,a,b; // Сколько (битов/байтов) памяти занимают a и b?

/\* —————————————————— \*/

printf("\nВведите два целых числа: "); scanf("%d %d",&a,&b);

printf("\n");

/\* ————————————————– \*/

printf("Операция 'инверсия битов' в числе a: 0x%x=%d\n", ~a,~a); /\* Объясните

запись 0x%x=%d \*/

printf("Операция 'инверсия битов' в числе b: 0x%x=%d\n", ~b,~b); /\* Объясните, как

работает операция инверсия \*/

printf("\n");

/\* ————————————————– \*/

printf("Операция 'сдвиг <<2' в числе a : 0x%x=%d\n", a<<2,a<<2); /\* Пропишите в

двоичном коде операцию сдвига на конкретном примере\*/

printf("Операция 'сдвиг >>2' в числе a : 0x%x=%d\n", a>>2,a>>2); /\* Пропишите в

двоичном коде операцию сдвига на конкретном примере\*/

printf("\n");

/\* ————————————————– \*/

printf("Операция 'битовое И' a&b : 0x%x=%d\n", a&b,a&b); /\* Объясните, как

работает операция битовое И \*/

printf("Операция 'битовое ИЛИ' a|b : 0x%x=%d\n", a|b,a|b); /\* Объясните, как

работает операция битовое ИЛИ \*/

printf("Операция 'битовое исключающее ИЛИ ': 0x%x=%d\n", a^b,a^b);

/\* Объясните, как работает операция битовое исключающее ИЛИ \*/

printf("\n");

getch();

}

Результат выполненной работы:

1. По 4 байта занимает каждая переменная
2. 0x%x=%d - Программа выводит инверсию числа a в шестнадцатеричном представлении без знака, выводится инверсия целого числа со знаком
3. Инверсия битов инвертирует биты, т.е. каждый бит со значением 1 получает знчение 0 и наоборот.
4. 00000111«2==00011100
5. 00000111»2==00000001
6. Сравнивает последовательно разряд за разрядом значения двух операндов. Для каждого разряда результат равен 1, когда оба соответствуюхих разряда значений операндов равны 1.
7. Сравнивает последовательно разряд за разрядом два своих операнда. Для каждого разряда результат равен 1, если один из 2-х ( но не оба) оба соответствующих разряда операндов равны 1.

3. Постановка задачи:

Объясните, как работает программа.

#include<stdio.h>

#include<conio.h>

void main()

{

unsigned int n;

printf("Введите натуральное число: "); scanf("%u",&n);

if (n&0x0001)

printf("Число %u является нечѐтным.\n",n);

else printf("Число %u является чѐтным.\n",n);

getch();

}

Результат выполненной работы:

Через операцию битого И проверяется число на четность, если в младшем разряде получается 1 в 2-ом представлении, тогда у нас число нечетное, а если получается 0, то число является четным и выполняется else.

4.Постановка задачи:

Запустите код и объясните результат программы «Использование

операции "сдвиг вправо" для вывода на экран двоичного

представления данного натурального числа»

Заполните комментарии к программе.

#include<stdio.h>

#include<conio.h>

void main()

{

long int a; /\* Исходное натуральное число:

если a<255, то для хранения числа требуется \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ памяти;

если a<65535, то для хранения числа требуется \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ памяти;

если a<2147483647, то для хранения числа требуется \_\_\_\_\_\_\_\_\_\_\_\_ памяти \*/

int i, // Параметр цикла

printf("Введите натуральное число : "); scanf("%lu",&a);

n=(a>255)?(a>65535)?31:15:7; // объясните это выражение

printf("Его двоичное представление: ");

for (i=n;i>=0;i–)

{

printf("%d",(a>>i)&0x1); // объясните выражение (a>>i)&0x1

if (i%8==0) //объясните, для чего служит эта операция

printf(" ");

}

printf("\n");

getch();

}

Результат выполненой работы:

1) если a<255, то для хранения числа требуется 1 байт памяти;

если a<65535, то для хранения числа требуется 2 байты памяти;

если a<2147483647, то для хранения числа требуется 4 байта памяти

2)проверяет в каком диапазоне находится число

3)

4) переводит в 8-ую сс

5.Постановка задачи:

Выполните программу и объясните результат КАЖДОЙ операции.

Код программы:

#include<stdio.h>

#include<conio.h>

void main()

{

int y,x,z,k;

x=03; y=02; z=01;

k=x|y&z; printf("1. %d\n",k); /\* Операции 1 \*/

k=x|y&~z; printf("2. %d\n",k); /\* Операции 2 \*/

k=x^y&~z; printf("3. %d\n",k); /\* Операции 3 \*/

k=x&y&&z; printf("4. %d\n",k); /\* Операции 4 \*/

x=1; y=-1;

k=!x|x; printf("5. %d\n",k); /\* Операции 5 \*/

k=-x|x; printf("6. %d\n",k); /\* Операции 6 \*/

k=x^x; printf("7. %d\n",k); /\* Операции 7 \*/

x<<=3; printf("8. %d\n",x); /\* Операции 8 \*/

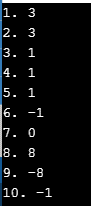
y<<=3; printf("9. %d\n",y); /\* Операции 9 \*/

y>>=3; printf("10. %d\n\n",y); /\* Операции 10 \*/

getch();

}

Результат выполненной работы:



Операции 1.

x=03; y=02; z=01; k=x|y&z;

k=(x|(02&01)); (вследствие приоритетов операций)

02&01  102&012  002

k=(x|00);

03|002  112|002  112

Результат: 03

Операции 2.

x=03; y=02; z=01; k=x|y&~z

k=(x|(02&(~01)));

k=(x|(02&10)); (вследствие приоритетов операций)

02