УДК 004.4:378.147

Галина КОВТОНЮК,
*кандидат педагогічних наук,
старший викладач кафедри математики
та інформатики
Вінницького державного педагогічного
університету імені Михайла Коцюбинського*

ДО ПИТАННЯ ВИВЧЕННЯ СТРУКТУР ДАНИХ У КУРСІ ІНФОРМАТИКИ МАЙБУТНІМИ ВЧИТЕЛЯМИ ФІЗИКО-МАТЕМАТИЧНИХ ДИСЦИПЛІН

Статтю присвячено методичним аспектам вивчення структур даних у курсі інформатики, що необхідні для формування інформатичної компетентності майбутніх учителів фізико-математичних дисциплін. Наведено класифікацію структур даних за різними ознаками. Подано приклад розробки лабораторної роботи, яка стосується такої динамічної структури даних, як черга. Запропоновано зразки програм із опрацювання черг. Наочно продемонстровано методи опрацювання такої структури даних для кращого розуміння і засвоєння її змісту.

Ключові слова: *інформатична компетентність, черга, структури даних, підготовка вчителів фізико-математичних дисциплін.*

Статья посвящена методическим аспектам изучения структур данных в курсе информатики, которые необходимы для формирования информатической компетентности будущих учителей физико-математических дисциплин. Дана классификация структур данных по различным признакам. Приведен пример разработки лабораторной работы, которая касается такой динамической структуры данных, как очередь. Предложены примеры программ с обработки очередей. Наглядно продемонстрированы методы обработки такой структуры данных для лучшего понимания и усвоения ее содержания.

Ключевые слова: *информатическая компетентность, структуры данных, подготовка учителей физико-математических дисциплин.*

The article is devoted to the methodical aspects of studying data structures in the course of informatics, which is necessary for the formation of informational competence of future teachers of physical and mathematical disciplines. Classifications of data structures on various grounds are given. An example of the development of laboratory work, which relates to such a dynamic data structure as the queue, is given. It offers examples of programs with processing queues. To better understand and assimilate the content of such a data structure, the methods of its processing are clearly demonstrated.

Key words: *informatical competence, data structures, preparation of teachers of physical and mathematical disciplines.*

Постановка проблеми. Як зазначалося в попередньому нашому дослідженні [2, с. 50], «інформатична компетентність є надзвичайно важливою компетентністю, якою повинен володіти сучасний учитель фізико-математичних дисциплін для ефективної організації пізнавальної діяльності школярів. Ця компетентність дає змогу не лише розв'язувати інформаційні задачі за допомогою відомих програмних засобів, а й проектувати та створювати власні програмні продукти». Зважаючи на це, вивчення інформатики, зокрема основ алгоритмізації і програмування, у формуванні інформатичної компетентності відіграє провідну роль. При цьому одним із найважливіших розділів алгоритмізації і програмування є «Структури даних».

Аналіз наукових досліджень і публікацій. Окремим аспектам вивчення алгоритмізації і програмування у вищій школі присвячені роботи А. Гуржія, М. Жалдака, В. Клочка, Е. Кузнєцова, М. Львова, О. Миленького, В. Монахова, Н. Морзе, В. Петрушина, С. Ракова, Ю. Рамського, Д. Румянцева, С. Семерікова, О. Співаковського, О. Спіріна, Н. Шаховської та ін.

Проблеми підготовки майбутніх учителів фізико-математичних дисциплін за допомогою ІКТ та технології формування в них інформатичної компетентності досліджували М. Жалдак, В. Жукова, Н. Морзе, М. Рагуліна, Ю. Рамський, С. Раков, О. Семеніхіна та ін.

Попередньо проведене нами дослідження [4] дало змогу зробити висновок, що для ефективної організації самостійної пізнавальної діяльності школярів учитель фізико-математичних дисциплін повинен володіти ґрунтовними знаннями з інформатики.

Мета статті – розкрити деякі методичні аспекти вивчення структур даних у курсі інформатики майбутніми вчителями фізико-математичних дисциплін.

Виклад основного матеріалу. Д. Кнут зазначає, що для правильного використання комп'ютера необхідні ґрунтовні знання структурних взаємозв'язків між даними, основних методів представлення структур усередині комп'ютера, а також методи роботи з ними [1].

Варто зазначити, що вивчення структур даних, зокрема статичних і динамічних, є обов'язковим не тільки в курсах з інформатики, які вивчають майбутні вчителі фізико-математичних дисциплін, але й невід'ємним компонентом навчальних програм з інформатики для 10–11 класів у профільних класах і класах із поглибленим вивченням інформатики [6].

Отже, під *структурою даних* розуміють сукупність різноманітних структурованих типів даних. Розрізняють прості та складні структури даних. Прості структури даних не можуть ділитися на складові, більші ніж біти. Тобто означені структури даних (типи) є неподільними одиницями, причому для простого типу даних чітко визначений його розмір та спосіб розміщення в оперативній пам'яті. До них належать числові, символьні, логічні типи і вказівники. Водночас складні (інтегровані) структури даних складаються з інших структур даних (простих або складних).

Залежно від наявності зв'язків між окремими елементами структури даних розрізняють незв'язані структури (масиви, рядки, стеки, черги) та зв'язані структури (зв'язані списки).

Відповідно до можливостей зміни розміру [8] розрізняють статичні (масиви, рядки, множини, записи, таблиці) та динамічні структури даних (стеки, черги, дерева, зв'язані списки). Зауважимо, що досить часто за цією ознакою виділяють такі типи: прості, статичні, напівстатичні, файли. При цьому до напівстатичних структур відносять стеки, черги, деки і рядки [5; 7].

За ознакою впорядкованості виокремлюють лінійні (масиви, рядки, множини, стеки, черги, деки, однозв'язні і двозв'язні списки) та нелінійні структури даних (таблиці, дерева, багатозв'язні списки, сітки).

Усі вищеозначені структури даних мають значну кількість різноманітних особливостей і відповідних методів їх обробки. Не будемо розкривати суть кожної з них, лише зауважимо, що їх засвоєння студентами (а тим паче учнями) є доволі непростим і досить тривалим процесом, який вимагає від викладача інформатики підбору такої методики і створення відповідних методичних розробок, які б максимально спростили цей процес.

Зауважимо, що для студентів фізико-математичних спеціальностей Вінницького державного педагогічного університету нами були розроблені відповідні лекції, а також тексти лабораторних і практичних робіт, які можна знайти на персональному сайті закладу [3].

Для прикладу більш детально зупинимося на лабораторній роботі з теми «Реалізація черги в IDE Lazarus». Після подання теми і мети роботи наводяться короткі теоретичні відомості, зокрема такі: *черга* (англ. *queue*) визначається, як динамічна структура даних, що працює за принципом «*перший прийшов – перший пішов*» (англ. *FIFO – first in, first out*). У черги є «*голова*» (англ. *head*) та «*хвіст*» (англ. *Tail*). Зазначається, що елемент, який додається до черги, потрапляє в її «хвіст», а елемент, що видаляється з черги, опиняється в її «голові». Така черга аналогічна звичній нам магазинній, в якій хто перший зайняв місце, той першим і обслуговуватиметься *(див. рис. 1)*.

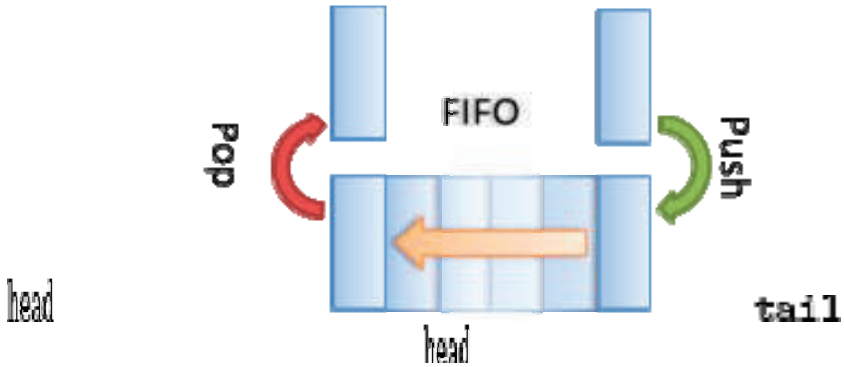

Рис. 1. Побудова черги

Після цього розглядаються операції із чергою:

1. Додати елемент у кінець черги (*Push*). Операція – додавання елемента у *«хвіст»* черги. При цьому довжина черги збільшується на одиницю. При спробі додати елемент у вже заповнену чергу відбувається її переповнення.

2. Видалити елемент із початку черги *(Pop)*. Операція – повернення елемента з *«голови»* та видалення його з черги, таким чином встановлюючи *«голову»* на наступний за видаленим елемент та зменшуючи довжину на одиницю. При спробі видалити елемент із порожньої черги виникає незаповнення.

Також зазначається, що чергу можна реалізувати за допомогою масиву a[1...n], в якому зберігають дані, та двох додаткових змінних – *head* і *tail*, в яких зберігають індекси відповідно «*голови*» та «*хвоста*» черги. Основні операції можна записати таким чином:

Поставити в чергу:
if tail=n **then** tail:=1;
else tail:= tail+1;
a[tail]:=x.

Отримати з черги:
x:=a[head];
if head=n **then** head:=1;
else head:=head+1.

Для реалізації черги за допомогою однозв'язного списку необхідно мати два вказівники – на «*голову*» (містить адресу початку черги) та «*хвіст*» (містить адресу кінця черги). Якщо черга порожня, то «*голова*» та «*хвіст*» мають значення **nil**.

Структура вузла:
type Pnode=^Node;
Node=record
data: integer;
next:Pnode;
end;
type queue=record // *Тип даних «черга»:*
head, tail: Pnode;
end.

Варто зауважити, що програми створення стека та черги відрізняються лише програмами запису нового елемента у відповідну структуру.

Після цього розглядається програма із коментарями *(див. рис. 2)*, яка формує чергу (заповнюється шляхом уведення цілих невід'ємних чисел), виводить вміст черги і суму її елементів, а пам'ять, зайнята її елементами, звільняється.

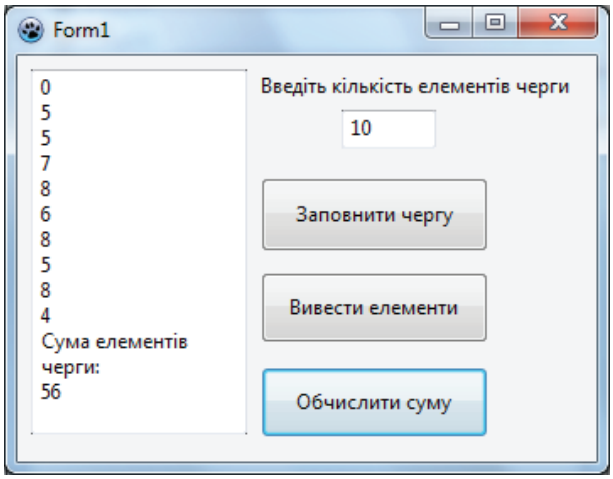

Рис. 2. Головна форма програми створення та опрацювання цілочислової черги

```
unit Unit1;
{$mode objfpc}{$H+}
interface
uses
Classes, SysUtils, FileUtil, Forms, Controls, Graphics, Dialogs, StdCtrls;
type
cherga =^cher;
cher = record
d: integer;
next : cherga;
end;
{ TForm1 }
TForm1 = class(TForm)
Button1, Button2, Button3: TButton;
Edit1: TEdit;
Label1: TLabel;
Memo1: TMemo;
procedure Button1Click(Sender: TObject);
procedure Button2Click(Sender: TObject);
procedure Button3Click(Sender: TObject);
procedure FormCreate(Sender: TObject);
private
{ private declarations }
public
{ public declarations }
end;
var
Form1:TForm1;
c_head,c_last,c: cherga;
i, n, sum: integer;
implementation
{$R *.lfm}
{ TForm1 }
procedure TForm1.FormCreate(Sender: TObject);
begin
// формування черги
new (c_head);
c_head^.next:= nil;
c_head^.d:= 0;
c_last:=c_head;
end;
procedure TForm1.Button1Click(Sender: TObject);
begin
n:=strtoint (edit1.text);
for i:= 1 to n do //заповнення черги n елементами
begin
new(c);
c^.next:=nil;
c_last^.next:= c;
c_last:=c;
c_last^.d:=random(10);
end;
end;
procedure TForm1.Button2Click(Sender: TObject);
begin
// виведення значень із черги:
sum:=0;
while c_head^.next<>nil do
begin
memo1.Lines.Add (inttostr (c_head^.d));
sum:=sum+c_head^.d;
c_head:=c_head^.next;
end;
dispose(c);
end;
procedure TForm1.Button3Click(Sender: TObject);
begin
memo1.Lines.Add ('Сума елементів черги:');
memo1.Lines.Add (inttostr (sum));
end;
end.
```

У другому прикладі (код цієї програми наводити не будемо) для більшої наочності і кращого сприйняття використовуються компоненти *TShape*, де наведено програму для формування черги. В означеній програмі при натисканні на кнопку «додати» додається елемент черги – одне слово з відомої мнемонічної фрази для запам'ятовування кольорів і з'являється фігура відповідного кольору *(див. рис. 3)*, а при натисканні на кнопку «вилучити» – вилучається *(див. рис. 4)*. Також програма знаходить суму елементів черги – відому мнемонічну фразу *(див. рис. 5)*.

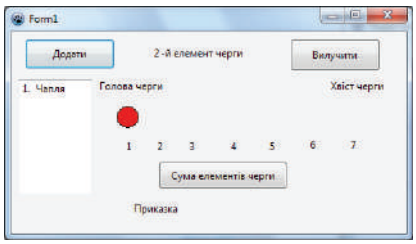

Рис. 3. Додавання елементів черги

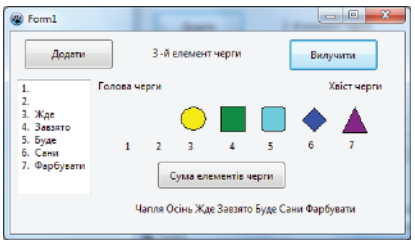

Рис. 4. Вилучення елементів черги

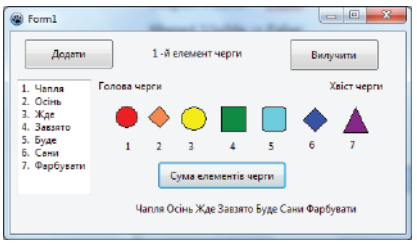

Рис. 5. Сума елементів черги

На нашу думку, вищенаведений приклад із візуалізацією виконання операцій над чергою сприяє кращому розумінню сутності черги і буде корисним як для студентів, так і для учнів.

Після теоретичних відомостей і прикладів студентам пропонується створити власний проект, зразок якого розглянемо нижче.

Проект «Черга». *Написати програму реалізації алгоритму з обов'язковими коментарями (програма реалізується в середовищі Lazarus, а її текст переписується в зошит). У роботі слід послуговуватися меню, в якому необхідно використати такі пункти:* ***«заповнити чергу», «вивести елементи», «обчислити», «про програму», «про автора»***.

Нижче подано 20 варіантів *індивідуальних завдань до проекту*:

1. Створити чергу цілих чисел. Знайти найбільший елемент черги.
2. Створити чергу цілих чисел. Знайти найменший елемент черги.
3. Створити чергу цілих чисел. Знайти середнє арифметичне елементів черги.
4. Створити чергу цілих чисел. Знайти суму додатних елементів черги.
5. Створити чергу цілих чисел. Знайти суму від'ємних елементів черги.
6. Створити чергу цілих чисел. Обчислити суму парних елементів черги.
7. Створити чергу цілих чисел. Обчислити суму непарних елементів черги.
8. Створити чергу цілих чисел. Обчислити суму елементів черги, які менші ніж 5.
9. Створити чергу цілих чисел. Обчислити суму елементів черги, які більші ніж 5.
10. Створити чергу цілих чисел. Обчислити кількість однозначних елементів черги.
11. Створити чергу цілих чисел. Обчислити кількість двозначних елементів черги.
12. Створити чергу цілих чисел. Обчислити кількість парних елементів черги.
13. Створити чергу цілих чисел. Обчислити кількість непарних елементів черги.
14. Створити чергу цілих чисел. Обчислити кількість додатних елементів черги.
15. Створити чергу цілих чисел. Обчислити добуток елементів черги.
16. Створити чергу символів. Визначити кількість голосних літер.
17. Створити чергу символів. Визначити кількість приголосних літер.
18. Створити чергу символів. Визначити кількість літер *а*.
19. Створити чергу символів. Визначити кількість літер *о*.
20. Створити чергу символів. З'ясувати, чи є в черзі слово «*студент*».

Висновок. Таким чином, вивчення структур даних у курсі інформатики є необхідним для формування інформатичної компетентності майбутніх учителів фізико-математичних дисциплін, а наочна демонстрація методів їх опрацювання сприяє кращому їх засвоєнню. Особливості вивчення цього розділу інформатики продемонстровано на прикладі вивчення черг, зокрема у статті наведено лабораторну роботу з прикладами та індивідуальними проектами. Такий підхід, на нашу думку, сприяє більш ефективному засвоєнню даного розділу зокрема і формуванню інформатичної компетентності загалом.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Кнут Д. Искусство программирования : в 4 т. / Д. Кнут ; пер. с англ. –Изд. 3-е. – М. : Вильямс, 2006. – Т. 1. – 682 с.

2. Ковтонюк Г. М. До питання формування інформатичної компетентності майбутніх учителів фізико-математичних дисциплін / Г. М. Ковтонюк // Нова педагогічна думка. – 2017. – Вип. 3(91). – С. 49–51.

3. Ковтонюк Г. М. Персональний сайт викладача як ефективний засіб організації самостійної пізнавальної діяльності майбутніх учителів фізико-математичних дисциплін / Г. М. Ковтонюк // Фізико-математична освіта. – 2017. – Вип. 4(14). – С. 205–208.

4. Ковтонюк Г. М. Формування професійної готовності майбутніх учителів фізико-математичних дисциплін до організації самостійної пізнавальної діяльності школярів: дис. ... канд. пед. наук : 13.00.04 / Г. М. Ковтонюк. – Вінниця, 2013. – 266 с.

5. Коротєєва Т. О. Алгоритми та структури даних : навч. посібник / Т. О. Коротєєва. – Львів : Вид. Львівської політехніки, 2014. – 280 с.

6. Начальні програми для 10–11 класів [Електронний ресурс]. – Режим доступу : https://mon.gov.ua/ua/osvita/zagalna-serednya-osvita/navchalni-programi/navchalni-programi-dlya-10-11-klasiv.

7. Ткачук В. М. Алгоритми і структура даних : навч. посібник / В. М. Ткачук. – Івано-Франківськ : Вид. Прикарп. нац. унів. ім. В. Стефаника, 2016. – 286 с.

8. Шаховська Н. Б. Алгоритми і структури даних : посібник для ВНЗ / Н. Б. Шаховська, Р. О. Голощук ; за заг. ред. В. В. Пасічника. – Львів : Магнолія, 2011. – 216 с.

Дата надходження до редакції: 05.08.2018 р.