УДК 371.68:004
О.М.Кривонос, П.Г.Шевчук.
Інститут інформаційних технологій і засобів навчання НАПН України

ЗНАЧЕННЯ СТИЛЮ ПРОГРАМУВАННЯ В ПРОЦЕСІ НАВЧАННЯ УЧНІВ ТА СТУДЕНТІВ

У статті розглядається стиль програмування, який використовується в навчальному процесі.

Ключові слова: програмування, навчання, стиль.

Одне з важливих завдань, що ставиться перед викладачами навчальних закладів це навчити учня або студента працювати в колективі, працювати разом з колективом узгоджено та ефективно. Колективна діяльність в навчанні включає вміння спільно вирішувати навчальні задачі, створювати результати спільної діяльності та брати за них відповідальність. У процесі навчання програмування залучити учнів до колективної діяльності можна під час виконання групового завдання зі створення багатомодульного програмного продукту. При виконанні групового завдання виникає необхідність дотримання єдиного стилю написання(оформлення) програмного коду. Дотримання єдиного стилю потрібне програмісту ще й для власної зручності в процесі подальшого удосконалення програми.

Методику навчання програмування детально досліджувалася науковці: В.Ю. Биков, Ю.О. Дорошенко, М.І. Жалдак, І.О. Завадський, Н.В. Морзе, Ю.С. Рамський, С.М. Сайдаметова, С.О. Семеріков, О.М. Спірін. Проте практично відсутні дослідження значення стилю написання коду комп'ютерних програм в процесі навчання програмування.

Під *стилем програмування* розуміють набір методів, прийомів або правил, що забезпечують створення програмістами правильних, ефективних та зрозумілих для інших програм [4]. Правила стилю програмування можна уподібнити до правил етикету адже вони також є результатом колективного досвіду. Головне завдання програмного стилю: програма повинна бути зрозумілою тим, хто її використовує та супроводжує. Н. Вірт зазначає, що програма не несе користі, якщо користувач не має змоги розібратися в ній [2].

Досвід навчання програмування майбутніх викладачів інформатики в Житомирському державному університеті імені Івана Франка підтверджує доцільність використання наступних способів підвищення зрозумілості програмного коду, а саме – код повинен містити: коментарі, табуляцію, зрозумілі ідентифікатори, певне розміщення операторів, дужки та відступи. Ці висновки базуються на роботах науковців [1, 5], які в свою чергу пропонують наступні ознаки якісного програмного стилю: прозора логіка, природні вирази, зрозумілі імена змінних, відповідне форматування, розгорнуті коментарі, відсутність нестандартних конструкцій.

Коментарі – це надзвичайно ефективний засіб зробити програму більш зрозумілою. «Некоментована програма – це найгірша помилка, яку може зробити програміст, а також ознака дилетантського підходу» [4]. Коментарі до програмного коду слід писати відразу під час написання програми. Саме так, швидше за все, можна забезпечити коментування усіх без винятку суттєвих структур алгоритму. В процесі створення коментарів доцільно дотримуватися правила «не кількість, а якість» – коментарі повинні не нагромаджувати програмний код а допомагати його розуміти. Розрізняють наступні види коментарів: коментарі призначення програми або підпрограми, коментарі призначення змінних, коментарі опис вводу-виведення, коментарі опис методу, коментар по об'єму пам'яті, коментар відомості про авторів, коментар дати внесення останніх змін до програмного коду.

Ідентифікатори (імена змінних, констант) повинні бути вдало підібрані, щоб бути зрозумілими при читанні програмного коду. Вдало поіменований ідентифікатор може замінити використання відповідного коментаря. Для ідентифікаторів використовують загально прийняті або легко зрозумілі позначення (наприклад: max – максимальне значення, min – мінімальне значення, i, j – індекси, temp – проміжне значення, sum – сума, S – площа, P – периметр тощо). Суворо не рекомендується використовувати схожі ідентифікатори (rez і res, або S0 та So). За загальноприйнятою домовленістю імена змінних починаються з малої літери, імена класів, методів – з великої літери, імена констант повністю складаються з великих літер. Крім того існує домовленість написання імен, що складаються з кількох слів, адже використанні в ідентифікаторі

проміжків(пробілів) не допускається. Так багатослівні імена констант розділяються підкресленням, а для імен змінних і класів кожне слово в багатослівному імені може починатися з великої букви [7].

Певні правила існують щодо **розміщення операторів в тексті програми**. Візуальне форматування коду, як засвідчує Дж. Макконелл [6], показує логічну структуру програми. Форматування виконується відступами, пропусками та табуляцією. Незважаючи на те, що переважна більшість мов програмування дозволяє розміщення декількох операторів в одному рядку рекомендується використовувати окремий рядок для кожного окремого оператора. В той же час окремі вирази бажано повністю розміщувати в одному рядку. Якщо частина виразу переноситься на наступний рядок, то знак бінарної операції повинен лишитися в попередньому рядку. Пропонується відокремлювати бінарні оператори від операндів одним пропуском.

Операторні дужки також відіграють важливу роль в стилі програмування. Як правило, їх виносять на наступний рядок відносно конструкції, що до якої ці дужки використовуються. А вміст скадового оперетора слід розташувати правіше відносно операторних дужок:

```
For i := 1 to N do
 Begin
   A := sin(5 * i) + sqrt(1 – sqr(cos(2 * i)));
   a := 1 / a;
   S := S + a
 End;
```

Ряд фахівців, зокрема Л.В. Гришко [3], пропонують певні рекомендації щодо написання умовного оператора. Стандартну форму умовного оператора If <умова> then <оператор1> else <оператор2> пропонується подати наступним чином:

- якщо умову не вдається розмістити в одному рядку, то продовження повинне починатися під першим символом виразу умови першого рядка;
- серія операторів <оператор1> повинна бути розташована з нового рядка під оператором then;
- серія операторів <оператор2> повинна бути розташована з нового рядка під оператором else;
- оператори then та else, в сою чергу, повинні бути записаним з нового рядка напроти оператора If.

```
If  <початок умови
    продовження умови >
then
<серія операторів 1>
else
< серія операторів 2>
```

Службові слова, якими починається та закінчується той чи інший оператор, слід записувати на одній вертикалі. Вкладені структури (цикли, розгалуження, складовий оператор) повинні розташовуватись з нового рядка і правіше відносно конструкцій, що їх використовують. Корисно також для позначення вкладених операторів застосовувати коментарі.

Дотримання студентом або учнем цих простих, на перший погляд, правил написання програмного коду сприяє навчанню програмування. В окремих випадках, дотримання учнем чіткого стилю форматування дозволяє стверджувати, що він, правильно описав алгоритм та написав програму, розуміє суть структурного чи то об'єктно-орієнтованого програмування, намагається дотримуватись правил колективної роботи над проектом. В той же час важко повірити в те що, не документований, без дотримання стилю код може бути коректним і стійким, що його можна супроводжувати і вносити необхідні зміни, зберігаючи коректність і стійкість.

Висновки

Фахівці з розробки програмного забезпечення різних фірм та організацій по усьому світі спільно сформували та розвивають систему правил, що допомагають зручніше працювати з текстом комп'ютерних програм а також організовувати колективне написання різних їх частин. В програмуванні традиційно виробилися певні стилі дотримання яких є ознакою високої фахової компетенції та культури програміста.

Колективне написання програмного коду можна успішно використовувати в навчанні, для розвитку учнівського колективу. Кожне з правил, що забезпечують дотримання стилю програмування дозволяє краще розкрити та пояснити ті чи інші сторони процесу написання

програм. Навчання стилю програмування полегшує процес навчання учня (студента) допомагає йому швидше сприймати та розуміти алгоритми комп'ютерних програм, підвищує загальну культуру мислення та діяльності. Навчання стилю доцільно застосовувати в усіх випадках навчання програмування.

1. Ален И. Голуб С и С++. Правила программирования / Ален И. Голуб // – М.: БИНОМ, 1996. – 272 с.
2. Вирт Н. Программирование на языке Модула-2 / Н. Вирт [пер. с англ.] // – М.: Мир, 1987. – 224 с.
3. Гришко Л.В. Навчання стилю програмування, як складова формування професійної культури майбутнього інженера-програміста // Вісник Черкаського університету, серія "Педагогічні науки". Випуск 143. – Черкаси, 2009. – С. 37-43.
4. Д. Ван Тассел. Стиль, разработка, эффективность, отладка и испытание программ / Д. Ван Тассел // – М.: Мир, 1985. – 332 с.
5. Керниган Б.В., Пайк Р. Практика программирования / Б.В. Керниган, Р. Пайк [пер. с англ.] // – М.:Вильямс, 2003 – 381 с.
6. Макконелл Дж. Анализ аолгоритмов. Вводный курс / Дж. Макконелл // – М.: Техносфера, 2002. – 304 с.
7. Биллиг В.А. Объектное программирование в классах на C# 3.0. Лекция: Корректность и устойчивость программных систем. / В.А. Биллиг. Объектное программирование в классах на C# 3.0. // Интернет-Университет Информационных Технологий [Електронний ресурс] — 30.01.2011. — Режим доступу: http://www.intuit.ru/department/pl/toopincsharp30/10/6.html