ПЯВУ. Лекция 8.

Основы программирования.

А.М. Задорожный

Контрольные вопросы

1. Что означает термин “Сортировка” в информатике?

2. Зачем применяется сортировка?

3. От чего и как зависит количество операций при сортировке пузырьком?

Содержание

1. Алгоритм поиска НОД (алгоритм Евклида)
2. Алгоритмы задач по простым числам (“решето” Эратосфена)
3. Подсчет количества единиц в байте
4. Приемы отладки программ

Контрольные упражнения. Целые

```
int n = 1, i = 0;
while (n != 0)
{
        i++;
        n <<= 1;
}
Console.WriteLine(i);
```

1. Что означает операция <<=? Опишите действия в теле цикла.
2. Что делает эта программа? Почему цикл закончится?
3. Что означает число, которое будет выведено на консоль программой?
4. Чему оно будет равно?

Контрольные упражнения. Целые

Что изменится, если заменить **!=** на **>**?

```
int n = 1, i = 0;
while (n > 0)          // Было !=
{
        i++;
        n <<= 1;
}
Console.WriteLine(i);
```

Контрольные упражнения. Числа с плавающей запятой (точкой)

```
double b = 1, eps = 1;
int i = 0;
while (b != b+eps)
{
        i++;
        eps /= 2;
}
Console.WriteLine(i);
```

1. Опишите действия в теле цикла программы?
2. Почему цикл закончится?
3. Что означает появившееся число?

Контрольные упражнения. Числа с плавающей запятой*

```
double b = 1;
int i = 0;
while (b != b*2)
{
        i++;
        b *= 2;
}
Console.WriteLine(i);
```

1. Опишите действия в теле цикла программы?
2. Почему цикл закончится?
3. Что означает появившееся число?

Алгоритм вычисления НОД. (Алгоритм Евклида)

Наибольший **О**бщий **Д**елитель двух натуральных чисел (x, y).

1. Если x == y, то NOD(x, y)==x==y.
2. Если x > y, то NOD (x, y) == NOD (x-y, y)
3. Если x < y, то NOD (x, y) == NOD (x, y-x)
4. Уменьшая большее из чисел на меньшее, мы обязательно придем к п. 1.

Алгоритм Евклида

```
while (x != y)
{
        if (x > y)
                x -= y;
        else
                y -= x;
}
```

Почему нельзя:

```
while (x != y)
{
        if (x > y)
                x %= y;
        else
                y %= x;
}
```

?

Выяснить, является ли число простым

```
// Входные данные – N

bool isPrimary = true;   // Пока считаем, что делителей нет
for (int i = 2; i < N; i++)     // Лучше i <= Math.Sqrt(N)
{
        if (N % i == 0)
        {
            isPrimary = false;
            break;
        }
}
```

Алгоритм поиска простых чисел меньших натурального N

“Решето Эратосфена”

1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13

1, **2**, 3, ~~4~~, 5, ~~6~~, 7, ~~8~~, 9, ~~10~~, 11, ~~12~~, 13

1, 2, **3**, ~~4~~, 5, ~~6~~, 7, ~~8~~, ~~9~~, ~~10~~, 11, ~~12~~, 13

1, 2, 3, ~~4~~, 5, ~~6~~, 7, ~~8~~, ~~9~~, ~~10~~, 11, ~~12~~, 13

1, 2, 3, ~~4~~, **5**, ~~6~~, 7, ~~8~~, ~~9~~, ~~10~~, 11, ~~12~~, 13

Алгоритм поиска простых чисел меньших натурального N

```
int N = 5;
bool[] prim = new bool[N + 1];        // Где алгоритм?

for (int i = 0; i < prim.Length; i++)
       prim[i] = true;

for (int i = 2; i < prim.Length; i++)
{
              if (prim[i])
              for (int j = 2 * i; j < prim.Length; j += i)
                     prim[j] = false;
}

for (int i = 2; i < prim.Length; i++)
       if (prim[i])
              Console.Write("{0},", i);
```

Контрольные вопросы

Какую задачу решают:

1. Алгоритм подсчета количества (элементов, удовлетворяющих условию);
2. Алгоритм нахождения суммы (элементов);
3. Алгоритм поиска наибольшего/наименьшего;
4. Алгоритм вычисления среднего арифметического;
5. Алгоритм поиска условно наибольшего/наименьшего;
6. Алгоритм вычисления условной суммы;
7. Алгоритм поиска заданного элемента массива;
8. Алгоритм подсчета количества слов в строке;
9. Алгоритм сортировки массива;
10. Алгоритм поиска НОД;
11. Решето Эратосфена;

Подсчет количества 1 в байте

```
//Входные данные byte x

int N = 0;
for (int i = 1; i < 255; i *= 2) // i <<=1;
{
        if ((i & x) != 0)
            N++;
}
```

Подсчет количества 1 в байте

```
//Входные данные byte x

int N = 0;
for (int i = 1; i < 255; i *= 2) // i <<=1;
{
        if ((i & x) != 0)
            N++;
}

//Быстрее:
int [] n = new int[256]{0,1,1,2,1,2,2,3,…};

int N = n[x];
```

Как посчитать количество 1 в целом?

- Массив целых займет всю память.
- Загрузка программы будет очень долгой.
- Заранее вычислить все значения не представляется возможным

Как посчитать количество 1 в целом?

```
int [] n = new int[256]{0,1,1,2,1,2,2,3,…};

uint x = ….

int N = 0;
byte t = (byte) x;
N += n[t];
t = (byte)(x>>8);
N += n[t];
t = (byte)(x >> 16);
N += n[t];
t = (byte)(x >> 24);
N += n[t];
```

Как посчитать количество 1 в целом? Оптимизация

```
int [] n = new int[256]{0,1,1,2,1,2,2,3,…};

uint x = ….

int N = 0;
while(x != 0)
{
    N += n[(byte)(x)];
    x >>=8;
}
```

Отладка программ

1. Вывод промежуточных контрольных значений
2. Средства Visual Studio:
    1. Точки остановки
    2. Выполнение до следующей точки остановки (F5)
    3. Наблюдаемые величины (контрольные значения)
    4. Пошаговое исполнение (F10)
    5. Переход к исполнению метода (F11)

Отладка программ. Вывод контрольных значений

Console.WriteLine(…);

Преимущества:

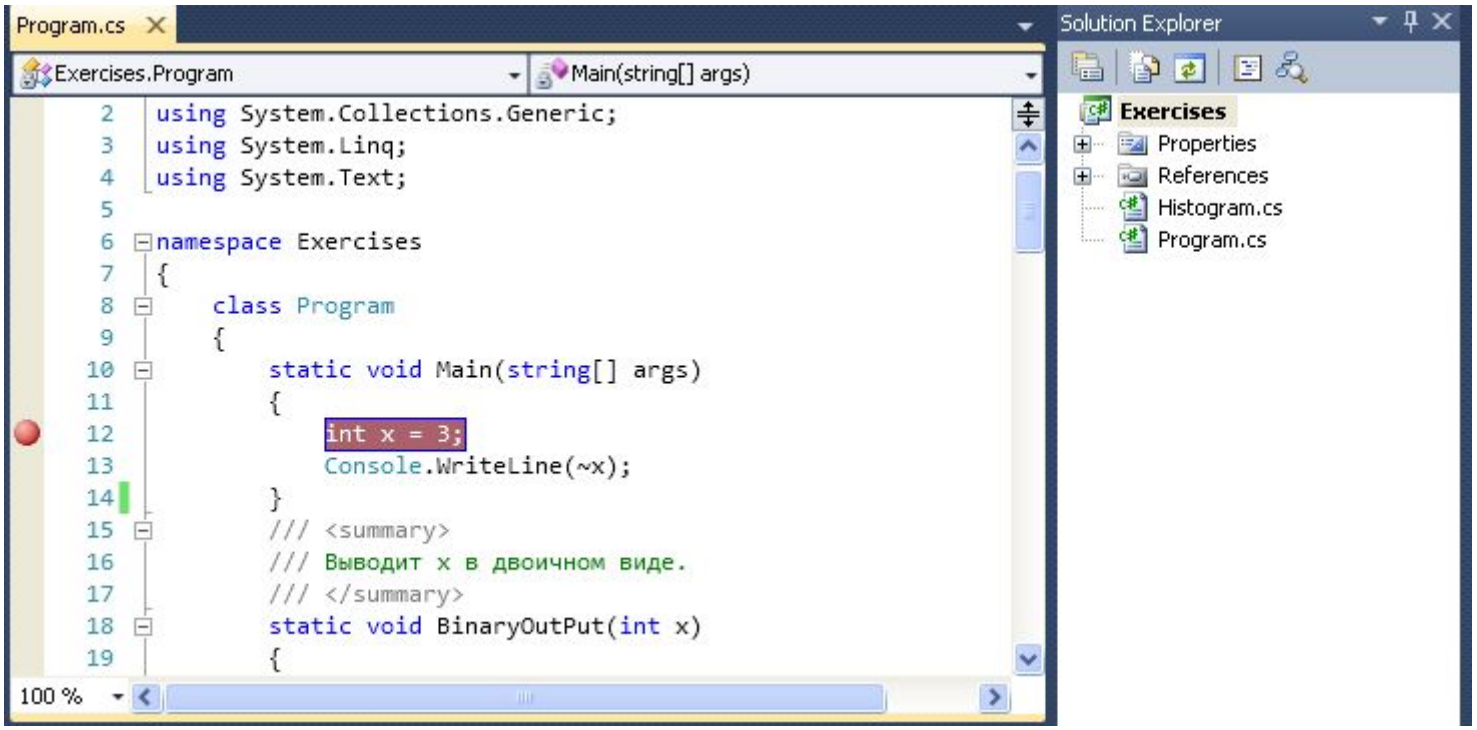
1. Универсальность

Недостатки:

1. Нужно изменять текст программы
2. Если выведено много, то трудно понять
3. Если нужно посмотреть дополнительную информацию, то придется изменить программу и выполнить заново.

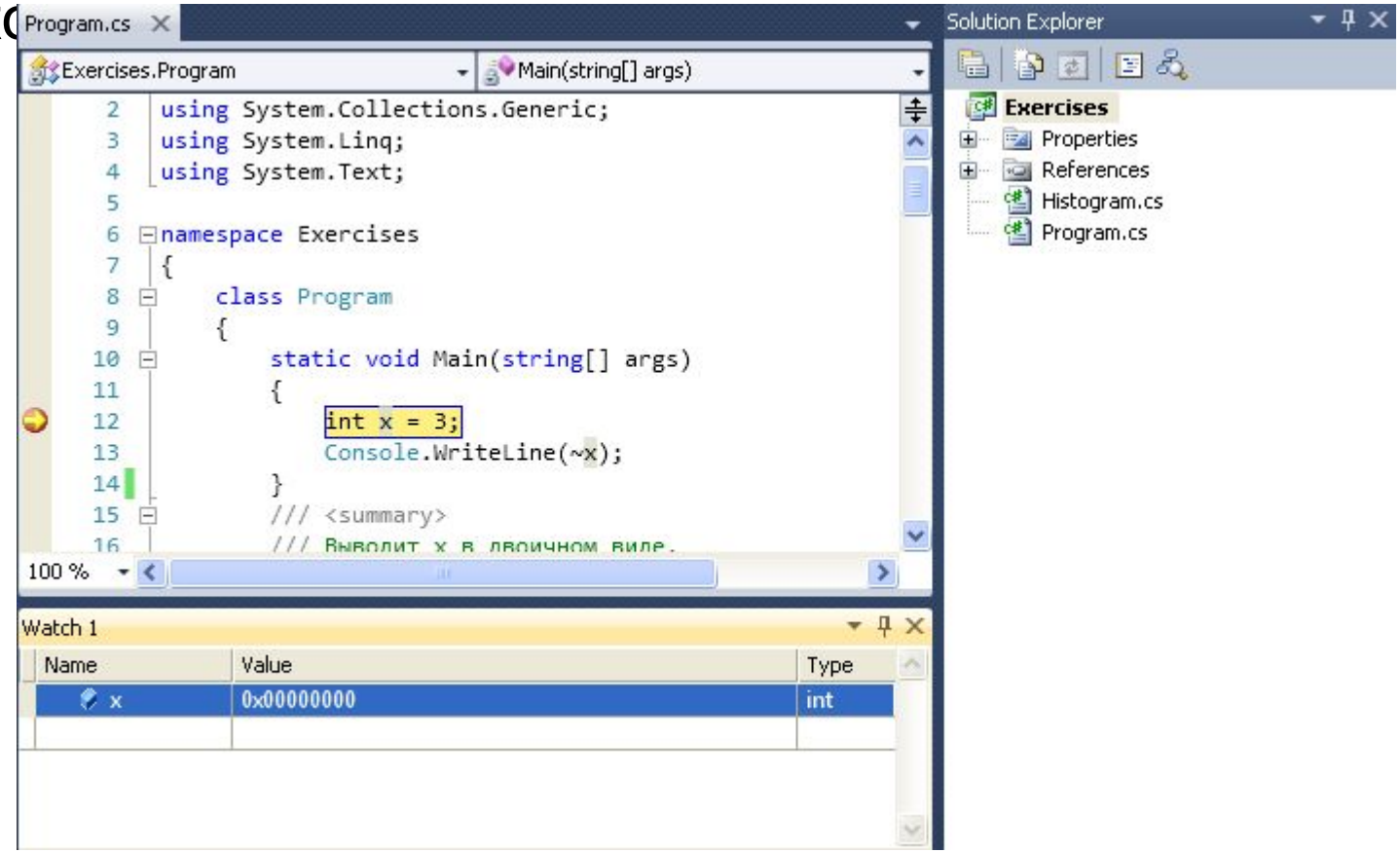
Отладка программ. Точки остановки

Добавление и удаление точки остановки



Отладка программ. Пошаговое исполнение

F5 – начать исполнение программы и остановиться в первой точке остановки. F10 – переход к следующей ко



Отладка программ. Контрольные значения

Добавление:

Выделить в тексте программы, когда она остановлена в одной из точек остановки, любое выражение и через контекстное меню указать: “Добавить контрольное значение”

Удаление:

В окне контрольных значений удалить ненужные значения

Отладка программ. Выполнение до нужной строки

Часто пошаговое выполнение требует много времени. Целесообразно пропустить часть остановок.

Если программа находится в режиме отладки, то выполнить ее до заданной строки (строки где установлен курсор) можно Ctrl + F10.