

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

**Український державний університет науки і технологій**

Кафедра «Комп’ютерні інформаційні технології»

**Лабораторна робота №2**

**з дисципліни «Системи трансляцій»**

**на тему: «****Компіляція отриманого коду за допомогою клангу. Виведення результатів.»**

Виконав:

Студент гр. ПЗ2011

Кулик С.В.

Прийняв:

Разносілін В.В.

Дніпро, 2023

**Тема:** Компіляція отриманого коду за допомогою клангу. Виведення результатів.

**Мета:**

• Ознайомлення з процесом компіляції програмного коду за допомогою компілятора Clang.

• Оволодіння навичками виведення результатів компіляції для перевірки та налагодження коду.

**Опис роботи програми-шаблону**

Програма-шаблон є базовою структурою або основою для створення інших програм або систем. Вона надає загальну структуру, функціональність та інтерфейс, які можна налаштувати та розширити для виконання конкретних завдань.

Основна мета програми-шаблону - спростити розробку програмних продуктів шляхом надання основного фундаменту, який можна використовувати знову і знову. Зазвичай вона містить загальні алгоритми, структури даних, функції та компоненти, які можуть бути перевикористані та налаштовані залежно від потреб конкретного проекту.

Програма складається з трьох файлів: main.cpp, stuff.cpp та stuff.h.

**Файл stuff.h** - структура, яка представляє дані про товари , а також є дві нереалізовані функції для фільтрації та сортування цих даних.

**Файл stuff.cpp**- реалізація функцій сортування та фільтрації. Цей файл взаємодіє з основною програмою, замінюючи тіло функцій на власний код у вигляді dll, що змінює логіку фільтрації та сортування.

**Файл main.cpp** - робота з таблицею, у вигляді csv-файлу. Спочатку програма відкриває цей файл, зчитує дані з нього.

Потім дані про товари фільтруються згідно умов у функції фільтрації, а потім сортуються, використовуючи функцію порівняння для визначення умов сортування, оброблені дані про товари записуються у вихідний файл.

**Текст програми**

1. **Main.cpp**

#include "stdlib.h"

#include "stdio.h"

#include <iostream>

#include <fstream>

#include "string"

#include "vector"

#include "stuff.h"

#include <windows.h>

using namespace std;

vector<string> SplitString(string s) {

vector<string> res;

string::const\_iterator a = s.begin();

string::const\_iterator b = s.end();

string cur;

bool inQuotes = false;

while (a != b) {

switch (\*a) {

case ' ':

case '\t': {

if (inQuotes) {

cur.push\_back(\*a);

} else {

if (!cur.empty()) {

res.push\_back(cur);

cur.clear();

}

}

break;

}

case '"': {

inQuotes = !inQuotes;

break;

}

default: {

cur.push\_back(\*a);

break;

}

}

++a;

}

if (!cur.empty()) {

res.push\_back(cur);

}

return res;

}

Good GoodFromString(string str) {

vector<string> t = SplitString(str);

Good res;

strcpy\_s(res.name, t[0].c\_str());

strcpy\_s(res.description, t[1].c\_str());

res.price = atof(t[2].c\_str());

strcpy\_s(res.producer, t[3].c\_str());

strcpy\_s(res.country, t[4].c\_str());

res.article = atoi(t[5].c\_str());

res.countInStock = atoi(t[6].c\_str());

res.countOrder = atoi(t[7].c\_str());

res.availability = t[8] == "true";

return res;

}

string GoodToString(Good good) {

string res;

res = res + "\"" + good.name + "\" ";

res = res + "\"" + good.description + "\" ";

res += to\_string(good.price) + " ";

res = res + "\"" + good.producer + "\" ";

res = res + "\"" + good.country + "\" ";

res += to\_string(good.article) + " ";

res += to\_string(good.countInStock) + " ";

res += to\_string(good.countOrder) + " ";

if (good.availability) {

res += "\"true\"";

} else {

res += "\"false\"";

}

return res;

}

vector<Good> LoadFromFile(string fname) {

vector<Good> res;

ifstream file(fname);

string str;

while (getline(file, str)){

res.push\_back(GoodFromString(str));

}

return res;

}

void SaveToFile(string fname, vector<Good> data){

ofstream F(fname);

if (!F.is\_open())

return;

for(int i=0; i<data.size(); i++)

F << GoodToString(data[i]) << endl;

F.close();

}

void Sort(vector<Good>& data){

int i=1;

while (i < data.size()) {

Good x = data[i];

int j = i-1;

//while (j >= 0 && data[j].price > x.price) {

bool val = greater\_than(data[j].name, data[j].price, data[j].producer,

data[j].country, data[j].article, data[j].countInStock, data[j].countOrder, data[j].availability,

x.name, x.price, x.producer, x.country,

x.article, x.countInStock, x.countOrder, x.availability);

while (j >= 0 && val) {

data[j+1] = data[j];

j--;

}

data[j+1] = x;

i++;

}

}

void BubbleSort(vector<Good>& data) {

int n = data.size();

bool swapped;

for (int i = 0; i < n - 1; i++) {

swapped = false;

for (int j = 0; j < n - i - 1; j++) {

//if (data[j].price > data[j + 1].price) {

bool val = greater\_than(data[j].name, data[j].price, data[j].producer,

data[j].country, data[j].article, data[j].countInStock, data[j].countOrder, data[j].availability,

data[j + 1].name, data[j + 1].price, data[j + 1].producer, data[j + 1].country,

data[j + 1].article, data[j + 1].countInStock, data[j + 1].countOrder, data[j + 1].availability);

if (val) {

swap(data[j], data[j + 1]);

swapped = true;

}

}

if (!swapped)

break;

}

}

const string input\_data = "D:\\data.csv";

const string output\_data = "D:\\result.csv";

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

std::cout << argv[0] << endl;

//std::cout << "my directory is " << ExePath() << "\n";

vector<Good> data = LoadFromFile(input\_data);

cout << data.size() << "Goods loaded" << endl;

vector<Good> res;

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

if (filter(data[i].name, data[i].description, data[i].price, data[i].producer, data[i].country,

data[i].article, data[i].countInStock, data[i].countOrder, data[i].availability)) {

res.push\_back(data[i]);

}

}

cout << res.size() << "Goods selected" << endl;

BubbleSort(res);

SaveToFile(output\_data, res);

return 0;

}

1. **Stuff.cpp**

#include "string"

#include "stuff.h"

bool \_\_cdecl filter(GoodName name, Description description, float price, ProducerName producer,

CountryName country, int article, int countInStock, int countOrder, bool availability){

return wcscmp(L"country", L"Ουκρανία") == 0;

}

bool \_\_cdecl greater\_than(GoodName name1, float price1, ProducerName producer1,

CountryName country1, int article1, int countInStock1, int countOrder1, bool availability1,

GoodName name2, float price2, ProducerName producer2,

CountryName country2, int article2, int countInStock2, int countOrder2, bool availability2){

return name1 > name2;

}

1. **Stuff.h**

const int MaxGoodName = 256;

const int MaxCountryName = 256;

const int MaxProducerName = 256;

const int MaxDescriptionSize = 256;

typedef char GoodName[MaxGoodName];

typedef char CountryName[MaxCountryName];

typedef char ProducerName[MaxProducerName];

typedef char Description[MaxDescriptionSize];

struct Good {

GoodName name;

Description description;

float price;

ProducerName producer;

CountryName country;

int article;

int countInStock;

int countOrder;

bool availability;

};

bool \_\_cdecl filter(GoodName name, Description description, float price, ProducerName producer,

CountryName country, int article, int countInStock, int countOrder, bool availability);

bool \_\_cdecl greater\_than(GoodName name1, float price1, ProducerName producer1,

CountryName country1, int article1, int countInStock1, int countOrder1, bool availability1,

GoodName name2, float price2, ProducerName producer2,

CountryName country2, int article2, int countInStock2, int countOrder2, bool availability2

);

**Шаблон для заміни коду**

Шаблон для заміни коду містить опис функції фільтрування та сортування. Параметри цих функцій були перейменовані для зручної роботи з dll кодом у програмі.

При створенні реалізації (тіла функції) за допомогою нашого компілятора замінюють у шаблоні замість рядків-заглушок ;%%FILTER%% та ;%%ORDER%%. Також у dll коді шаблону можна побачити дві заглушки ;%%FILTER CONST%% та ;%%ORDER CONST%%, які використовуються для опису рядкових констант, необхідних для коректної роботи з рядками.

**DLL код шаблону**

; ModuleID = 'stuff.cpp'

source\_filename = "stuff.cpp"

target datalayout = "e-m:w-p270:32:32-p271:32:32-p272:64:64-i64:64-f80:128-n8:16:32:64-S128"

; Function Attrs: mustprogress nofree norecurse nosync nounwind readnone willreturn uwtable

define dso\_local noundef zeroext i1 @"?filter@@YA\_NQEAD0M00HHH\_N@Z"(ptr nocapture noundef readnone %name, ptr nocapture noundef readnone %description, float noundef %price, ptr nocapture noundef readnone %producer, ptr nocapture noundef readnone %country, i32 noundef %article, i32 noundef %countInStock, i32 noundef %countOrder, i1 noundef zeroext %availability) local\_unnamed\_addr #0 {

;filter

%var.0 = fcmp ogt float %price, 0x4049000000000000

%var.1 = icmp sgt i32 %countInStock, 50

%var.2 = and i1 %var.0, %var.1

ret i1 %var.2

;filter

}

; Function Attrs: argmemonly mustprogress nofree nounwind readonly willreturn

declare dso\_local i32 @strcmp(ptr nocapture noundef, ptr nocapture noundef) local\_unnamed\_addr #1

; Function Attrs: argmemonly mustprogress nofree nounwind readonly willreturn uwtable

define dso\_local noundef zeroext i1 @"?greater\_than@@YA\_NQEADM00HHH\_N0M00HHH1@Z"(ptr nocapture noundef readonly %l.name, float noundef %l.price,

ptr nocapture noundef readnone %l.producer, ptr nocapture noundef readnone %l.country, i32 noundef %l.article, i32 noundef %l.countInStock,

i32 noundef %l.countOrder, i1 noundef zeroext %l.availability, ptr nocapture noundef readonly %r.name, float noundef %r.price,

ptr nocapture noundef readnone %r.producer, ptr nocapture noundef readnone %r.country, i32 noundef %r.article, i32 noundef %r.countInStock,

i32 noundef %r.countOrder, i1 noundef zeroext %r.availability) local\_unnamed\_addr #0 {

;greater\_than

%var.0 = icmp sgt i32 %l.countInStock, %r.countInStock

%var.1 = icmp sgt i32 %r.article, %l.article

%var.2 = and i1 %var.0, %var.1

ret i1 %var.2

;greater\_than

}

attributes #0 = { mustprogress nofree norecurse nosync nounwind readnone willreturn uwtable "frame-pointer"="none" "min-legal-vector-width"="0" "no-trapping-math"="true" "stack-protector-buffer-size"="8" "target-cpu"="x86-64" "target-features"="+cx8,+fxsr,+mmx,+sse,+sse2,+x87" "tune-cpu"="generic" }

attributes #1 = { argmemonly mustprogress nofree nounwind readonly willreturn "frame-pointer"="none" "no-trapping-math"="true" "stack-protector-buffer-size"="8" "target-cpu"="x86-64" "target-features"="+cx8,+fxsr,+mmx,+sse,+sse2,+x87" "tune-cpu"="generic" }

attributes #2 = { mustprogress nofree norecurse nosync nounwind readnone willreturn uwtable "frame-pointer"="none" "min-legal-vector-width"="0" "no-trapping-math"="true" "stack-protector-buffer-size"="8" "target-cpu"="x86-64" "target-features"="+cx8,+fxsr,+mmx,+sse,+sse2,+x87" "tune-cpu"="generic" }

!llvm.linker.options = !{!0, !1, !2, !3, !4, !5}

!llvm.module.flags = !{!6, !7, !8}

!llvm.ident = !{!9}

!0 = !{!"/FAILIFMISMATCH:\22\_MSC\_VER=1900\22"}

!1 = !{!"/FAILIFMISMATCH:\22\_ITERATOR\_DEBUG\_LEVEL=0\22"}

!2 = !{!"/FAILIFMISMATCH:\22RuntimeLibrary=MT\_StaticRelease\22"}

!3 = !{!"/DEFAULTLIB:libcpmt.lib"}

!4 = !{!"/FAILIFMISMATCH:\22\_CRT\_STDIO\_ISO\_WIDE\_SPECIFIERS=0\22"}

!5 = !{!"/FAILIFMISMATCH:\22annotate\_string=0\22"}

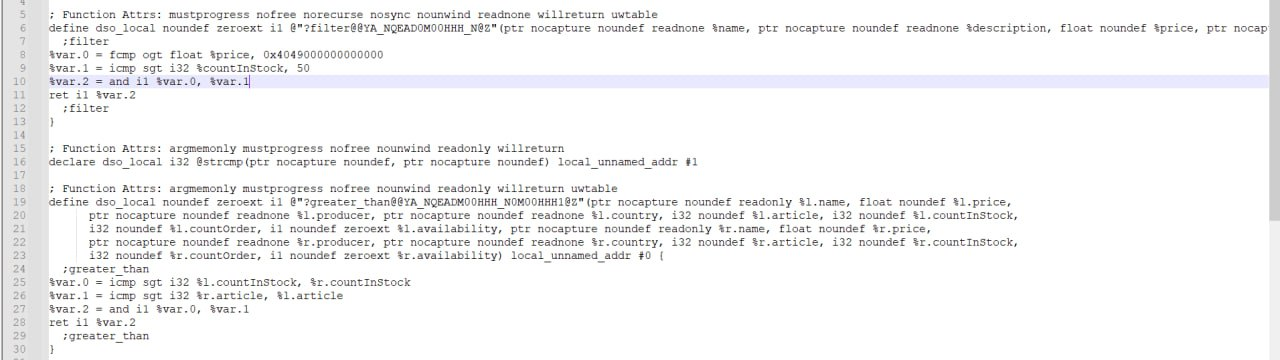
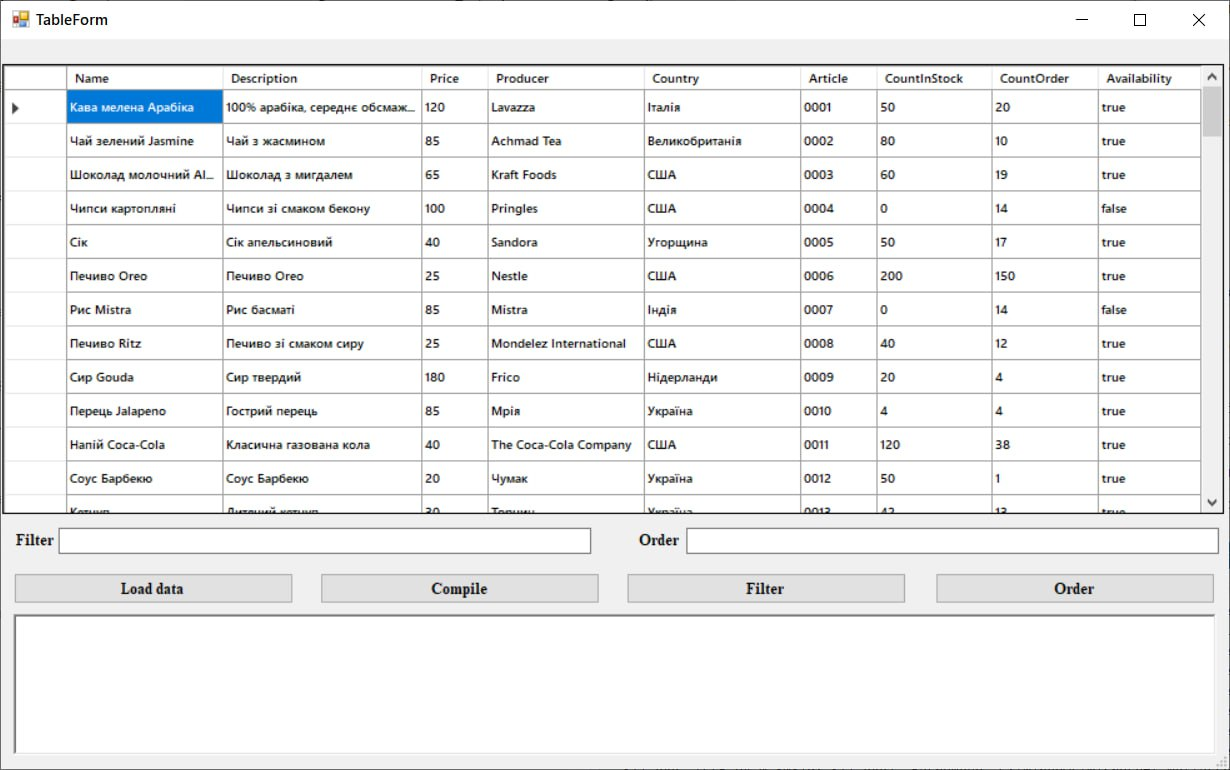
!6 = !{i32 1, !"wchar\_size", i32 2}

!7 = !{i32 7, !"PIC Level", i32 2}

!8 = !{i32 7, !"uwtable", i32 2}

!9 = !{!"clang version 15.0.1"}

**Результати роботи програми**

**Висновок**

Компілятор Clang є одним з популярних компіляторів в середовищі розробки програмного забезпечення. Він підтримує багато мов програмування, зокрема C, C++, Objective-C та Objective-C++. Основна його перевага полягає у швидкості компіляції та точності діагностики помилок.

Виведення результатів компіляції за допомогою Clang є важливою частиною процесу розробки програмного забезпечення. Його різноманітні параметри та опції дозволяють контролювати процес компіляції, зберігати скомпільований код, виводити попередження та помилки, а також забезпечувати можливість налагодження. Оволодіння цими навичками є важливим для ефективного розробника програмного забезпечення.