Увод в програмирането

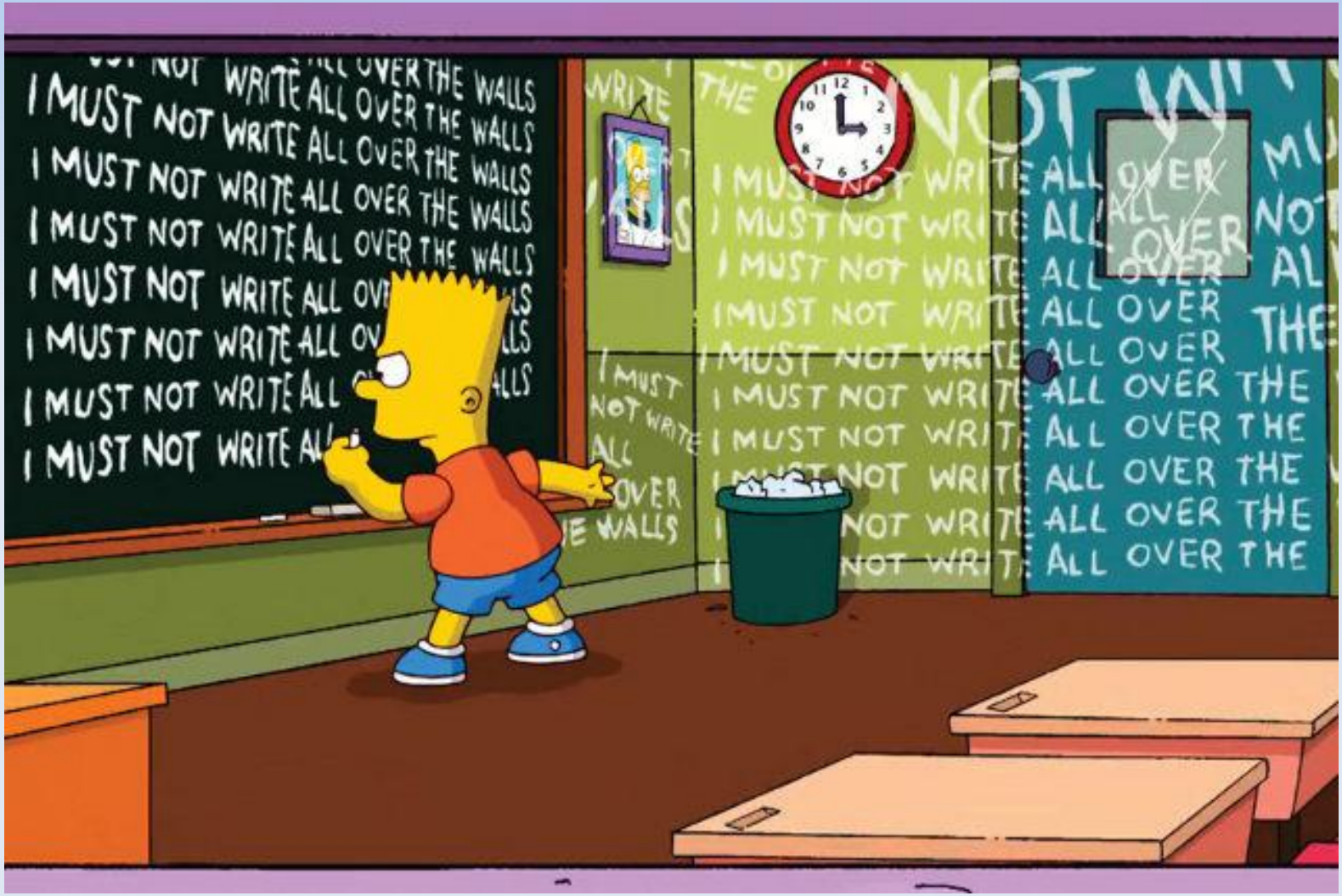
Лекция 4:

Цикли (първа част)

Преговор

- if (<условие>) <оператор> [else <оператор>]
- <условие> ? <израз> : <израз>
- switch (<израз>) {
  case <израз>: {<оператор>} [break;]
  case <израз>: {<оператор>} [break;]
  …
  [default: {<оператор>} [break;]
  };

Повтарящи се операции



- С познатите до момента средства не можем да напишем нещо по-умно от следното:

```
cout << "I must not write all over the walls\n";
cout << "I must not write all over the walls\n";
cout << "I must not write all over the walls\n";
cout << "I must not write all over the walls\n";
cout << "I must not write all over the walls\n";
```

- С новия материал вече ще можем:

```
for (int i = 0; i < n; i++)
  cout <<
    "I must not write all over the walls\n";
```

Циклични процеси - дефиниции

- Изчислителен процес, при който оператор или група оператори се изпълняват многократно за различни стойности на техни параметри, се нарича *цикличен*
- Два вида:
  - Броят на повторенията е известен предварително
    - Пример: смятане на *n*! (факториел)
  - Броят не е известен предварително, а зависи от условие
    - Пример: въвеждат се числа от клавиатурата до стигане на отрицателно
- *Итерация*

Оператори за цикъл в C++

- Ще разгледаме последователно следните оператори:
  - for
  - while
  - do-while

for

Синтаксис на оператор for

- for (<инициализация>; <условие>; <корекция>)

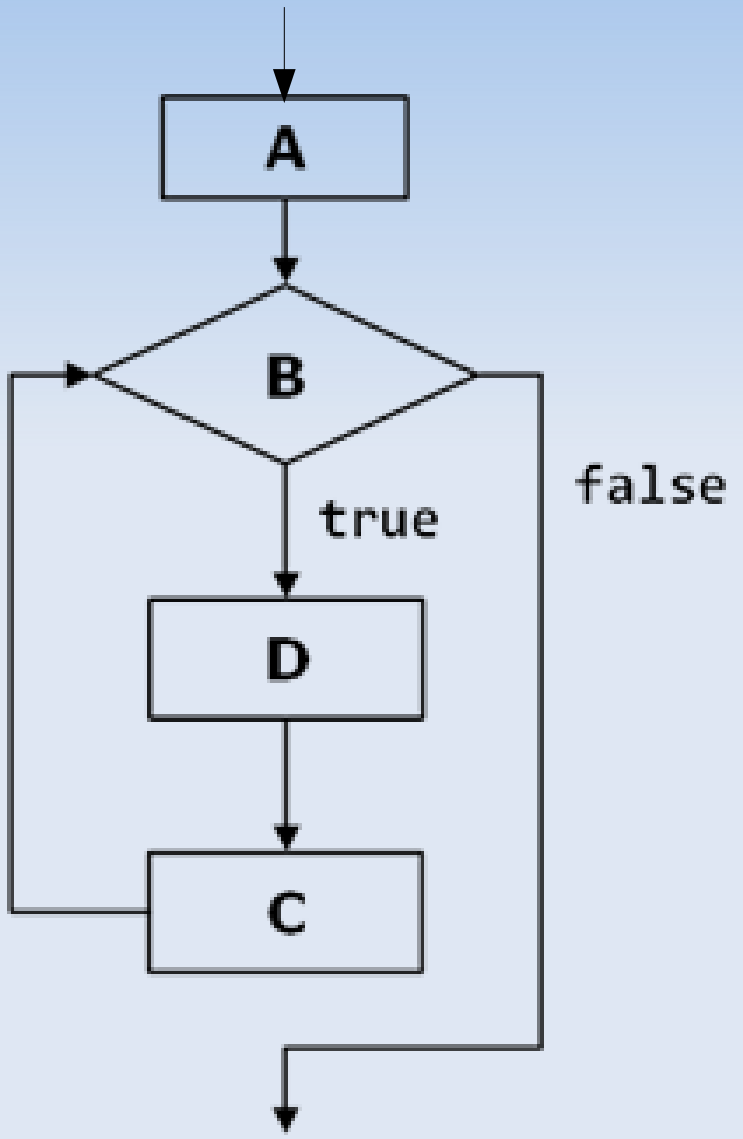
  <оператор>

- for – запазена дума
- <инициализация> – код, който се изпълнява еднократно в началото, напр. дефиниция с инициализация на променливи – int i = 0, j = 1
- <условие> – булев израз
  - по-скоро такъв, чийто резултат може да се преобразува до bool, например int (както и при if)
- <корекция> – един или няколко оператора, разделени със ”,” и незавършващи с ”;”
- <оператор> – *тяло* на цикъла – точно един произволен оператор

Семантика на for

```
for (A; B; C)
{
    D;
}
```

```
for (int i=0; i<10; i++)
{
    /* loop body */
}
```



Примери с for (1)

- ```
  for (int i = 1; i <= 10; i = i + 1)
      cout << i << " ";
  ```

  Резултат: 1 2 3 4 5 6 7 8 9 10

- ```
  for (int i = 1, x = 0, y = 1;
       i < 5; i = i + 1, y = y * x) {
      x = x + i;
  }
  ```

- Факториел

Съкратени оператори за присвояване

- Вместо iAmAVeryLongName = iAmAVeryLongName + 5 можем да напишем iAmAVeryLongName += 5
  - `+= -= *= /= %=`
- Вместо c = c + 1 можем да напишем ++c и c++
  - `++ --`
- Разлика между ++a и a++:
  - ```
    int a = 5, b; b = ++a; cout << a << b; // 66
                  // a = a + 1; b = a;
    ```
  - ```
    int a = 5, b; b = a++; cout << a << b; // 65
                  // b = a; a = a + 1;
    ```

Примери с for (2)

- Сумата от четните и произведението на нечетните числа, принадлежащи на интервала [*a*, *b*], където *a* и *b* са дадени цели числа (*a* < *b*)
- Първите *n*+1 члена на рекурентната редица:
  $a_0 = 1;$
  $a_i = i.a_{i-1} + 1/i,$ за $i = 1, 2, \ldots, n$

Бележки (следващи от дефиницията)

- Тялото се изпълнява, докато условието е изпълнено (true).
- Възможно е тялото да не се изпълни нито веднъж – ако още в началото условието е false
- Ако искаме в тялото да се изпълнява повече от един оператор, използваме блок (както при if)
- Всяка от 4-те части на for може да бъде празна, напр. for (; i < n;);
  - Ако пропуснем условието и не предвидим друг начин за прекратяване, цикълът е безкраен

while

Синтаксис на оператор while

- while (<условие>) <оператор>
- while – запазена дума
- <условие> – булев израз
- За условието и оператора важат същите бележки като при for

Семантика на оператор while

- Пресмята се стойността на <условие>
- Ако тя е false, изпълнението на оператора while завършва, без да се е изпълнило тялото му нито веднъж
- В противен случай, изпълнението на <оператор> и пресмятането на стойността на <условие> се повтарят, докато <условие> е true

Примери с while

- Задачи с обхождане на цифрите на цяло положително число (в 10-ична / *k*-ична бройна система)
  - Брой цифри
  - Сума от цифрите на число
  - Брой срещания на цифрата *d* в записа на число
  - Най-малката цифра на число

do-while

Синтаксис и семантика

- Синтаксис:
  - do <оператор> while(<условие>);
  - do също е запазена дума
- Семантика:
  1. Изпълнява се тялото
  2. Проверява се условието
  3. Ако то не е изпълнено, се прекратява операторът за цикъл
  4. В противен случай се връщаме на стъпка 1
- При while тялото може да не се изпълни нито веднъж, докато при do-while то се изпълнява поне веднъж

Пример

- Въвеждане на стойност от потребителя с проверка за коректност – ако потребителят въведе грешна стойност, да бъде подканян да я въведе отново, докато не стане коректна

Избор на подходящ оператор за цикъл (1)

- С всеки от разгледаните три оператора (for, while, do-while) може да се реализира всичко, което може да се напише и с другите два
- Всеки от операторите обаче е по-подходящ в определени ситуации:
  - for – обикновено се използва за цикли с известен брой итерации
  - while – цикли с условие, което се проверява преди всяка стъпка (цикъл с предусловие)
  - do-while – цикли с условие, което се проверява след всяка стъпка

Избор на подходящ оператор за цикъл (2)

- Задача: да преобразуваме всеки от следните три оператора в останалите два:
- `for (k = 0; k < 10; k++) cout << k;`
- `do cin >> k; while (k < 1 || k > 10);`
- `while (k > 5) k = k - 5;`
- Ще видим, че при избор на нетипичен за целта оператор за цикъл може да се получи по-труден за четене код

Вложени цикли (nested loops)

- Очаквано: след като тялото на един цикъл е оператор, този оператор може да бъде друг цикъл (от същия или друг тип)
- Колко реда ще се отпечатат на екрана?

```
for (int i = 0; i < 50; i++)
    for (int j = 0; j < 5; j++)
        cout << "I love you" << endl;
```

Вложени цикли - примери

- Да се отпечата таблицата за умножение
- Сума от факториелите на числата от 1 до *n*

$$\sum_{i=1}^{n} i!$$

  - С вложени цикли
  - Оптимизиран вариант без вложени цикли

Допълнителен материал

- Искаме да напишем курсова работа / дипломна работа / научна публикация с много математически формули?
- TeX
- `\[\sum_{i=1}^{n}{i!}\]`

Област на променливите

Област на видимост на променлива

- Областта на видимост на една променлива:
  - започва от нейната декларация
  - продължава до края на блока (оператора), в който е декларирана

```
if (1 < 2) {
    cout << x; // грешно – x още не е декларирана
    double x = 548918.543095849306; // дефиниция
    cout << x;
    for (int i = 0; i < 3; i++) {
        cout << x;
    }
}
cout << x; // грешно – блокът на x е завършил
```

Област на x

Област на променливи и оператори за цикъл

- ```
  for (int i = 0; …) { cout << i << " "; }
  cout << i; // по стандарт i вече не се вижда
  ```
- ```
  do {
      int x; cin >> x;
  } while (x < 0); // грешно – x е видима само в
  // тялото на цикъла
  ```

Променливи с еднакви имена

```
{
    int a = 1;
    double a = 8.2; // грешно – вече
                    // има a в същия блок!
    {
        cout << a; // 1
        int a = 2; // изненада
        cout << a; // 2, скрива външното a
    }
    cout << a; // отново 1
    int a = -24830; // грешно – вече
                    // има a в същия блок
}
```

- Първо контролно на Информатика – кога?
- Първо контролно на ИС – кога?
- Ако са дистанционни, подгответе се според инструкциите в Moodle

Обобщение и въпроси

- for, while, do-while
- +=, -=, …, ++, –
- Област на променлива