ПРИЛОЖЕНИЯ НА СТРУКТУРАТА ОТ ДАННИ СТЕК

Галя Шивачева

Резюме: В доклада са представени начини за организиране на стек в статичен и динамичен вид и примери за тяхното приложение. Представената структура допълва известните от литературата източници като подобрява нагледността на материала и улеснява прилагането му за целите на обучението. Препоръчаният начин на работа с тази структура от данни води до повишаване на неговата ефективност при използване за обучение. Описаният начин за организация на динамичен стек и статичен такъв с използване на масив, дава възможност да се изследват основните операции с двата типа структури от данни и да се избегнат затрудненията при тяхното овладяване от обучаемите.

Ключови думи: Структура от данни, Стек, Масив

1. Увод

Във Факултет „Техника и технологии“ – Ямбол студентите специалност „Автоматика и компютърни системи“ изучават дисциплината „Алгоритми и структури от данни“. Учебната програма освен алгоритми включва и структурите от данни: масив, линеен едносвързан списък, двусвързан списък, стек, опашка, дек, дърво и граф. Структурата от данни стек намира множество приложения, което налага търсенето на нови, усъвършенствани начини за нейното усвояване от студентите [2,4,5].

Целта на доклада е да се представят

APPLICATIONS OF THE DATA STRUCTURE STACK

Galya Shivacheva

Abstract: The article presents ways to organize stacks in static and dynamic form and examples of their application. The presented structure complements the sources known from the literature by improving the visibility of the material and facilitating its application for training purposes. The recommended way of working with this data structure is to increase its performance in training. The described method of organization of dynamic and static stack using such an array, gives the opportunity to explore the basic operations with two types of data structures and to avoid difficulties in their assimilation from the students.

Keywords: Data structure, Stack, Array

1. Introduction

At the Faculty of Engineering and Technology - Yambol, students specializing in Automation and Computer Systems study the subject Algorithms and Data Structures.

In addition to algorithms, the curriculum also includes data structures: array, linear unrelated list, double-linked list, stack, queue, deck, tree, and graph.

The Data Structure stack finds multiple applications, requiring new, advanced ways to learn from students [2,4,5].

The aim of the report is to present the ways to organize a stack of data

начините за организиране на структура от данни стек като се предложат примери улесняващи усвояването й от обучаемите.

2. Статична реализация на стек с масив

Дефинира се един индекс, който сочи към върха на стека. При добавяне на нов елемент индексът се увеличава с единица и елементът се записва на върха на стека.

При премахване на елемент – проверява се дали стека не е празен и след това стойността на елемента се извлича и индексът се намалява с единица.

При добавяне на елемент трябва да се прави проверка за препълване на масива, което е основен **недостатък** на статичната реализация.

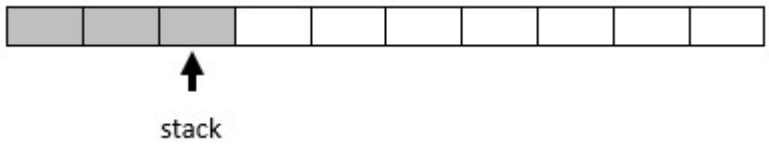
structure by offering examples facilitating its assimilation from learners.

2. Static realization of stack by array

The static realization of stack by array defines an index that points to the top of the stack. When adding a new item, the index is incremented by one, and the item is recorded at the top of the stack.

When removing an item, check if the stack is empty and then the value of the item is retrieved and the index is reduced by one.

When an element is added, an overflow check of the array must be performed, which is a major **drawback** of the static conversion.



Фиг.1. Статична реализация на стек с масив

Fig.1. Static realization of stack with array

3. Динамична реализация на стек

Дефинира се структура с две полета: стойност на елемент и указател към следващ елемент и функции, които реализират основните операции със структурата стек: добавяне на нов елемент, премахване на елемент, извличане стойността на елемент, проверка за празен стек и др. След това във функцията main се използват тези функции за реализиране на стек.

За всяка функция се илюстрира със схема как се променя структурата на стека за всеки от операторите.

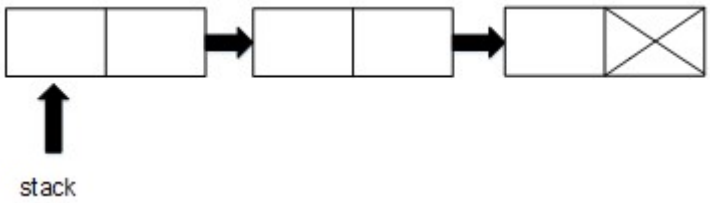
Например за добавяне на нов елемент в стек първо схематично се представя стек с три елемента (фигура 2).

3. Dynamical realization of stack

A two-array structure is defined: an element and a pointer value to the next element and functions that perform the basic stack operations: adding a new element, removing an element, extracting an element value, checking for an empty stack, and more. Then, in the main function, these functions are used to implement a stack.

For each function is illustrated in the diagram how to change the structure of the stack for each of the operators.

For example, to add a new element to a stack, a stack of three elements is schematically represented (Figure 2).

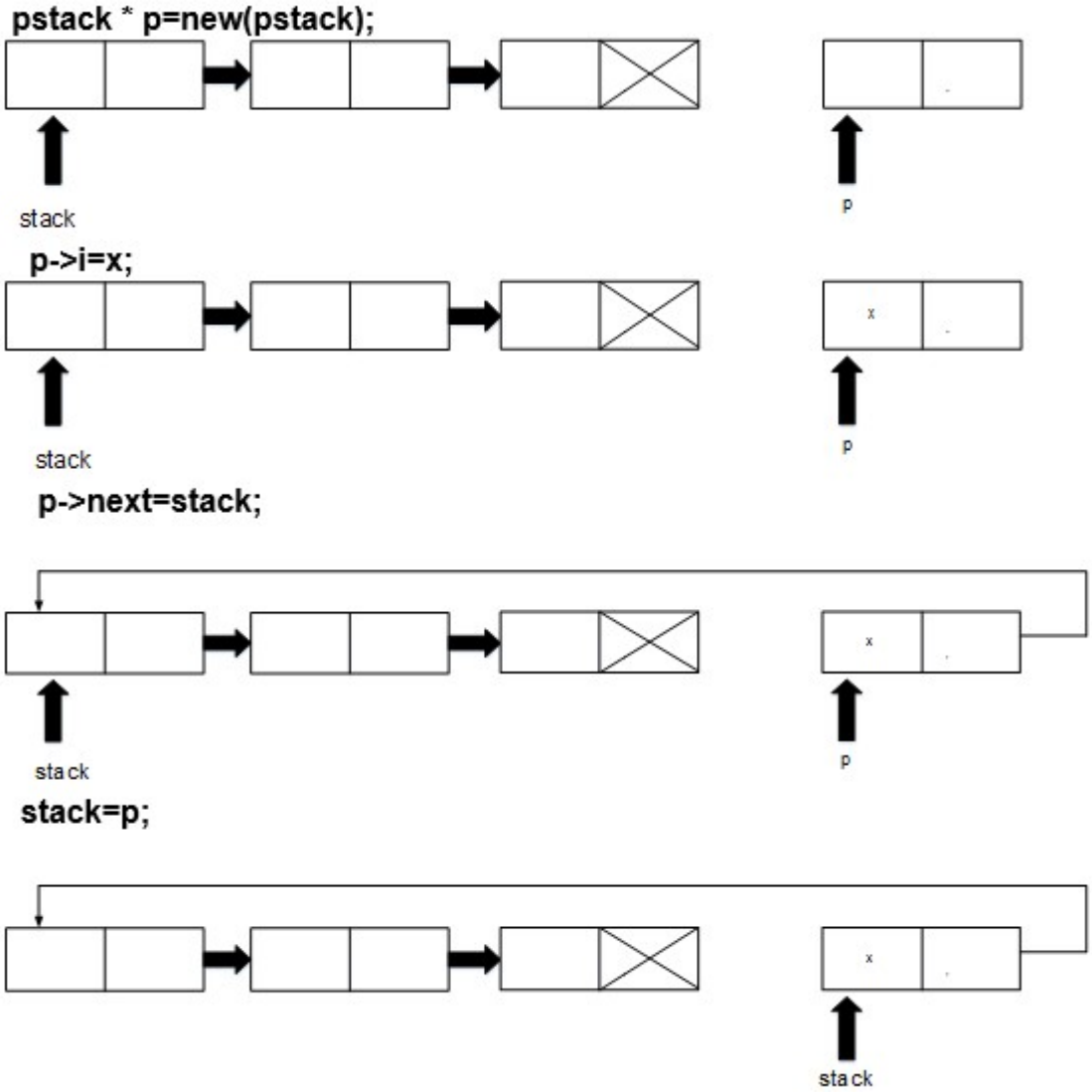


Фиг.2. Динамична реализация на стек

Fig.2. Dynamic realization of stack

След това се илюстрира програмния код оператор по оператор (фигура 3).

Then, the operator code is then illustrated operator per operator (Figure 3).



Фиг.3. Стъпки на изпълнението на програма, реализирана с алгоритъм за стек

Fig.3. Stages of execution of the program realizing algorithm for stack

4. Стандартна библиотека STL (Standart Template Library)

Библиотеката съдържа пет основни видове компоненти:

- алгоритми (algorithm): определят изчислителни процедури (sort, unique,

4. Standard library STL (Standart Template Library)

The library contains five main types of components:

- Algorithm: Define computational procedures (sort, unique, count,

count, reverse, next_permutation, lower_bound, upper_bound и др.).
- контейнери (container): управляват набор от обекти в паметта.
- итератори (iterator): осигуряват на алгоритмите средства за достъп до съдържанието на контейнерите.
- функционални обекти (function object): капсулират функции в обектите за използване на други компоненти.
- адаптери (adaptor): адаптират компонентите за осигуряване на различен интерфейс.

Стекът е представител на контейнерите - намира се в заглавния файла **<stack>**, който се включва в програма на C++ чрез **#include <stack>** (фигура 4).

reverse, next_permutation, lower_bound, upper_bound, etc.);
- Containers: manage a set of objects in memory;
- Iterators: provide algorithms with means of accessing the contents of the containers;
- Function objects: Encapsulate functions in objects to use other components;
- Adapters: Adapt the components to provide a different interface.

The stack is a representative of the containers - located in the <stack> header file, which is included in the C++ program by **#include <stack>** (Figure 4).

stack<int> s;	// дефинира стек с елементи от тип *int* //definition of stack with elements of type *int*
void push(X);	// добавя елемент в стека //adding element in the stack
void pop();	// премахва елемент от върха на стека // removes an element from the top of the stack
T top();	// връща елемента на върха на стека //return element to the top of the stack
bool empty();	// проверява дали стека е празен // checks whether the stack is empty

5. Приложения

Симулация на рекурсия чрез програмен стек. В почти всички езици за програмиране, неявно за програмиста част от паметта се използва за програмен стек. Когато се изпълняват рекурсивни извиквания, локалните данни за всяко поредно извикване се разполагат в паметта на принципа на стека (на ниво компилатор/операционна система реализацията на този процес е точно такава: съществува част от паметта, наречена програмен стек, и регистър-указател към върха на стека) [3].

Проверка за съответствие на броя на отварящите и затварящите скоби

Да се напише програма, която прочита

5. Applications

Recursion simulation through program stack. In almost all programming languages, implicitly for the programmer, a portion of the memory is used for a program stack. When recursive calls are executed, the local data for each consecutive call is stored in stack memory (at compiler/operating system level, the implementation of this process is exactly the same: there is a piece of memory called a program stack and a directory register at the top On the stack) [3].

Check the number of opening and closing brackets to match. Write a program that reads the source code

изходния код на програма на Си и проверява дали двойките (и); { и }; [и]; /* и */ участват симетрично в него.

Изчисляването на изрази се извършва лесно чрез стек [3].

Представяне на аритметичен израз в обратен полски запис. Аритметичният израз се състои от едноцифрени числа, аритметични операции, зададени със съответен знак (+) за събиране, (-) за изваждане, (*) за умножение и (/) за деление.

Алгоритъмът се състои от следните стъпки:

1. Ако поредният символ е отваряща скоба - включва се в стека.

2. Ако поредният символ е цифра - записва се директно в изходния низ.

3. Ако поредният символ е знак за операция - включва се в стека при условие, че е празен или на върха му се намира знак за операция с по-нисък приоритет.

4. Ако на върха на стека се намира операция с по-висок приоритет, тя се изключва от стека и се добавя към изходния низ, а на нейно място се включва текущата операция.

5. При затваряща скоба се изключва всичко до отварящата скоба и се добавя в изходния низ без нея.

6. Когато входният низ се изпразни, цялото съдържание на стека се добавя към изходния низ.

Пример: входен низ: **((2+4)*3-4*(5-2))/2**

изходен низ (в обратен полски запис):

2 4 + 3 * 4 5 2 - * - 2 /

Пресмята се аритметичен израз, записан в обратен полски запис. Тези изчисления се извършват в следния ред:

- Ако поредният символ е цифра – включва се в стека;
- Ако поредният символ е знак за операция изключваме 2 елемента от стека и прилагаме върху тях операцията, съответна на знака като резултата се

of a C program and checks whether the pairs (и); {и}; [и]; /*и*/ participate in it symmetrically. Calculation of expressions is easy by stack [3].

Presentation of arithmetic expression in reverse Polish notation. The arithmetic expression consists of one-digit numbers, arithmetic operations assigned with a corresponding sign (+) for addition, (-) for subtraction, (*) for multiplication and (/) for division.

The algorithm consists of the following steps:

1. If the next character is an opening bracket – It's included in the stack.

2. If the next character is a digit - it is written directly to the output string.

3. If the next character is a sign of an operation, it is included in the stack provided it is empty or a lower priority operation sign is at the top of the stack.

4. If a higher priority operation is at the top of the stack, it is excluded from the stack and added to the output string, and the current operation is included in its place.

5. With a clamping bracket, everything is off to the opening bracket and added to the output string without it.

6. When the input string is empty, the entire contents of the stack are added to the output string.

Example: input string:

((2+4)*3-4*(5-2))/2

Output string (in Reverse Polish notation): **2 4 + 3 * 4 5 2 - * - 2 /**

An arithmetic expression written in a reverse Polish notation is calculated. These calculations are made in the following order:

- If the next character is a digit - it is included in the stack;
- If the next character is a sign of an operation, we exclude 2 elements

включва в стека;
- След обработка на целия низ в стека се намира крайния резултат от пресмятанията.

Изходен резултат за примерния низ: **3**

Пресмята се аритметичен израз с числа – използват се два стека – един за операциите и един за числата.

Разглежда се всеки символ от аритметичния израз:
- Ако поредният символ е отваряща скоба – включва се в стека с операциите;
- Ако поредният символ е знак за операция (Докато стека не е празен и приоритета на операцията е по-малък или равен на операцията на върха в стека с операциите, пресмятаме с операцията от стека). Включва се знака за операцията в стека;
- Ако поредният символ е цифра – включва се цялото число, започващо с тази цифра в стека с числата;
- Ако поредният символ е затваряща скоба (Докато операцията в стека е различна от отваряща скоба пресмятаме операциите като за всяка операция изключваме две числа от стека с числата и върху техните стойности прилагаме операцията.

Резултатът от операцията се записва в стека с числата). Изключваме нова операция от стека и така, докато има знак за операция в стека извършваме пресмятания.

6. Организация на учебния материал в DSLearning

Учебният материал по всяка от темите е организиран на модулен принцип. Задачите са разпределени в следните три основни модула:
- Емпиричен модул;
- Теоретичен модул;
- Практикоприложен модул.

Задачите в емпиричния модул са насочени към формиране у обучаемите на

from the stack and apply the operation corresponding to the character on the stack and the result is included in the stack;
- After processing the entire string in the stack, the final result of the calculations is found.

Output for the sample string: **3**

An arithmetic expression with numbers is calculated - two stacks are used - one for operations and one for numbers.

Any symbol of the arithmetic expression is considered:
- If the next character is an opening bracket, it is included in the stack of operations;
- If the next character is a sign of an operation (While the stack is not empty and the priority of the operation is less than or equal to the operation at the top of the stack with the operations, we calculate the operation from the stack). Include the sign for the operation in the stack;
- If the next character is a digit - include the integer starting with that number in the stack with the numbers;
- If the next character is a closing clamp (While the operation in the stack is different from an opening clamp, we calculate the operations as for each operation we exclude two numbers from the stack with the numbers and their values we apply the operation.

The result of the operation is recorded in the stack with the numbers). We exclude a new operation from the stack, so we perform calculations while there is a sign for an operation in the stack.

6. Organization of the training material in DSLearning

The training material on each

умения и способности за опериране с обекти от логическия обем на понятието и с дефиницията на понятието, както и за разпознаване и прилагане на характеристичните му свойства, съотнесени към признаците за съществуване на понятието.

Теоретичният модул е насочен към формиране у обучаемите на умения за откриване на определящи признаци, коректно използване на фактическата структура на понятията, интерпретиране на моделите на поведение на обектите и съпоставка на тези модели чрез сравнителни анализи.

Формирането у обучаемите на умения за моделиране на обекти и свойствата им, разширение на функционалността на понятията, генериране на нови понятия са основна задача при практико-приложния модул [1].

7. Заключение

В доклада са представени начини за организиране на стек в статичен и динамичен вид и примери за тяхното приложение. Представената структура допълва известните от литературата източници като подобрява нагледността на материала и улеснява прилагането му за целите на обучението.

Описаният начин за организация на динамичен стек и статичен такъв с използване на масив, дава възможност да се изследват основните операции с двата типа структури от данни и да се избегнат затрудненията при тяхното овладяване от обучаемите.

Благодарности

Работата по настоящия доклад е свързана с изследвания по научен проект 2.ФТТ/30.04.2015г. „Приложение на виртуални лаборатории във висшите училища“

subject is organized on a modular basis. The tasks are divided into the following three basic modules:

- Empirical module;
- Theoretical module;
- Practical application module.

The tasks in the empirical module are aimed at forming the students the skills and abilities to operate with objects from the logical volume of the concept and the definition of the concept as well as to recognize and apply its characteristic properties related to the features of the existence of the concept.

The theoretical module is aimed at forming learners' skills to identify identifying features, to use the factual structure of concepts correctly, to interpret the patterns of behavior of objects and to compare these models with comparative analyzes.

Forming learners of skills to model objects and their properties, expanding the functionality of concepts, generating new concepts is a major task in the practical application module [1].

7. Conclusion

The article presents ways to organize stacks in static and dynamic form and examples of their application.

The presented structure complements the sources known from the literature by improving the visibility of the material and facilitating its application for training purposes.

The described method of organization of dynamic and static stack using an array, gives the opportunity to explore the basic operations with two types of data structures and to avoid difficulties in their assimilation from the students.

Acknowledgements

The work on this report is related to

studies under scientific project 2.FTT/30.04.2015. "Application of virtual labs in universities"

8. Литература

[1] Dyankova, V., M. Yankov, R. Boyadzhieva. (2011). Learning of Concepts on Data Structures with WEB-based DSLearning System, 40 Years Shumen University 1971-2011, Shumen, ISBN: 978-954-577-603-8, pp. 269-274. (in Bulgarian)

[2] Hristova, I. (2007). Abstract data type (stack, queue) – methodical aspects of it teaching, Annual of the university of Mining and geology "st. Ivan Rilski", Vol. 50, Part IV, Humanitarian sciences and Economics. (in Bulgarian)

[3] Nakov, P., P. Dobrikov. (2002). Programming = ++ Algorithms; Third Edition, ISBN: 954-8905-06-X. (in Bulgarian)

[4] Nedeva, V., Z. Zlatev, S. Atanasov. (2012). Effective Resources Use for Virtual Laboratories through Cloud Computing and Services. In ICVL, The 7th International Conference on Virtual Learning, pp.322-328

[5] Shivacheva, I. (2015). Multimedia in education – art and professionalism. Journal of Innovation and entrepreneurship, vol.3-4, ISSN 1314-9180, pp.24-37.

8. References

Контакти

ас. Галя Шивачева

Тракийски Университет,
Факултет „Техника и технологии",
ул.Граф Игнатиев No38, 8602,
Ямбол, България,
e-mail: shivacheva_g@abv.bg

Contacts:

assistant professor Galya Shivacheva

Trakia University, faculty of Technics and technologies, 38 Graf Ignatiev str., 8602,
Yambol, Bulgaria,
e-mail: shivacheva_g@abv.bg