

Балтийский государственный технический университет
«ВОЕНМЕХ» им. Д. Ф. Устинова

Факультет «О» Естественнонаучный

Кафедра О7 «Информационные системы и программная инженерия»

Практическая работа №2
по дисциплине «Программирование на ЯВУ»
на тему «КЛАССЫ: ОСНОВНЫЕ ПОНЯТИЯ И ОПРЕДЕЛЕНИЯ»

Выполнил:
Студент Костров Г. Ю.
Группа О712Б
Преподаватель:
Васюков В. М.

Санкт-Петербург
2022 г

Постановка задачи:

Описать три класса: базовый класс «Строка» и производные от него класс «Строка-идентификатор» и класс, заданный индивидуальным вариантом. Обязательные для всех классов методы: конструктор без параметров, конструктор, принимающий в качестве параметра Си-строку, конструктор копирования, деструктор, перегрузка операции присваивания «=». Во всех методах всех классов предусмотреть печать сообщения, содержащего имя метода. Для конструкторов копирования каждого класса дополнительно предусмотреть диагностическую печать количества его вызовов, рекомендуется использовать статические члены класса.

Вариант 12

Дополнительные методы для класса «Строка-идентификатор»: перевод всех символов строки (кроме цифр) в нижний регистр, переопределение операции индексации [], переопределение операции больше или равно >=.

Производный от «Строки» класс Шестнадцатеричная строка.

Строки данного класса могут содержать только символы шестнадцатеричных цифр (как в верхнем, так и в нижнем регистре) и символы - и +, задающие знак числа, которые могут находиться только в первой позиции числа, при отсутствии знака число считается положительным. Если в составе инициализирующей строки будут встречены любые символы, отличные от допустимых, Шестнадцатеричная строка принимает нулевое значение.

Содержимое данных строк рассматривается как шестнадцатеричное число.

Обязательные методы: определение, можно ли представить данное число в формате unsigned int, проверки на равенство ==, проверки на больше >.

Текст программы:

//String.h

```
#ifndef PRACTICE2_STRING_H
#define PRACTICE2_STRING_H

#include <iostream>
#include <string.h>

using namespace std;

class String {
protected:
    char *m_string;
    int length;
    int id;

public:
    string type;
    static int count;

    virtual void toLower() { std::cout << "B\n"; }
    virtual void hexToDec() { std::cout << "B\n"; }
    char &operator[](int index) {
        count << m_string[index];
        return m_string[index];
    }

    // дефолтный конструктор
    String();
};
```

```

// конструктор принимающий на вход строку
String(char *string);

// конструктор копирования
String(const String &str);

// деструктор
virtual ~String();

// перегрузка оператора '='
String &operator=(String &otherString);

// методы
int getStringLength();
char * getString();
int getId();

void setLength(int length) {
    this->length = length;
}

void printProperties(string type) {
    cout << type << " #"<< id << ": " << m_string << endl;
    cout << "Length: " << length << endl;
    cout << endl;
}
};

#endif //PRACTICE2_STRING_H

```

//String.cpp

```

#include "String.h"
#include <string.h>

using namespace std;

// конструктор дефолтный
String::String() {
    m_string = new char[100];
    this->type = "String";
    length = 0;
}

// конструктор с параметром
String::String(char *string) {
    this -> m_string = string;
    this-> type = "String";
    count++;
    id = count;
}

// конструктор копирования
String::String(const String &str) {
    m_string = str.m_string;
    length = str.length;
    count++;
    id = count;
}

```

```

// деструктор
String::~String() {
    cout << "Destructor String" << endl;
    delete m_string;
}

// оператор присваивания
String &String::operator=(String &otherString) {
    m_string = otherString.m_string;
    length = otherString.length;

    count++;
    id = count;
    return *this;
}

// геттеры
int String::getStringLength() {
    return this-> length;
}

char * String::getString() {
    return this-> m_string;
}

int String::getId() {
    cout << this-> id << endl;
    return this->id;
}

```

//HexString.h

```

#ifndef PRACTICE2_HEXSTRING_H
#define PRACTICE2_HEXSTRING_H
#include <iostream>
#include "IdentifiedString.h"
#include "String.h"

class HexString : public String {
private:
    // переменная hexNumbers показывает какие символы может содержать строка
    const char *hexNumbers = "+-0123456789AaBbCcDdEeFf";
    // длина числа (знак не считается). Удобно для конвертации из hex в dec и в
    обратную

    // метод переводит символы в десятичные числа
    int char2dec(char hexChar);

public:

    // обозначает какого знака число
    bool isPositive = true;

    // дефолтный конструктор
    HexString() : String() { m_string = new char[100]; };

    // деструктор
    ~HexString();

```

```

// конструктор принимающий в качестве параметра строку
HexString(char *str) {
    setString(str);
    this-> type = "HexString";
    count++;
    this-> id = count;
}

// конструктор копирования
HexString(const HexString &str);

// сеттер поля m_string
void setString(char *string);

// перевод hex строки в десятичное число
void hexToDec();

// перегрузка оператора >
bool operator > (const HexString &str) {
    return (length > str.length);
}

// перегрузка оператора ==
bool operator == (HexString &str) {
    return (this->getString() == str.getString());
}

// получение информации о строке
void info();

};

#endif //PRACTICE2_HEXSTRING_H

```

//HexString.cpp

```

#include <cmath>
#include "HexString.h"

HexString::HexString(const HexString &str) {
    m_string = str.m_string;
    length = str.length;
    isPositive = str.isPositive;

    count++;
    id = count;
}

HexString::~HexString() {
    cout << "Destructor HexString" << endl;
    delete m_string;
}

int HexString::char2dec(char hexChar) {
    if ((hexChar >= '0') && (hexChar <= '9'))
        return hexChar - '0';
    else if ((hexChar >= 'A') && (hexChar <= 'F'))

```

```

        return hexChar - 'A' + 10;
    else if ((hexChar >= 'a') && (hexChar <= 'f'))
        return hexChar - 'a' + 10;
}

// заполнение поля m_string
void HexString::setString(char *string) {
    int i = 0, j;
    bool flag = false;

    while (string[i] != '\0') {
        j = 0;
        while (hexNumbers[j] != '\0') {
            if (hexNumbers[j] == string[i]) {
                flag = true;
                break;
            }
            j++;
        }

        if (flag == false) {
            char *empty = "";
            m_string = empty;
            length = 0;
            return;
        }
        flag = false;
        i++;
    }

    if (i != 0) {
        length = i;
        m_string = string;

        if (m_string[0] == '+') {
            isPositive = true;
            length--;
        } else if (m_string[0] == '-') {
            isPositive = false;
            length--;
        } else {
            isPositive = true;
        }
    }
}

// перевод hex строки в десятичное число
void HexString::hexToDec() {
    int i;
    int l = length;
    int decimal = 0;
    i = (m_string[0] == '+') ? 1 : ((m_string[0] == '-') ? 2 : 0);

    if (i == 2) {
        cout << "Impossible to convert to unsigned int" << endl;
        return;
    }
}

```

```

while (m_string[i] != '\0') {
    decimal += char2dec(m_string[i]) * pow(16, --l);
    i++;
}
decimal *= isPositive ? 1 : -1;
cout << "Decimal: " << decimal << endl;
// return decimal;
}

// получение информации о строке
void HexString::info() {
    cout << "HexString " << id << ": " << m_string << endl;
    cout << "Length: " << length << endl;
    cout << endl;
}

```

//IdentifiedString.h

```

#ifndef PRACTICE2_IDENTIFIEDSTRING_H
#define PRACTICE2_IDENTIFIEDSTRING_H
#include "String.h"

class IdentifiedString : public String {
public:
    // дефолтный конструктор
    IdentifiedString() : String() {
        this->type = "IdentifiedString";
        m_string = new char[100];
    }

    // конструктор принимающий на вход строку
    IdentifiedString(char *str) {

        // проверка введенной строки на наличие пробелов, если есть пробелы то
        строка будет пустой
        setString(str);
        count++;
        this-> id = count;
        this->type = "IdentifiedString";
    }

    // конструктор копирования
    IdentifiedString(const IdentifiedString &str);

    // деструктор
    ~IdentifiedString();

    // проверка строки, если содержит пробелы записать пустую строку
    void setString(char *string);

    // перевод буквенной части числа в нижн. регистр
    void toLower();

    // перегрузка оператора индексации [ ]
    char &operator[](int index);

    // перегрузка оператора >=
    bool operator >= (const IdentifiedString &str);
}

```

```
};
```

```
#endif //PRACTICE2_IDENTIFIEDSTRING_H
```

//IdentifiedString.cpp

```
#include "IdentifiedString.h"
```

```
// конструктор копирования
```

```
IdentifiedString::IdentifiedString(const IdentifiedString &str) {  
    m_string = str.m_string;  
    length = str.length;  
    count++;  
    id = count;  
}
```

```
// деструктор
```

```
IdentifiedString::~~IdentifiedString() {  
    cout << "Destructor IdentifiedString" << endl;  
    delete m_string;  
}
```

```
// проверка строки
```

```
void IdentifiedString::setString(char *string) {  
    int i = 0;  
    bool containSpace = false;  
  
    while (string[i] != '\0') {  
        if (string[i] == ' ') {  
            containSpace = true;  
            break;  
        }  
        i++;  
    }  
    if (containSpace == false) {  
        m_string = string;  
        length = i;  
    } else {  
        m_string = "";  
        length = 0;  
    }  
}
```

```
// перевод буквенной части числа в нижн. регистр
```

```
void IdentifiedString::toLower() {  
    int i = 0;  
    while (m_string[i] != '\0') {  
        if ('A' <= m_string[i] && m_string[i] <= 'F')  
            m_string[i] += 32;  
        i++;  
    }  
}
```

```
// перегрузка оператора индексации >=
```

```
bool IdentifiedString::operator >= (const IdentifiedString &str) {  
    return (length >= str.length);  
}
```



```
// перепрыжка оператора []
char &IdentifiedString::operator[](int index) {
    cout << m_string[index];
    return m_string[index];
}
```

//Menu.h

```
#if !defined(MENU_H)
#define MENU_H

#include <iostream>
#include "string.h"

#include "..\helpFunctions\functions.h"

using namespace std;

// class HelpFunctions {
//     public:
//         static void printPropertiesFunc (String ** list, int order );
// };

// void HelpFunctions::printPropertiesFunc (String ** list, int order ) {
//     system("cls");
//     list[order]->printProperties(list[order]->type);
//     system("pause");
// }

int getVariant(int);

void printMainMenu();
void printInizializationMenu();
void printTestingMenu();
void printAllStrings(String**, int);
void exitMenu(int);
int printInizializationMenuChooseType(int );

#endif // menu.h
```

//Menu.cpp

```
#include <iostream>
#include <string.h>
#include "String.h"
#include <stdio.h>
#include "IdentifiedString.h"
#include "HexString.h"
#include "menu.h"

using namespace std;
```

```

void printAllStrings(String ** list, int N) {
    int getVariant(int);
    int l = 0;
    IdentifiedString* p;
    HexString* pl;

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

        int var = printInizializationMenuChooseType(i);

        // обработка строки ввода
        string temp;
        getline(cin, temp);
        int tmp = temp.length();
        char* str;
        str = new char[tmp + 1];
        strcpy(str, temp.c_str());

        // создание объекта в зависимости от выбранного типа
        if (var == 1) {
            IdentifiedString* p;
            p = new IdentifiedString(str);
            list[l] = p;

        } else {
            HexString* pl;
            pl = new HexString(str);
            list[l] = pl;

        }

        l++;

    }

    // cout << list[0]->getString() << endl;
    // list[0]->getStringLength();
    // cout << list[1]->getString() << endl;
    // list[1]->getStringLength();
    // system("pause");

}

void printMainMenu() {

    system("cls"); // очищаем экран

    string menu[] = {
        "What do you want to do?",
        "1. Initialization",
        "2. Testing" };

    printMenu(menu, 3);
    exitMenu(3);
}

void printInizializationMenu() {

```

```

    system("cls"); // очищаем экран

    string menu[] = {
        "What do you want to do?",
        "1. Set the number of elements",
        "2. Set initial values" };

    printMenu(menu, 3);
    exitMenu(3);
}

int printInizializationMenuChooseType(int i) {
    int getVariant(int);

    system("cls"); // очищаем экран
    cout << "Chosse type for" << " string #" << i + 1 << endl;
    cout << "1. " << "Identifier string" << endl;
    cout << "2. " << "Hexadecimal string" << endl;
    cout << "> ";

    int var = getVariant(2);

    cout << endl << "Enter string" << endl;
    cout << "> ";

    return var;
}

void printTestingMenu() {

    system("cls"); // очищаем экран

    string menu[] = {
        "What do you want to do?",
        "1. String",
        "2. Identifier string",
        "3. Hexadecimal string" };

    printMenu(menu, 4);
    exitMenu(4);
    // cout << "4. Set operands" << endl;

}

// class menus
void printStringMenu() {

    system("cls"); // очищаем экран

    string menu[] = {
        "What do you want to do?",
        "1. To lower",
        "2. Choose index of string" };

    printMenu(menu, 3);
    // cout << "3. Hexadecimal string" << endl;
}

void printHexMenu() {
    // exitMenu();
}

```

```
void printIdentifiedMenu() {
    // exitMenu();
}
```

//Main.h

```
#if !defined(MAIN_H)
#define MAIN_H

#include <iostream>
#include "String.h"

#include "..\helpFunctions\functions.h"

String ** initializationMenu(String**);
void testingMenu(String ** list);
void printTypesOfStrings(String ** list, int var);
int typeCheck(String ** list, string var, int i, int countOfStrings);
void objectMenu (String ** list, string str, int numberOfElement);
int idenStr(String ** list, int order);
int hexStr(String ** list, int order);

#endif // main.h
```

//Main.cpp

```
#include <iostream>
#include "HexString.h"
#include "menu.h"
#include "main.h"

// подключаем функции-помощники
#include "..\helpFunctions\functions.cpp"

using namespace std;

int String::count = 0;

int masLength = 0;
int initializedMass = 0;

int main() {
    int variant;
    int flag = 0;
    String** listOld;
    String** list;

    // for testing

    // char lol[11] = "-10AB";
    // char lol1[12] = "11AAADDDDF22";
    // IdentifiedString str1(lol);
```

```

// HexString str2(lol);
// str1[1];
// str1.info();
// str1.hexToDec();
// str1.getString();
// cout << (str1 == str2) << endl;
// system("pause");

do {
    printMainMenu();
    variant = getVariant(3);
    switch (variant) {
        case 1:
            list = initializationMenu(listOld);
            break;

        case 2:

            if (initializedMass == 1) {
                testingMenu(list);

            } else {
                cout << "You didn't enter any strings" << endl;
                system("pause");
            }

            break;

        case 3:
            break;
    }

} while(variant != 3);

return 0;
}

String ** initializationMenu(String ** list) {
    int variant;
    int flagIn = 0;

    do {
        printInizializationMenu();
        variant = getVariant(3);

        static int count = 0;
        // String** list;

        switch (variant) {
            case 1: {
                if (count != 0) {
                    cout << "You have already initialized the array" << endl;
                    system("pause");
                    break;
                } else {
                    system("cls");
                    cout << "Enter list size" << endl << ">";
                    count = getVariant(10);
                    cout << count << endl;
                    masLength = count;
                    list = new String*[count];
                }
            }
        }
    }
}

```

```

        break;
    }
}
case 2:
    if (count == 0) {
        cout << "Initialized the array first" << endl;
        system("pause");
        break;
    } else {
        printAllStrings(list, count);
        initializedMass++;
    }

    flagIn++;
    break;

case 3:
    break;
}

} while(variant != 3);
return list;
}

void testingMenu(String ** list) {
    int variant;
    do {
        printTestingMenu();
        variant = getVariant(4);

        if (variant != 4) printTypesOfStrings(list, variant);

    } while(variant != 4);
    // cout << "End" << endl;
    // system("pause");
}

void printTypesOfStrings(String ** list, int var) {
    // cout << "Start" << list[0]->getString() << endl;
    // system("pause");
    int countOfStrings = 1;
    string str;
    switch (var) {
        case 1:
            str = "String";
            break;
        case 2:
            str = "IdentifiedString";
            break;
        case 3:
            str = "HexString";
            break;
    }

    do {
        countOfStrings = 1;
        system("cls");
        for (int j = 0; j < masLength; j++) {
            // cout << "Start loop " << flag << " " << j << " var" << var << endl;
            // system("pause");
            countOfStrings = typeCheck(list, str, j, countOfStrings);
        }
    }
}

```

```

    }
    exitMenu(countOfStrings);
    var = getVariant(countOfStrings);

    if (var != countOfStrings) {
        objectMenu(list, str, var);
    }

} while (var != countOfStrings);

}

void objectMenu (String ** list, string str, int numberOfElement) {

    int var;
    int counter = 0;
    int order = 0;

    for(int j = 0; j < masLength; j++) {
        if (list[j]->type == str) {
            order = j;
            ++counter;
            if (counter == numberOfElement) break;
        }
    }

    if (str == "HexString") {
        do {
            system("cls");
            var = hexStr(list, order);
        } while (var != 3);

    } else if (str == "IdentifiedString") {
        do {
            system("cls");
            var = idenStr(list, order);
        } while (var != 4);

    }

}

int typeCheck(String ** list, string var, int i, int count) {
    // cout << list[i]->getString() << endl;
    // cout << list[i]->type << var << endl;
    if (var == list[i]->type) {
        cout << count++ << ". " << list[i]->type << " #" << i+1 << endl;
    }
    return count;
}

int idenStr(String ** list, int order) {

```

```

int var;

string menu[] = {
    "What do you want to do?",
    "1. Print properties",
    "2. To lower register",
    "3. Choose index of string" };

printMenu(menu, 4);
exitMenu(4);

var = getVariant(4);

switch (var) {
    case 1:
        system("cls");
        list[order]->printProperties(list[order]->type);
        system("pause");
        break;
    case 2:
        system("cls");
        list[order]->toLower();
        cout << "String is " << list[order]->getString() << endl;
        system("pause");
        break;
    case 3:
        system("cls");
        cout << "choose index from 0 to " << list[order]->getStringLength() <<
endl << ">";
        int index = getVariant(list[order]->getStringLength());
        system("cls");
        cout << "Elem #" << index << " - " << (list[order]-
>getString())[index-1] << endl;

        system("pause");
        break;
}

return var;
}

int hexStr(String ** list, int order) {

```

```

int var;

string menu[] = {
    "What do you want to do?",
    "1. Print properties",
    "2. Heximal to decimal" };

printMenu(menu, 3);
exitMenu(3);

var = getVariant(3);
system("cls");
switch (var) {
    case 1:
        list[order]->printProperties(list[order]->type);
        break;

    case 2:

```



```

        list[order]->hexToDec();
        break;

    }
    system("pause");

    return var;
}

```

Результат работы программы:

Рисунок 1: Основное меню.

Рисунок 2: Инициализация строк(меню).

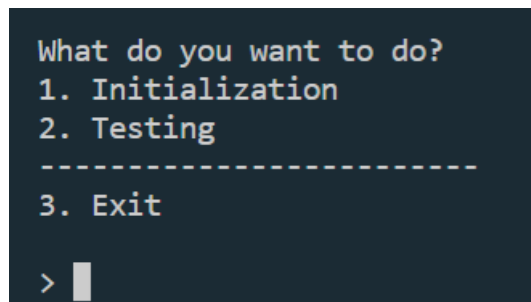
Рисунок 3: Инициализация строк(выбор типа строки и присвоение ей значения).

Рисунок 4: Тестирование (выбор типа строки для дальнейшего теста).

Рисунок 5: Тестирование (выбор экземпляра строки).

Рисунок 6: Тестирование (меню функций экземпляра строки).

Рисунок 7: Тестирование (вывод строки и её длины, тк строка была инициализирована не только 16 цифрами-числами, она превратилась в пустую строку).

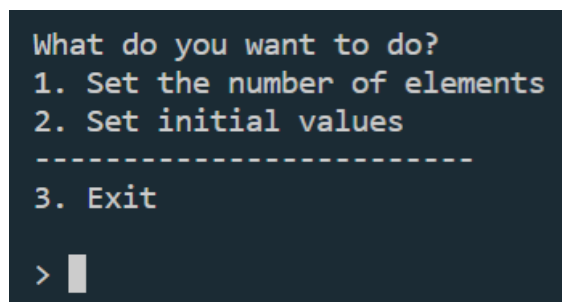


```

What do you want to do?
1. Initialization
2. Testing
-----
3. Exit
> 

```

Рисунок 1



```

What do you want to do?
1. Set the number of elements
2. Set initial values
-----
3. Exit
> 

```

Рисунок 2

```
Chosse type for string #2
1. Identifier string
2. Hexadecimal string
> 2

Enter string
> newStr
```

Рисунок 3

```
What do you want to do?
1. String
2. Identifier string
3. Hexadecimal string
-----
4. Exit

> 
```

Рисунок 4

```
1. HexString #2
2. HexString #3
3. HexString #4
4. HexString #5
-----
5. Exit

> 
```

Рисунок 5

```
What do you want to do?
1. Print properties
2. Heximal to decimal
-----
3. Exit

> 
```

Рисунок 6

```
HexString #3:  
Length: 0
```

```
Для продолжения нажмите любую клавишу . . . █
```

Рисунок 7