|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ **Информатика и системы управления**

КАФЕДРА **Компьютерные системы и сети (ИУ6)**

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.03.01 Информатика и вычислительная техника**

**Отчет**

|  |  |
| --- | --- |
| **по лабораторной работе №** | 10 |

**Название:**

QT Создание контейнеров.

**Дисциплина:** Объектно-ориентированное программирование

Вариант 16.

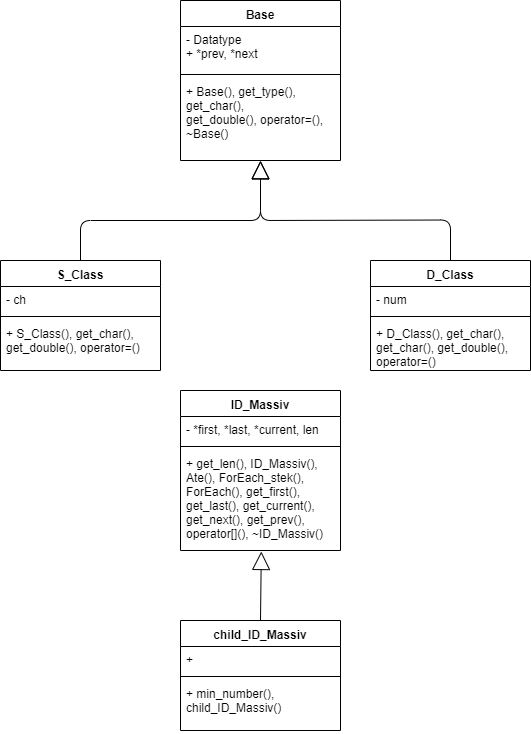
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студент | ИУ6-21Б |  |  | Т.Е.Старжевский |
|  | (Группа) |  | (Подпись, дата) | (И.О. Фамилия) |
|  |  |  |  |  |
| Преподаватель |  |  |  | О.А.Веселовская |
|  |  |  | (Подпись, дата) | (И.О. Фамилия) |

Москва, 2022

Моделировать массив, в качестве элементов которого могут использоваться вещественные числа и буквы латинского алфавита. Операции: запись элемента, чтение элемента, печать элементов. При записи и чтении элементов обращение к ним организовать по индексу. Создать класс-потомок, который содержит функцию вычисления минимального числа в массиве. Тестировать полученную модель.

Разработать собственную иерархию классов, готовые контейнеры Qt не использовать. Пользовательский интерфейс для работы с моделью реализовать на Qt. В отчете представить диаграмму классов и обосновать выбранную структуру представления данных.

**Диаграмма классов предметной области:**

****

**Код программы:**

#ifndef MYCONTAINERS\_H

#define **MYCONTAINERS\_H**

#include <iostream>

using namespace std;

enum class **DataType** { **Char**, **Double** };

class **Base**

{

protected:

DataType **type**;

public:

Base\* **prev**, \* **next**;

**Base**()

{

prev = next = nullptr; type = DataType::Char;

}

DataType **get\_type**() { return type; }

virtual char\* ***get\_char***() = 0;

virtual double\* ***get\_double***() = 0;

virtual Base& operator**=**(char **var**) = 0;

virtual Base& operator**=**(double **var**) = 0;

virtual Base& operator**=**(int **var**) = 0;

virtual Base& operator**=**(Base& **var\_class**) = 0;

virtual ~**Base**() { static int **i** = 1; i++; /\*cout << "\nDestructor Base " << i++ << "\n"; \*/ };

friend ostream& operator**<<**(ostream& **os**, Base& **var\_class**);

};

class **S\_Class** : public Base

{

private:

char **ch**;

public:

**S\_Class**(char **ich** = '\0') :Base(), ch(ich) { this->type = DataType::Char; }

char\* ***get\_char***() override

{

return &ch;

}

double\* ***get\_double***() override

{

return nullptr;

}

Base& operator **=** (Base& **var\_class**) override

{

this->ch = \*(var\_class.*get\_char*());

return \*this;

}

Base& operator **=** (char **var**) override

{

this->ch = var;

return \*this;

}

Base& operator**=** (double **var**) override

{

this->ch = static\_cast<char>(var);

return \*this;

}

Base& operator **=** (int **var**) override

{

this->ch = static\_cast<char>(var);

return \*this;

}

};

class **D\_Class** : public Base

{

private:

double **num**;

public:

**D\_Class**(double **inum** = 0.0) :Base(), num(inum) { this->type = DataType::Double; }

**D\_Class**(int **inum** = 0) :Base(), num(inum) { this->type = DataType::Double; }

char\* ***get\_char***() override

{

return nullptr;

}

double\* ***get\_double***() override

{

return &num;

}

Base& operator **=** (Base& **var\_class**) override

{

this->num = \*(var\_class.*get\_double*());

return \*this;

}

Base& operator **=** (double **var**) override

{

this->num = var;

return \*this;

}

Base& operator **=** (int **var**) override

{

this->num = var;

return \*this;

}

Base& operator **=** (char **var**) override

{

this->num = static\_cast<double>(var);

return \*this;

}

};

class **ID\_Massiv**

{

protected:

Base\* **first**, \* **last**, \* **current**;

int **len**;

public:

int **get\_len**() { return len; }

**ID\_Massiv**() { first = last = current = nullptr; len = 0; }

void **Ate**(Base\* **var\_class**);

void **Ate**(int **var**) { D\_Class\* **var\_clas** = new D\_Class(var); Ate(*var\_clas*); }

void **Ate**(double **var**) { D\_Class\* **var\_clas** = new D\_Class(var); Ate(*var\_clas*); }

void **Ate**(char **var**) { S\_Class\* **var\_clas** = new S\_Class(var); Ate(*var\_clas*); }

void **ForEach\_stek**(void (\***func**)(Base\* **var\_class**));

void **ForEach**(void (\***func**)(Base\* **var\_class**));

Base\* **get\_first**() { return current = first; };

Base\* **get\_last**() { return current = last; };

Base\* **get\_current**() { return current; };

Base\* **get\_next**() { if (current->next!=nullptr) return current = current->next;

else return 0;};

Base\* **get\_prev**() { if(current->prev != nullptr) return current = current->prev;

else return 0;};

Base& operator**[]**(int **number**);

~**ID\_Massiv**();

};

ostream& operator**<<**(ostream& **os**, Base& **var\_class**);

class **child\_ID\_Massiv** : public ID\_Massiv

{

public:

**child\_ID\_Massiv**() :ID\_Massiv() {}

double **min\_number**();

};

#endif // MYCONTAINERS\_H

#include "MyContainers.h"

void ID\_Massiv:: **Ate**(Base \***var\_class**)

{

if (first == nullptr)

{

last = current = first = var\_class; len++;

}

else {

var\_class->next = first;

first->prev = var\_class;

first = var\_class;

len++;

}

}

void ID\_Massiv::**ForEach\_stek**(void (\***func**)(Base\* **var\_class**))

{

current = first;

while (current != nullptr)

{

func(*current*);

current = current->next;

}

}

void ID\_Massiv::**ForEach**(void (\***func**)(Base\* **var\_class**))

{

current = last;

while (current != nullptr)

{

func(*current*);

current = current->prev;

}

}

Base& ID\_Massiv::operator**[]**(int **number**)

{

current = last;

int **i** = 0; bool **key** = true;

while (current != nullptr && key)

{

if (i++ == number)

{

return \*current;

key = false;

}

current = current->prev;

}

}

ID\_Massiv::~**ID\_Massiv**()

{

current = first;

while (first != nullptr)

{

first = first->next;

delete current;

current = first;

}

}

ostream& operator**<<**(ostream& **os**, Base& **var\_class**)

{

if (var\_class.type == DataType::Char)

os << \*(var\_class.*get\_char*());

else

os << \*(var\_class.*get\_double*());

return os;

}

double child\_ID\_Massiv::**min\_number**()

{

int **i** = 0, **index** = 0; double **min** = 0; bool **is\_first** = true;

current = last;

while (current != nullptr)

{

if (current->get\_type() == DataType::Double)

{

if (is\_first) {

min = \*current->*get\_double*(); index = i; is\_first = false;

}

else

{

if (\*current->*get\_double*() < min) {

min = \*current->*get\_double*(); index = i;

}

}

}

current = current->prev;

i++;

}

return min;

}

#include <iostream>

#include "MyContainers.h"

#include <conio.h>

//#include <QCoreApplication>

//#include <QLocale>

//#include <QTranslator>

using namespace std;

void **show**(Base\* **var\_class**)

{

if (dynamic\_cast<S\_Class\*>(var\_class))

cout << \*var\_class->*get\_char*() << endl;

else cout << \*var\_class->*get\_double*() << endl;

}

int **main**(int **argc**, char\* **argv**[])

{

/\*

QCoreApplication a(argc, argv);

QTranslator translator;

const QStringList uiLanguages = QLocale::system().uiLanguages();

for (const QString &locale : uiLanguages) {

const QString baseName = "LAB\_10\_" + QLocale(locale).name();

if (translator.load(":/i18n/" + baseName)) {

a.installTranslator(&translator);

break;

}

}

return a.exec();

\*/

child\_ID\_Massiv **Massiv**;

std::cout << "Simple test, Press '1' for number | '2' for symbol | '#' for end: \t";

cout << "Press\n'1' for symbol | '2' for float | '#' for end... ";

char **ch** = 0;

while ((ch = \_getch()) != '#')

switch (ch)

{

case '1': {

char **ch** = '0';

cout << "Input char symbol: ";

cin >> ch;

Massiv.Ate(ch);

cout << "Success, add new or end?\t";

} break;

case '2': {

double **number** = 0;

cout << "Input float number: ";

cin >> number;

Massiv.Ate(number);

cout << "Success, add new or end?\t";

} break;

default: cout << "\nWrong symbol\n'1' for symbol | '2' for float| '#' for end... ";

break;

}

//Console

cout << "\n===== Press any key for show all massiv[i] ======\t";

\_getch();

cout << endl;

for (int **i** = 0; i < Massiv.get\_len(); i++)

{

if (Massiv[i].get\_type() == DataType::Double)

cout << i << " - double - " << Massiv[i] << endl;

else cout << i << " - char - " << Massiv[i] << endl;

}

cout << "\n===== Press any key for show all massiv.forEach ======\t";

\_getch();

cout << endl;

Massiv.ForEach(show);

int **i**;

cout << "\n===== Input number for change this element of massiv: ";

cin >> i;

cout << endl;

cout << "You want to change element " << Massiv[i];

if (dynamic\_cast<S\_Class\*>(&Massiv[i]))

{

char **ch**;

cout << " It is a char element, so input new symbol: ";

cin >> ch;

Massiv[i] = ch;

}

else {

double **num**;

cout << " It is a float element, so input new number: ";

cin >> num;

Massiv[i] = num;

}

cout << "\nNow Massiv[i] consist element: " << Massiv[i] << endl;

cout << "\n===== Press any key for show all massiv[i] ======\t";

\_getch();

cout << endl;

for (int **i** = 0; i < Massiv.get\_len(); i++)

{

if(Massiv[i].get\_type() == DataType::Double)

cout << i << " - double - " << Massiv[i] << endl;

else cout << i << " - char - " << Massiv[i] << endl;

}

cout << "\n===== Press any key for show min number ======\t";

\_getch();

cout << endl;

cout << "Min number: " << Massiv.min\_number() << endl;

cout << "\n===== Press any key for see adrress first & last element ======\t";

\_getch();

cout << endl;

cout << "First address: " << Massiv.get\_first();

cout << "\nNext element: " << Massiv.get\_next();

cout << "\nPrev element: " << Massiv.get\_prev();

cout << "\nLast element: " << Massiv.get\_last() << endl;

system("pause");

return 0;

}

**Тестирование:**

|  |  |  |
| --- | --- | --- |
| Исходные данные | Ожидаемый результат | Полученный результат |
| T, I, M, A, 5.2, 7,8, R, 15, L, 9 | Вывод информации об объектах |  |

При заданных значениях программа работает корректно.

**Вывод:** Научился создавать контейнеры на QT.