

**Колледж космического машиностроения и технологий**

Разработка библиотеки динамической компоновки**.**

**Сопровождающий документ**

**По МДК.01.01 «Системное программирование»**

**Тема: «Разработка библиотеки динамической компоновки»**

Выполнил студент

Соколов Ярослав Алексеевич

Группа П2-17

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_(Подпись)

\_\_\_\_\_\_\_\_\_\_\_\_ (Дата сдачи работы)

Проверил

преподаватель

Гусятинер Леонид Борисович

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_(Подпись)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_(Оценка)

Королев, 2020

[**Диаграмма классов**](#_s573dpdo44dz) **3**

[**Проектирование библиотеки динамической компоновки.**](#_hlvgxzrsv21g) **4**

# **Введение:**

Для операционных систем Microsoft Windows, перечисленных в разделе "Применяется к" большая часть функциональности операционной системы обеспечивается библиотеками динамической компоновки (DLL). Кроме того, при запуске программы на одной из этих операционных систем Windows большая часть функциональности программы может быть предоставлена библиотеками DLL. Например, некоторые программы могут содержать множество различных модулей, и каждый модуль программы автономный и распространяется в DLL.

Использование DLL способствует модуляризации кода, повторному использованию кода, эффективному использованию памяти и уменьшению занимаемого пространства диска. Таким образом, операционная система и программы загружаются быстрее, работают быстрее и занимают меньше дискового пространства на компьютере.

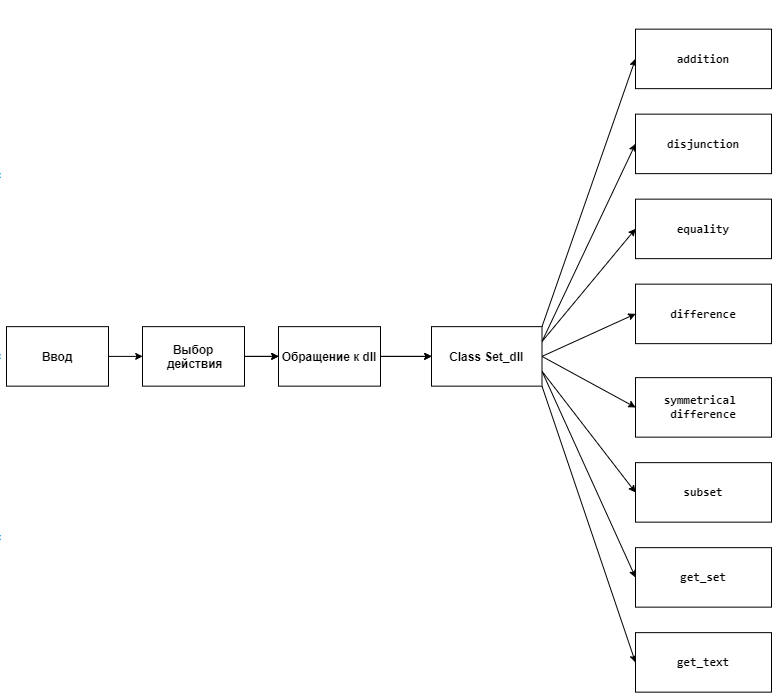
DLL — это библиотека, содержащая код и данные, которые могут использоваться одновременно более чем одной программой. Например, в операционных системах Windows Comdlg32 DLL выполняет общие функции, связанные с диалоговым окном. Таким образом, каждая программа может использовать функциональные возможности, содержащиеся в этой DLL для реализации открытого диалогового окна. Это помогает стимулировать повторное использование кода и эффективное использование памяти.

# **Диаграмма классов**

Описание диаграммы классов: Пользователь вводит данные, программа обращается к dll чтобы получить доступ к классу Set\_dll для того, чтобы вызвать процедуру для обработки действия.

Class Set\_dll содержит в себе процедуры:

* **get\_text**(Отображение результата)
* **get\_set**(Получение множеств от пользователя)
* **addition** (Объединение множеств)
* **disjunction** (Пересечение множеств)
* **equality** (Проверка на равенство множеств)
* **difference** (Разность множеств )
* **symmetrical\_difference** (Симметричная разность множеств)
* **subset** (Проверка является множество 'B' подмножеством 'A')



# Рисунок 1. Диаграмма класса

# **Проектирование библиотеки динамической компоновки.**

Список функций класса:

* **get\_text**(Отображение результата)
* **get\_set**(Получение множеств от пользователя)
* **addition**(Объединение множеств)
* **disjunction**(Пересечение множеств)
* **equality**(Проверка на равенство множеств)
* **difference**(Разность множеств )
* **symmetrical\_difference**(Симметричная разность множеств)
* **subset**(Проверка является множество 'B' подмножеством 'A')

**get\_text** (Отображение результата) описание: Функция получает множество представленное структурой. Если полученный массив из структуры пустой, то он выводит “Empty”

**get\_set** (Получение множеств от пользователя) описание: Получает из поля ввода строку. Возвращает множество.

**addition** (Объединение множеств) описание: Функция получает два множества представленные структурами. Возвращает множество.

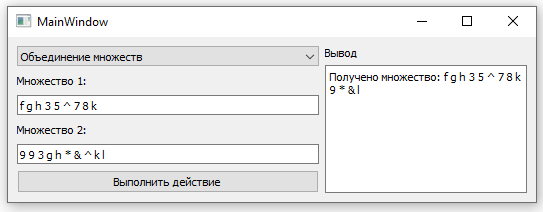


Рисунок 2. Результат работы функции **addition**

**disjunction** (Пересечение множеств) описание: Функция получает два множества представленные структурами. Возвращает множество.

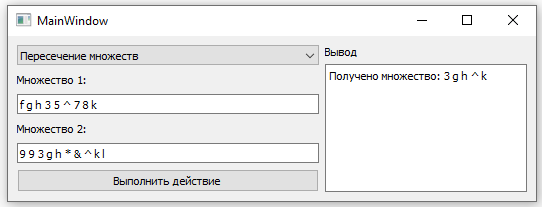


Рисунок 3. Результат работы функции **disjunction**

**equality** (Проверка на равенство множеств) описание: Функция получает два множества представленные структурами. Возвращает Bool значение.

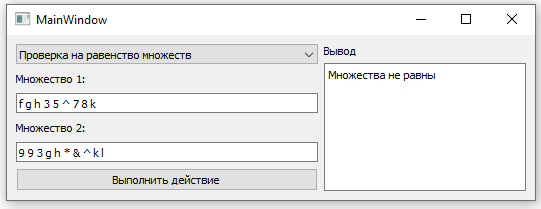


Рисунок 4. Результат работы функции **equality**

**difference** (Разность множеств ) описание: Функция получает две структуры, которые состоят из массива и его размера.

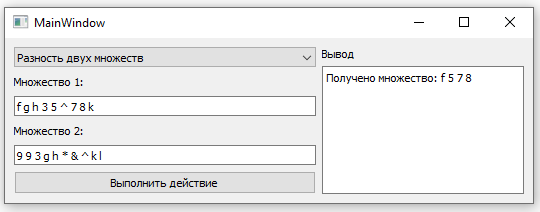


Рисунок 5. Результат работы функции **difference**

**symmetrical\_difference** (Симметричная разность множеств) описание: Функция получает две структуры, которые состоят из массива и его размера. Возвращает множество.

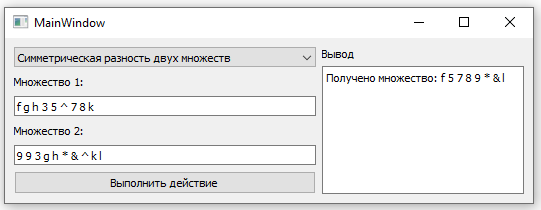


Рисунок 6. Результат работы функции **symmetrical\_difference**

**subset** (Проверка является множество 'B' подмножеством 'A') описание: Функция получает две структуры, которые состоят из массива и его размера. Возвращает Bool значение.

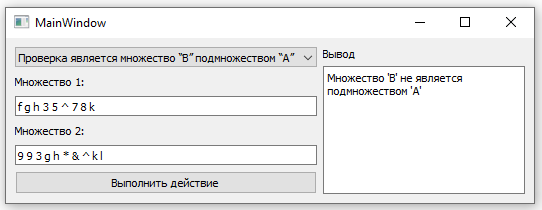


Рисунок 7. Результат работы функции **subset**

# **Заключение**:

С помощью DLL программу можно модулиризовать на отдельные компоненты. Например, бухгалтерская программа может быть продана по модулю. Каждый модуль может быть загружен в основную программу во время выполнения, если этот модуль установлен. Поскольку модули разделены, время загрузки программы быстрее, а модуль загружается только при запросе этой функциональности.

Кроме того, обновления легче применять к каждому модулю, не затрагивая другие части программы. Например, у вас может быть программа начисления заработной платы, и налоговые ставки меняются каждый год. Когда эти изменения изолированы от DLL, вы можете применить обновление без необходимости повторной сборки или установки всей программы.

Некоторые преимущества, предоставляемые при использовании DLL:

* Использование меньшего количества ресурсов  
  Когда несколько программ используют одну и ту же библиотеку функций, DLL может уменьшить дублирование кода, который загружается на диске и в физической памяти. Это может значительно повлиять на производительность не только программы, которая работает на переднем плане, но и других программ, работающих на операционной системе Windows.
* Способствует развитию модульной архитектуры  
  DLL помогает развивать модульные программы. Это помогает разрабатывать большие программы, требующие нескольких языковых версий или программы, требующие модульной архитектуры. Примером модульной программы является бухгалтерская программа, имеющая много модулей, которые могут быть динамически загружены во время выполнения.
* Упрощение развертывания и установки  
  Когда функция в DLL нуждается в обновлении или исправлении, развертывание и установка DLL не требует повторного соединения программы с DLL. Кроме того, если несколько программ используют одну и ту же DLL, то все несколько программ получат выгоду от обновления или исправления. Эта проблема может чаще возникать при использовании сторонней DLL, которая регулярно обновляется или исправляется.

**Приложение 1: set\_dll.h**

#ifndef SET\_DLL\_H

#define SET\_DLL\_H

#include "Set\_dll\_global.h"

*class* SET\_DLL\_EXPORT Set\_dll

{

*private*:

char \*set;

int size;

*public*:

Set\_dll();

Set\_dll addition(Set\_dll Set\_A, Set\_dll Set\_B);*//Объединение* *множеств*

Set\_dll disjunction(Set\_dll Set\_A, Set\_dll Set\_B);*//Пересечение* *множеств*

bool equality(Set\_dll Set\_A, Set\_dll Set\_B);

*//Проверка* *на* *равенство* *множеств*

Set\_dll difference(Set\_dll Set\_A, Set\_dll Set\_B);

*//Разность* *множеств*

Set\_dll symmetrical\_difference(Set\_dll Set\_A, Set\_dll Set\_B);*//Симметричная* *разность* *множеств*

bool subset(Set\_dll Set\_A, Set\_dll Set\_B);

*//* *Проверка* *является* *множество* *'B'* *подмножеством* *'A'*

Set\_dll get\_set(QString text);

QString get\_text(Set\_dll data);

void test();

};

#endif *//* *SET\_DLL\_H*

**Приложение 2: Set\_dll\_global.h**

#ifndef SET\_DLL\_GLOBAL\_H

#define SET\_DLL\_GLOBAL\_H

#include <QtCore/qglobal.h>

#if defined(SET\_DLL\_LIBRARY)

# define SET\_DLL\_EXPORT Q\_DECL\_EXPORT

#else

# define SET\_DLL\_EXPORT Q\_DECL\_IMPORT

#endif

#endif *//* *SET\_DLL\_GLOBAL\_H*

**Приложение 3: set\_dll..cpp**

#include "set\_dll.h"

#include <stdio.h>

#include <malloc.h>

#include <stdlib.h>

#include <QString>

#include <QStringList>

#include<QDebug>

Set\_dll::Set\_dll()

{

}

void Set\_dll::test(){

qDebug()<<"Hi";

}

Set\_dll Set\_dll::get\_set(QString text){

QRegExp rx(" ");

QStringList query = text.split(rx);

Set\_dll data;

query.removeDuplicates();

data.set = (char\*)malloc(query.size() \* *sizeof*(int));

data.size = query.size();

*for*(int i = 0; i < query.size(); i++){

data.set[i] = \*query.value(i).toUtf8().data();

qDebug() << data.set[i];

}

*return* data;

free(data.set);*//Очищаем* *память*

}

QString Set\_dll::get\_text(Set\_dll data){

QString text ="Получено множество: ";

*for*(int i = 0; i < data.size; i++) text = text + data.set[i]+" ";

*return* text;

}

Set\_dll Set\_dll::addition(Set\_dll Set\_A, Set\_dll Set\_B){*//* *Объединение* *множеств*

int k = 0;

Set\_dll data;

data.size=Set\_A.size;

*for* (int i = 0; i < Set\_B.size; i++){*//Вычисляем* *размер* *итогово* *моссива*

*for*(int j = 0; j < Set\_A.size; j++){

*if* (Set\_A.set[j] != Set\_B.set[i]) k++;

}

*if* (k == Set\_A.size){

data.size++;

}

k=0;

}

data.set = (char\*)malloc(data.size \* *sizeof*(int));*//Выделяем* *место* *под* *массив*

k = 0; data.size = Set\_A.size;

*for* (int i = 0; i < Set\_A.size; i++) data.set[i]=Set\_A.set[i];*//Заполняем* *массив* *первым* *множеством*

*for* (int i = 0; i < Set\_B.size; i++)*//Заполняем* *массив* *вторым* *множеством*

{

*for*(int j = 0; j < Set\_A.size; j++)

{

*if* (Set\_A.set[j] != Set\_B.set[i]) k++;

}

*if* (k == Set\_A.size)

{

data.set[data.size]=Set\_B.set[i];

data.size++;

}

k=0;

}

*return* data; *//Возвращаем* *результат*

free(data.set);*//Очищаем* *память*

}

Set\_dll Set\_dll::disjunction(Set\_dll Set\_A, Set\_dll Set\_B){*//Пересечение* *множеств*

Set\_dll data;

data.size = 0;

*for* (int i = 0; i < Set\_B.size; i++){*//Вычисляем* *размер* *итогово* *моссива*

*for*(int j = 0; j < Set\_A.size; j++){

*if* (Set\_A.set[j] == Set\_B.set[i]) data.size++;

}

}

data.set = (char\*)malloc(data.size \* *sizeof*(int));*//Выделяем* *место* *под* *моссив*

data.size = 0;

*for* (int i = 0; i < Set\_A.size; i++) data.set[i]=Set\_A.set[i];*//Заполняем* *моссив* *первым* *множеством*

*for* (int i = 0; i < Set\_B.size; i++)*//Заполняем* *моссив* *вторым* *множеством*

{

*for*(int j = 0; j < Set\_A.size; j++)

{

*if* (Set\_A.set[j] == Set\_B.set[i])

{

data.set[data.size]=Set\_B.set[i];

data.size++;

}

}

}

*return* data; *//Возвращаем* *результат*

free(data.set);*//Очищаем* *память*

}

bool Set\_dll::equality(Set\_dll Set\_A, Set\_dll Set\_B){*//Проверка* *на* *равенство* *множеств*

int k = 0;

*for* (int i = 0; i < Set\_B.size; i++){*//Проверям* *на* *равенство* *по* *элементно*

*for*(int j = 0; j < Set\_A.size; j++){

*if* (Set\_A.set[j] == Set\_B.set[i]) k++;

}

}

*return* (k == Set\_B.size && Set\_A.size == Set\_B.size);

}

Set\_dll Set\_dll::difference(Set\_dll Set\_A, Set\_dll Set\_B){*//Разность* *множеств*

int k = 0 ;

Set\_dll data;

data.size = 0;

*for* (int i = 0; i < Set\_A.size; i++){*//Вычисляем* *размер* *итогово* *моссива*

*for*(int j = 0; j < Set\_B.size; j++){

*if* (Set\_A.set[i] != Set\_B.set[j]) k++;

}

*if* (k == Set\_B.size) data.size++;

k=0;

}

data.set = (char\*)malloc(data.size \* *sizeof*(int));*//Выделяем* *место* *под* *моссив*

data.size = 0; k = 0;

*for* (int i = 0; i < Set\_A.size; i++)*//Заполняем* *моссив*

{

*for*(int j = 0; j < Set\_B.size; j++)

{

*if* (Set\_A.set[i] != Set\_B.set[j]) k++;

}

*if*(k == Set\_B.size){

data.set[data.size]=Set\_A.set[i];

data.size++;

}

k = 0;

}

*return* data; *//Возвращаем* *результат*

free(data.set);*//Очищаем* *память*

}

Set\_dll Set\_dll::symmetrical\_difference(Set\_dll Set\_A, Set\_dll Set\_B){*//Симетричная* *разность* *множеств*

int k = 0 ;

Set\_dll data;

data.size = 0;

*for* (int i = 0; i < Set\_A.size; i++){*//Вычисляем* *размер* *итогово* *моссива* *ч.1*

*for*(int j = 0; j < Set\_B.size; j++){

*if* (Set\_A.set[i] != Set\_B.set[j]) k++;

}

*if* (k == Set\_B.size) data.size++;

k=0;

}

*for* (int i = 0; i < Set\_B.size; i++){*//Вычисляем* *размер* *итогово* *моссива* *ч.2*

*for*(int j = 0; j < Set\_A.size; j++){

*if* (Set\_A.set[j] != Set\_B.set[i]) k++;

}

*if* (k == Set\_A.size) data.size++;

k=0;

}

data.set = (char\*)malloc(data.size \* *sizeof*(int));

data.size = 0; k = 0;

*for* (int i = 0; i < Set\_A.size; i++)*//Заполняем* *моссив* *первым* *множеством*

{

*for*(int j = 0; j < Set\_B.size; j++)

{

*if* (Set\_A.set[i] != Set\_B.set[j]) k++;

}

*if*(k == Set\_B.size){

data.set[data.size]=Set\_A.set[i];

data.size++;

}

k = 0;

}

*for* (int i = 0; i < Set\_B.size; i++)*//Заполняем* *моссив* *вторым* *множеством*

{

*for*(int j = 0; j < Set\_A.size; j++)

{

*if* (Set\_B.set[i] != Set\_A.set[j]) k++;

}

*if*(k == Set\_A.size){

data.set[data.size]=Set\_B.set[i];

data.size++;

}

k = 0;

}

*return* data; *//Возвращаем* *результат*

free(data.set);*//Очищаем* *память*

}

bool Set\_dll::subset(Set\_dll Set\_A, Set\_dll Set\_B){*//* *Проверка* *является* *множество* *'B'* *подмножеством* *'A'*

int k = 0;

*for* (int i = 0; i < Set\_A.size ; i++)*//Поэлементно* *сравниваем* *множества*

*for*(int j = 0; j < Set\_B.size ; j++){

*if*(Set\_A.set[i] == Set\_B.set[j]) k++;

}

*return* k == Set\_B.size;

}