Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

«Национальный исследовательский университет ИТМО»

Факультет информационных технологий и программирования

Программирование

Лабораторная работа №3

STL.Контейнеры.

Выполнил студент группы № M3109
Гумбатов Владислав Юрьевич
Подпись:

Санкт-Петербург
2022

Задача

Целью лабораторной работы является написание реализация собственного кольцевого буфера в виде stl-совместимого контейнера.

Буфер должен обладать следующими возможностями:

1) Вставка и удаление в конец
2) Вставка и удаление в начало
3) Доступ в конец, начало
4) Доступ по индексу
5) Изменение капасити

Также требуется реализовать обобщенные алгоритмы:

1) all_of
2) any_of
3) none_of
4) one_of
5) is_sorted
6) is_partitioned
7) find_not
8) find_backward
9) is_palindrome

Входные данные:

Входными данными в программе является размер кольцевого буфера, либо итераторы на начало и конец в случае алгоритмов.

Выходные данные:

В качестве выходных данных программа выводит результаты применения операций на буфер, либо результат алгоритмов.

Код:

```
#include <iostream>

#define size unsigned long long

template <typename T>
class buff {

public:
    buff(size cap = 1) {

        rbuff = new T[cap];

        condition = new status[cap];

        for (int i = 0; i < cap; i++) {

            rbuff[i] = 0;
            condition[i] = white;


        }
        this->cap = cap;
        this->begin = 0;
        this->end = 0;
        this->sizeofring = 0;
    }

    void print() {

        for (int i = 0; i < cap; i++) {

            std::cout << rbuff[i] << " ";



        }
        std::cout << std::endl;
        //for (int i = 0; i < cap; i++) {

        //  std::cout << condition[i] << " ";


        //}


    }

    void printInOrder() {


        int i = begin;

        if (sizeofring == 1) {

            std::cout << rbuff[i] << " ";
```

```
        std::cout << rbuff[i] << " ";
        std::cout << std::endl;
        return;
    }

    for (i; i != end; i++) {


        try
        {
            if (i > cap - 1) {

                throw - 1;

            }
        }
        catch (int j)
        {
            if (j == -1) {

                i = 0;

            }
        }

        if (condition[i] == white) {

            continue;

        }

        if (i == end) {

            std::cout << rbuff[i] << " ";
            std::cout << std::endl;
            return;

        }


        std::cout << rbuff[i] << " ";


    }
    if (i == end) {

        std::cout << rbuff[i] << " ";
        std::cout << std::endl;
        return;

    }


}
```

```
void push_back(T temp) {

    if (sizeofring == 0) {
        sizeofring++;
        rbuff[end] = temp;
        condition[end] = blue;
        return;

    }

    if (sizeofring > 1 && sizeofring != cap) {
        sizeofring++;
        end++;
        rbuff[end] = temp;
        condition[end] = blue;
        return;
    }

    if (sizeofring == cap) {

        try
        {
            end++;
            begin++;
            if (end < 0 || end > cap - 1) {

                throw - 1;
            }

            if (begin > cap - 1) {

                throw - 2;
            }
        }
        catch (int i)
        {
            if (i == -1) {

                end = 0;

            }
            if (i == -2) {

                begin = 0;

            }
        }

        rbuff[end] = temp;
        return;
    }

}
```

```
void push_front(T temp) {

    if (sizeofring == 0) {

        sizeofring++;
        rbuff[begin] = temp;
        condition[begin] = blue;
        return;

    }
    if (sizeofring == 1 && condition[cap - 1] == white) {

        sizeofring++;
        rbuff[cap - 1] = temp;
        begin = cap - 1;
        condition[cap - 1] = blue;
        return;

    }
    if (sizeofring > 1 && condition[cap - 1] == blue) {


        if (end != begin - 1) {
            sizeofring++;
            rbuff[begin - 1] = temp;
            begin--;
            condition[begin] = blue;
            return;
        }
        else {

            rbuff[begin - 1] = temp;
            begin--;
            try
            {
                end--;
                if (end < 0 || end > cap) {

                    throw - 1;
                }
            }
            catch (int i)
            {
                if (i == -1) {

                    end = cap - 1;
                }


            }
            rbuff[begin] = temp;
            return;

        }
```

```
T& getStart() {

    return rbuff[begin];
}

T& getEnd() {

    return rbuff[end];

}

T& operator[] (size i) {


    try
    {
        if (i >= cap) {

            throw - 1;
        }

        return rbuff[i];
    }
    catch (int j)
    {
        if (j == -1) {


            std::cout << "Error: Overflow" << std::endl;

            exit(-1);

        }
    }
}

void raiseCap(size i) {

    size newSize = cap + i;

    T* temp = new T[newSize]();

    status* temp2 = new status[newSize]();

    for (size j = 0; j < cap   ; j++) {

        temp[j] = rbuff[j];

        temp2[j] = blue;


    }
```

```
    rbuff = temp;

    condition = temp2;

    temp = nullptr;

    delete[] temp;

    temp2 = nullptr;

    delete[] temp2;

    cap = newSize;
}

void lowerCap(size i) {

    if (cap == 0) {

        std::cout << "Error: Overflow" << std::endl;

        exit(-1);

    }

    size newSize = cap - i;


    T* temp = new T[newSize]();

    int n = 0;


    for (int c = begin; c != end; c++) {



        try
        {
            if (c > cap - 1) {

                throw - 1;

            }
        }
        catch (int j)
        {
            if (j == -1) {

                c = 0;

            }
        }

        if (c == end) {
```

```
            break;
        }
        if (n > cap - 1) {
            break;
        }
        temp[n] = rbuff[c];
        n++;
    }

    try
    {
        end--;
        if (end < 0 || end > cap - 1) {
            throw - 1;
        }
    }
    catch (int j)
    {
        if (j == -1) {
            end = cap-1;
        }
    }

    rbuff = temp;
    temp = nullptr;
    delete[] temp;
    cap = newSize;
    begin = 0;
}
```

```
private:

    T* rbuff;

    size cap;
    size begin;
    size end;
    size sizeofring;


    enum status
    {
        white,
        blue,
        red
    };


    status* condition;
};
```

```
#include <iostream>

template <typename T>
bool isPos(T temp) {

    try
    {
        if (temp > 0) {

            return true;
        }
        else {

            return false;
        }
    }
    catch (const std::exception&)
    {
        std::cout << "Error" << std::endl;

    }

}

template <typename T, typename T2, typename T3>
bool any_of(T s, T2 e, T3 func) {

    for (s; s != e; ++s) {

        if (isPos(*s)) {

            return true;

        }



    }

    return false;

}

template <typename T, typename T2, typename T3>
bool all_of(T s, T2 e, T3 func) {

    for (s; s != e; ++s) {

        if (!isPos(*s)) {

            return false;

        }
```

```
    return true;

}

template <typename T, typename T2, typename T3>
bool none_of(T s, T2 e, T3 func) {

    for (s; s != e; ++s) {

        if (isPos(*s)) {

            return false;

        }


    }

    return true;

}

template <typename T, typename T2, typename T3>
bool one_of(T s, T2 e, T3 func) {

    int cnt = 0;
    for (s; s != e; ++s) {

        if (isPos(*s)) {

            cnt++;

        }


    }

    if (cnt == 1) {

        return true;
    }

    else {

        return false;
    }

}

template <typename T, typename T2>
bool is_sorted(T s, T2 e) {

    T2 end = --e;
```

```
    e++;

    for (s; s != e; ++s) {

        try
        {

            T tmp = ++s;


            --s;
            if (*s > *tmp) {

                return false;

            }

            if (*tmp == *end) {

                break;
            }

        }
        catch (const std::exception&)
        {
            break;
        }



    }

    return true;
}

template <typename T, typename T2, typename T3>
auto find_not(T s, T2 e, T3 search) {

    for (s; s != e; ++s) {

        if (*s != search) {



            return *s;

        }


    }

    return -1;

}

template <typename T, typename T2, typename T3>
auto find_backward(T s, T2 e, T3 search) {

    --e;

    for (e; e != s; --e) {

        if (*e != search) {



            return *e;

        }


    }

    return -1;

}
```

```cpp
#include <iostream>
#include <list>
#include <vector>
#include "buff.h"
#include "algo.h"


#define endl std::endl
#define cout std::cout



int main() {


    cout << "Create ring buffer:" << endl;

    cout << endl;

    buff <int> a(4);

    a.print();

    cout << endl;

    cout << "Push_back 1:" << endl;

    cout << endl;

    a.push_back(1);

    cout << endl;

    cout << "Print in memory:" << endl;

    cout << endl;

    a.print();

    cout << endl;

    cout << "Print in order:" << endl;

    cout << endl;

    a.printInOrder();

    cout << endl;

    cout << "Push_front 2:" << endl;

    cout << endl;

    a.push_front(2);
```

```
cout << "Print in memory:" << endl;

cout << endl;

a.print();

cout << endl;

cout << "Print in order:" << endl;

a.printInOrder();

cout << endl;

cout << "Push_back 3:" << endl;

cout << endl;

a.push_back(3);

cout << endl;

cout << "Print in memory:" << endl;

cout << endl;

a.print();

cout << endl;

cout << "Print in order:" << endl;

cout << endl;

a.printInOrder();

cout << endl;

cout << "Push_front 4:" << endl;

cout << endl;

a.push_front(4);

cout << endl;

cout << "Print in memory:" << endl;

cout << endl;

a.print();

cout << endl;
```

```
cout << "Print in order:" << endl;

cout << endl;

a.printInOrder();

cout << endl;

cout << "Push_front 5:" << endl;

cout << endl;

a.push_front(5);

cout << endl;

cout << "Print in memory:" << endl;

cout << endl;

a.print();

cout << endl;

cout << "Print in order:" << endl;

cout << endl;

a.printInOrder();

cout << endl;

cout << "Push_front 6:" << endl;

cout << endl;

a.push_front(6);

cout << endl;

cout << "Print in memory:" << endl;

cout << endl;

a.print();

cout << endl;

cout << "Print in order:" << endl;

cout << endl;

a.printInOrder();
```

```
cout << endl;

cout << "Rise cap to 5:" << endl;

cout << endl;

a.raiseCap(1);

cout << endl;

cout << "Print in memory:" << endl;

cout << endl;

a.print();

cout << endl;

cout << "Print in order:" << endl;

cout << endl;

a.printInOrder();

cout << endl;

cout << "Push back 7:" << endl;

cout << endl;

a.push_back(7);

cout << endl;

cout << "Print in memory:" << endl;

cout << endl;

a.print();

cout << endl;

cout << "Print in order:" << endl;

cout << endl;

a.printInOrder();

cout << endl;


cout << "low cap to 4:" << endl;

cout << endl;
```

```
    a.lowerCap(1);

    cout << endl;

    cout << "Print in memory:" << endl;

    cout << endl;

    a.print();

    cout << endl;

    cout << "Print in order:" << endl;

    cout << endl;

    a.printInOrder();

    cout << endl;

}
```

Вывод:

Программа успешно работает. Цель лабораторной работы достигнута.