Прізвище: Якіб'юк
Ім'я: Ігор
Група: ПП-16
Варіант: 28
Дата захисту: 05.03.2024р.

Кафедра: САПР
Дисципліна: Алгоритмізація та програмування. Ч.2
Перевірив: Патерега Ю.І.

ЗВІТ
до лабораторної роботи №1
на тему " ДИНАМІЧНІ ОБ'ЄКТИ ТА СПОСОБИ ЇХ ВИКОРИСТАННЯ.
СПИСКИ МЕТОДИЧНІ ВКАЗІВКИ"

Мета роботи: ознайомитись із особливостями застосування динамічних об'єктів складної структури: списками, стеками та чергами; з операціями, які виконуються над елементами цих об'єктів. Набути практичних навичок програмування з використанням динамічних об'єктів складної структури.

Відповіді на контрольні запитання:

1. **Яка особливість динамічного рядка?**

Одна з особливостей динамічного рядка полягає в тому, що він може змінювати свій розмір в процесі виконання програми. Це дозволяє ефективніше використовувати пам'ять, оскільки програма може виділяти пам'ять для рядка тільки тоді, коли це потрібно, і звільняти пам'ять, коли рядок більше не потрібен.

2. **Що таке список і навіщо він використовується?**

Список - це структура даних, яка складається з вузлів, кожен з яких містить певне значення та посилання на наступний або попередній вузол. Список використовується для зберігання та організації послідовності даних, які можуть змінюватись в процесі виконання програми.

3. **Чим відрізняється однонаправлений список від двонаправленого списку?**

Однонаправлений список містить посилання на наступний вузол, тоді як двонаправлений список містить посилання на наступний та попередній вузол. Це дає можливість здійснювати швидкий доступ до попереднього вузла у двонаправленому списку. Двонаправлений список потребує більше пам'яті, оскільки утворюється більше посилань.

4. **У чому особливість кільцевого списку?**

Особливість кільцевого списку полягає в тому, що останній вузол посилається на перший вузол, створюючи таким чином замкнену послідовність. Це дозволяє здійснювати ефективний доступ до будь-якого вузла у списку.

5. **Як видалити елемент у двонаправленому списку?**

Для видалення елемента у двонаправленому списку необхідно спочатку знайти його вузол, потім видалити зв'язок з попереднім та наступним вузлами, та звільнити пам'ять, яку він займав.

6. Чим відрізняється стек від черги?

Стек - це структура даних, в якій елементи зберігаються та отримуються в зворотному порядку LIFO (Last-In-First-Out). Черга - це структура даних, в якій елементи зберігаються та отримуються в порядку FIFO(First-In-First-Out). Основна відмінність між стеком та чергою полягає в порядку, в якому елементи зберігаються та отримуються зі структури. У стеку останній елемент, який був доданий, є першим, який буде вилучено (LIFO), тоді як у черзі перший елемент, який був доданий, є першим, який буде вилучено (FIFO).

Індивідуальне завдання:
28.Сформувати однонаправлений список S1 з цілочисельними елементами, вибрати елементи, які є парними числами, і перенести їх у список S2. Вивести обидва списки.

Код програми:
```
#include <iostream>
#include <chrono>

using namespace std;
using namespace std::chrono;

struct Node {
    int data;
    Node* next;
};

struct LinkedList {
    Node* head;

    LinkedList() : head(nullptr) {}

    void insert(int val) {
        Node* new_node = new Node;
        new_node->data = val;
        new_node->next = head;
        head = new_node;
    }

    bool search(int val) {
        Node* temp = head;
        while (temp != nullptr) {
            if (temp->data == val)
                return true;
            temp = temp->next;
        }
        return false;
    }

    void remove(int val) {
        Node* temp = head;
        Node* prev = nullptr;
        while (temp != nullptr && temp->data != val) {
            prev = temp;
            temp = temp->next;
        }
```

```
        if (temp == nullptr) {
            cout << "Value not found" << endl;
            return;
        }
        if (prev == nullptr) {
            head = temp->next;
        }
        else {
            prev->next = temp->next;
        }
        delete temp;
    }

    void display() {
        Node* temp = head;
        while (temp != nullptr) {
            cout << temp->data << " ";
            temp = temp->next;
        }
        cout << endl;
    }

    void filteredList() {
        LinkedList L;
        auto start = high_resolution_clock::now(); // Початок вимірювання часу
        Node* temp = head;
        while (temp != nullptr) {
            if (temp->data % 2 == 0) {
                L.insert(temp->data);
            }
            temp = temp->next;
        }
        auto stop = high_resolution_clock::now(); // Кінець вимірювання часу
        auto duration = duration_cast<microseconds>(stop - start); // Обчислення
тривалості виконання

        cout << "Filtered List: ";
        L.display();
        cout << "Time taken by filteredList: " << duration.count() << " microseconds" <<
endl;
    }
};

int main() {
    LinkedList list;
    list.insert(1);

    list.insert(2);
    list.insert(3);
    list.insert(4);
    list.insert(5);
    list.insert(6);
    list.insert(7);
    list.insert(8);
    list.insert(9);
    list.insert(10);

    cout << "Original List: ";
    list.display();
    list.filteredList();
}
```

Результати виконання програми:

Filtered List: 2 4 6 8 10 12 14 16 18 20 22 24 26 28 30 32 34 36 38 40 42 44 46 48 50 52 54 56 58 60 62 64 66 68 70 72 74 76 78 80 82 84 86 88 90 92 94 96 98 100 102 104 106 108 110 112 114 116 118 120 122 124 126 128 130 132 134 136 138 140 142 144 146 148 150 152 154 156 158 160 162 164 166 168 170 172 174 176 178 180 182 184 186 188 190 192 194 196 198 200 202 204 206 208 210 212 214 216 218 220 222 224 226 228 230 232 234 236 238 240 242 244 246 248 250 252 254 256 258 260 262 264 266 268 270 272 274 276 278 280 282 284 286 288 290 292 294 296 298 300 302 304 306 308 310 312 314 316 318 320 322 324 326 328 330 332 334 336 338 340 342 344 346 348 350 352 354 356 358 360 362 364 366 368 370 372 374 376 378 380 382 384 386 388 390 392 394 396 398 400 402 404 406 408 410 412 414 416 418 420 422 424 426 428 430 432 434 436 438 440 442 444 446 448 450 452 454 456 458 460 462 464 466 468 470 472 474 476 478 480 482 484 486 488 490 492 494 496 498 500 502 504 506 508 510 512 514 516 518 520 522 524 526 528 530 532 534 536 538 540 542 544 546 548 550 552 554 556 558 560 562 564 566 568 570 572 574 576 578 580 582 584 586 588 590 592 594 596 598 600 602 604 606 608 610 612 614 616 618 620 622 624 626 628 630 632 634 636 638 640 642 644 646 648 650 652 654 656 658 660 662 664 666 668 670 672 674 676 678 680 682 684 686 688 690 692 694 696 698 700 702 704 706 708 710 712 714 716 718 720 722 724 726 728 730 732 734 736 738 740 742 744 746 748 750 752 754 756 758 760 762 764 766 768 770 772 774 776 778 780 782 784 786 788 790 792 794 796 798 800 802 804 806 808 810 812 814 816 818 820 822 824 826 828 830 832 834 836 838 840 842 844 846 848 850 852 854 856 858 860 862 864 866 868 870 872 874 876 878 880 882 884 886 888 890 892 894 896 898 900 902 904 906 908 910 912 914 916 918 920 922 924 926 928 930 932 934 936 938 940 942 944 946 948 950 952 954 956 958 960 962 964 966 968 970 972 974 976 978 980 982 984 986 988 990 992 994 996 998 1000

Time taken by filteredList: 68 microseconds

D:\Algorithmization&Programming2\Labs\Lab1\x64\Debug\Lab1.exe (process 10748) exited with code 0.

Press any key to close this window . . .

Аналіз результатів:

Використовуючи структури, я створив модель вузла(елемента), що являє собою дані і посилання на наступний елемент, а також модель однонапрямленого списку цих вузлів з методами: пошуку, додавання, видалення, фільтрації елементів. Беручи до уваги завдання: створити однонапрямлений список з цілочисельними елементами, мені необхідно створити ще один список, в якому будуть тільки парні числа. Використовуючи цикл, а також бібліотеку chrono для заміру часу.

Висновок:

На даній лабораторній роботі я навчився працювати з динамічними об’єктами, стеками, чергами і однонапрямленими списками, використовуючи різноманітні методи для роботи з ними. Зрозумів особливості роботи з ними і потреби у використанні. Виконав індивідуальне завдання, продемонструвавши скріншоти вище.

Додаткове завдання з використанням масивів:

```
#include <iostream>
#include <stdexcept>
#include <chrono>

struct CustomArray {
    int* data;
```

```
int capacity;
int size;

CustomArray(int initialCapacity = 10) {
    capacity = initialCapacity;
    data = new int[capacity];
    size = 0;
}


~CustomArray() {
    delete[] data;
}


void insert(int value) {
    if (size == capacity) {
        capacity *= 2;
        int* newData = new int[capacity];
        for (int i = 0; i < size; ++i) {
            newData[i] = data[i];
        }
        delete[] data;
        data = newData;
    }
    data[size++] = value;
}


int get(int index) const {
    if (index < 0 || index >= size) {
        throw std::out_of_range("Index out of range");
    }
    return data[index];
}


void set(int index, int value) {
    if (index < 0 || index >= size) {
        throw std::out_of_range("Index out of range");
    }
    data[index] = value;
}


int getSize() const {
    return size;
}


CustomArray copyAndFilterEvenNumbers() const {
    auto start = std::chrono::high_resolution_clock::now();

    CustomArray result(size);

    for (int i = 0; i < size; ++i) {
        if (data[i] % 2 == 0) {
            result.insert(data[i]);
        }
    }

    auto stop = std::chrono::high_resolution_clock::now();
```

```
        auto duration = std::chrono::duration_cast<std::chrono::microseconds>(stop -
start);
        std::cout << "Time taken to filter even numbers: " << duration.count() << "
microseconds" << std::endl;

        return result;
    }
};

int main() {
    CustomArray arr;


    for (int i = 1; i <= 10000; ++i) {
        arr.insert(i);
    }


    CustomArray evenNumbersArray = arr.copyAndFilterEvenNumbers();


    std::cout << "Even numbers in the new array: ";
    for (int i = 0; i < evenNumbersArray.getSize(); ++i) {
        std::cout << evenNumbersArray.get(i) << " ";
    }
    std::cout << std::endl;

    return 0;
}
```

Результат:

Time taken to filter even numbers: 2 microseconds
Even numbers in the new array: 2 4 6 8 10 12 14 16 18 20 22 24 26 28 30 32 34 36 38 40 42 44 46 48 50 52 54 56 58 60 62 64 66 68 70 72 74 76 78 80 82 84 86 88 90 92 94 96 98 100 102 104 106 108 110 112 114 116 118 120 122 124 126 128 130 132 134 136 138 140 142 144 146 148 150 152 154 156 158 160 162 164 166 168 170 172 174 176 178 180 182 184 186 188 190 192 194 196 198 200 202 204 206 208 210 212 214 216 218 220 222 224 226 228 230 232 234 236 238 240 242 244 246 248 250 252 254 256 258 260 262 264 266 268 270 272 274 276 278 280 282 284 286 288 290 292 294 296 298 300 302 304 306 308 310 312 314 316 318 320 322 324 326 328 330 332 334 336 338 340 342 344 346 348 350 352 354 356 358 360 362 364 366 368 370 372 374 376 378 380 382 384 386 388 390 392 394 396 398 400 402 404 406 408 410 412 414 416 418 420 422 424 426 428 430 432 434 436 438 440 442 444 446 448 450 452 454 456 458 460 462 464 466 468 470 472 474 476 478 480 482 484 486 488 490 492 494 496 498 500 502 504 506 508 510 512 514 516 518 520 522 524 526 528 530 532 534 536 538 540 542 544 546 548 550 552 554 556 558 560 562 564 566 568 570 572 574 576 578 580 582 584 586 588 590 592 594 596 598 600 602 604 606 608 610 612 614 616 618 620 622 624 626 628 630 632 634 636 638 640 642 644 646 648 650 652 654 656 658 660 662 664 666 668 670 672 674 676 678 680 682 684 686 688 690 692 694 696 698 700 702 704 706 708 710 712 714 716 718 720 722 724 726 728 730 732 734 736 738 740 742 744 746 748 750 752 754 756 758 760 762 764 766 768 770 772 774 776 778 780 782 784 786 788 790 792 794 796 798 800 802 804 806 808 810 812 814 816 818 820 822 824 826 828 830 832 834 836 838 840 842 844 846 848 850 852 854 856 858 860 862 864 866 868 870 872 874 876 878 880 882 884 886 888 890 892 894 896 898 900 902 904 906 908 910 912 914 916 918 920 922 924 926 928 930 932 934 936 938 940 942 944

946 948 950 952 954 956 958 960 962 964 966 968 970 972 974 976 978 980 982 984 986 988 990 992 994 996 998 1000

D:\Algorithmization&Programming2\Labs\Lab1\Lab1Array\x64\Debug\Lab1Array.exe (process 20640) exited with code 0.
Press any key to close this window . . .