Лабораторна робота№2 РОБОТА ЗІ СТЕКОМ

**Мета:** продемонструвати причини руйнування стеку.

**Індивідуальне завдання:** використовуючи компілятори vc від MS Visual Studio та GCC, створити програму, що демонструє наявність проблеми, де переповнення змінної-масиву на рівні стека в функції може привести до руйнування стеку - при спробі доступа до недійного адресу в пам'яті. Довести, що дана недійсна адреса в пам'яті пов'язана з даними, що записуються до масиву.

ХІД РОБОТИ

Для наглядної демонстрації руйнування стеку в обох сферах розробки було використано наступний код:

/\*

* main.cpp

\*

* Created on: 28 февр. 2019 г.
* Author: pavlova-mv

\*/

//#define \_CRT\_SECURE\_NO\_WARNINGS #include <stdio.h>

#include <cstdlib> #include <cstring>

#define DESTROY\_VALUE\_1 10000

#define DESTROY\_VALUE\_2 10000

void checkStackDestroy() {

char buffer[DESTROY\_VALUE\_1 \* DESTROY\_VALUE\_2]; bool value = buffer[0] > 0;

}

void checkStackInit(const char\* src) { char buffer[5];

strcpy(buffer, src);

}

void checkStackSafeInit(const char\* src) { char buffer[5];

strcpy\_s(buffer, src);

}

void checkStackUndefined() {

int stack[DESTROY\_VALUE\_1];

printf("stack info: %d", stack[DESTROY\_VALUE\_1 + 1]);

}

int main() {

char\* line = "Check Line";

checkStackDestroy(); checkStackUndefined(); checkStackInit(line); checkStackSafeInit(line); system("pause");

return 0;

}

Першу частину лабораторної роботи було виконано у сфері розробки Visual Studio 2015. Спочатку проведемо запуск усіх функцій ігноруючи /GS. На рис.1 наведено дизасембльований код при такому запуску.

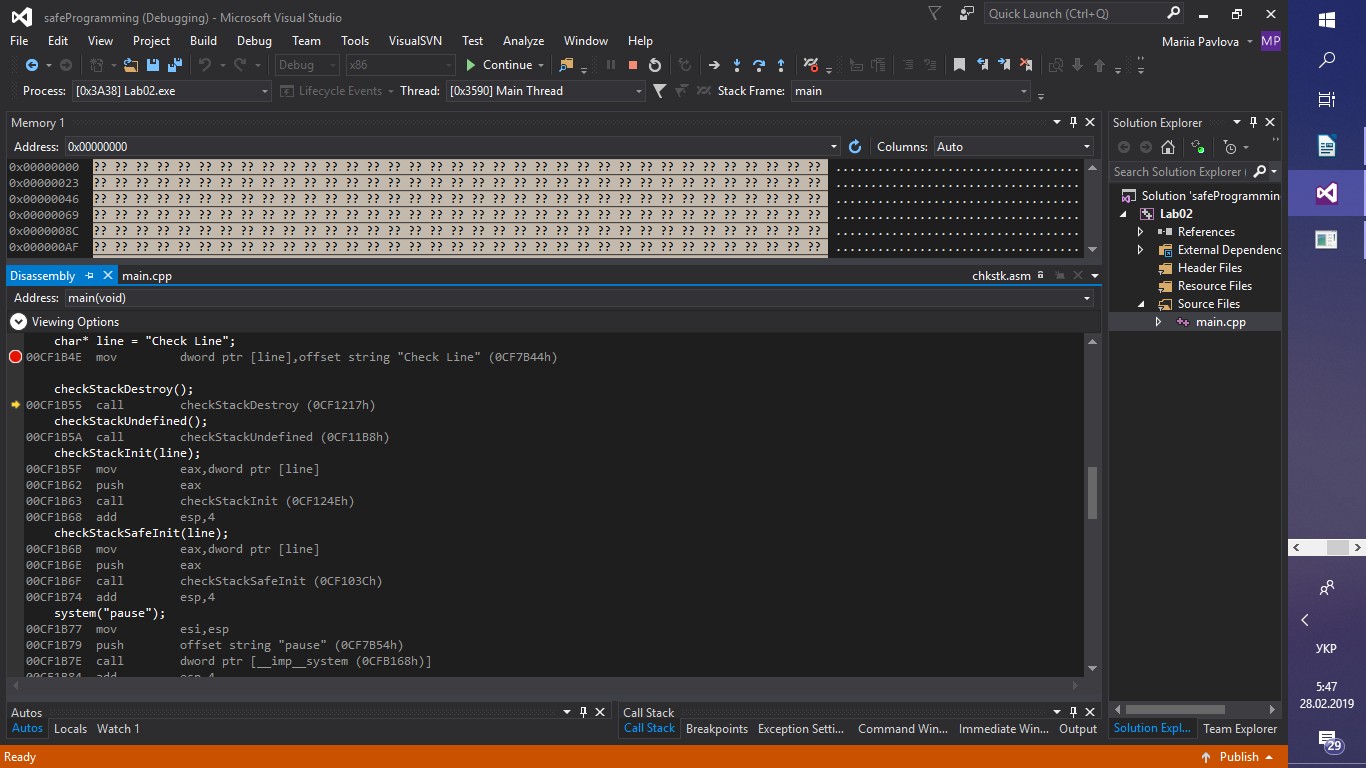


Рисунок 1 — Ассемблерні команди

Оскільки перевірка стеку ігнорувалася визвалася виключна ситуація переповнення стеку (рис.2).

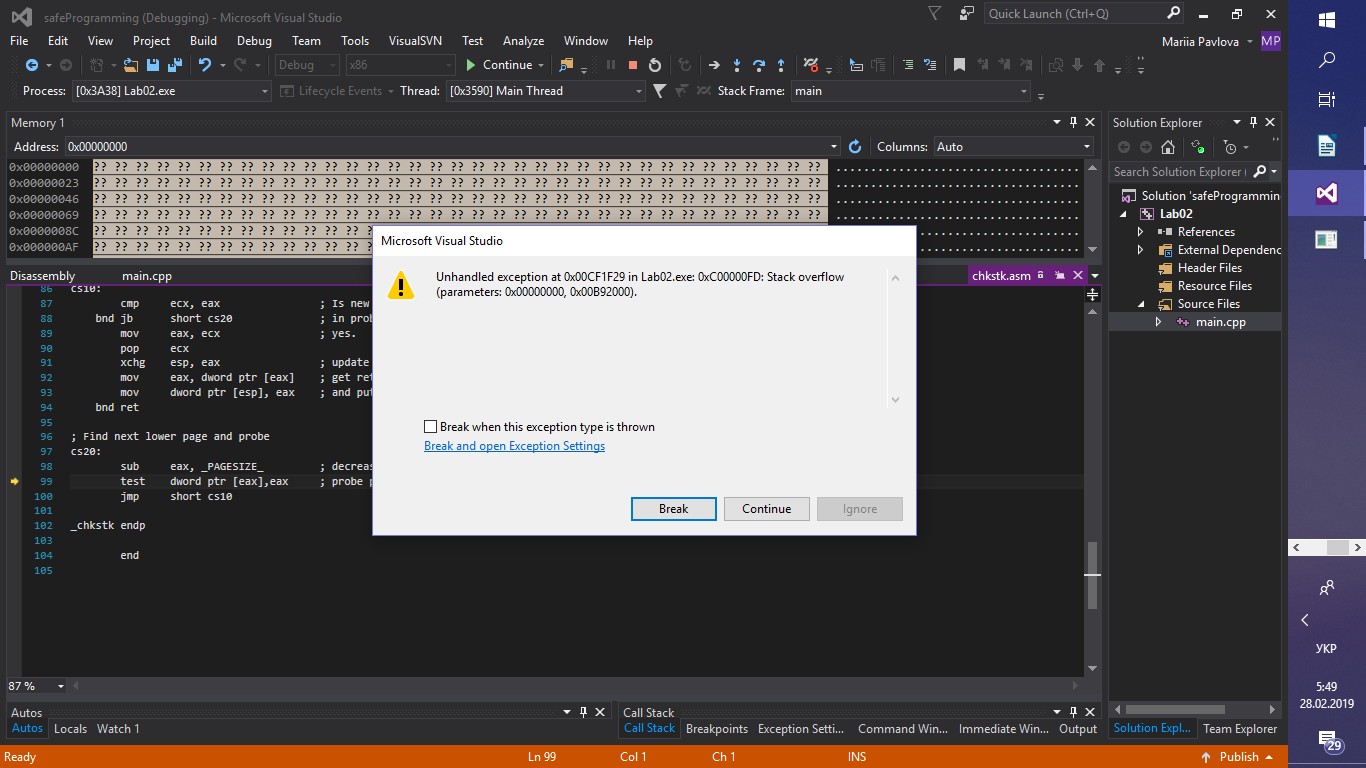


Рисунок 2 — Руйнування стеку

Оскільки використовувалася небезпечна функція було викликано виключну ситуацію порушення стеку (рис.3).

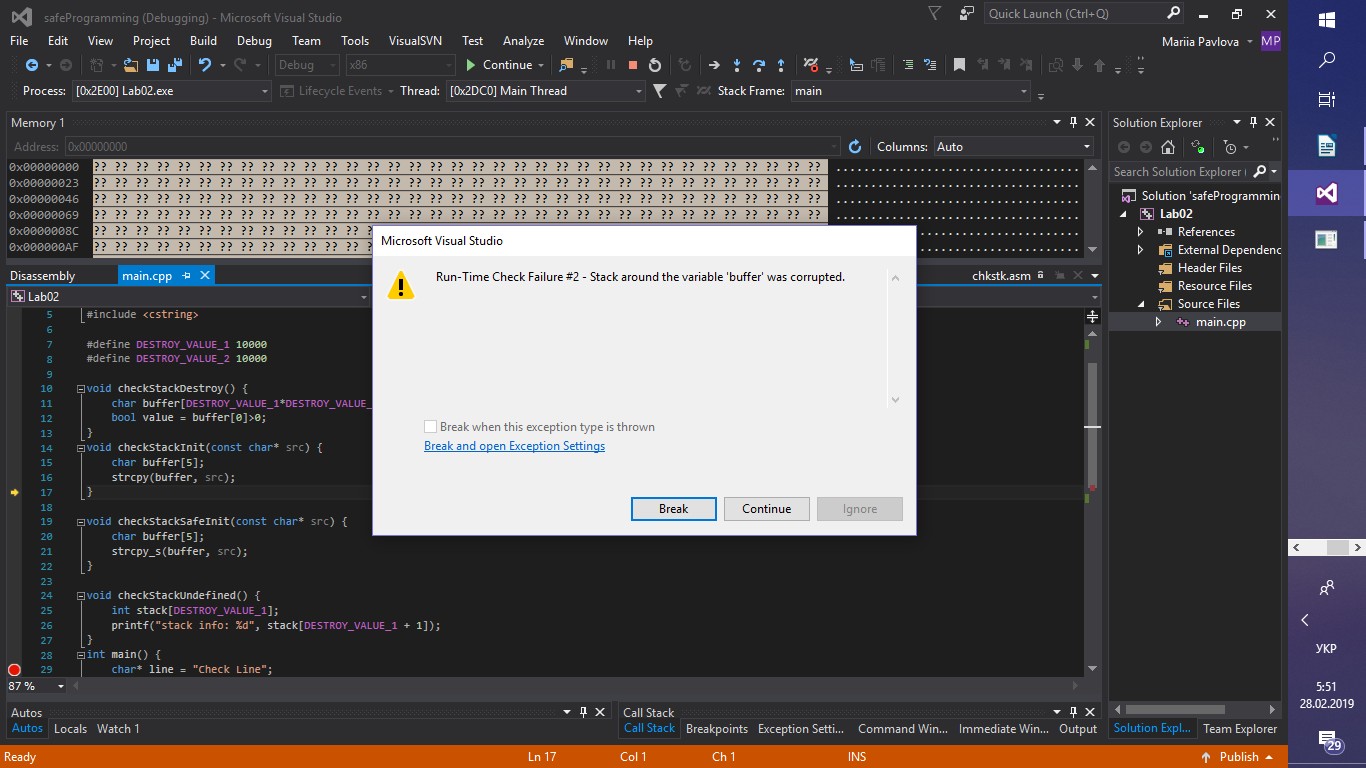


Рисунок 3 — Порушення стеку

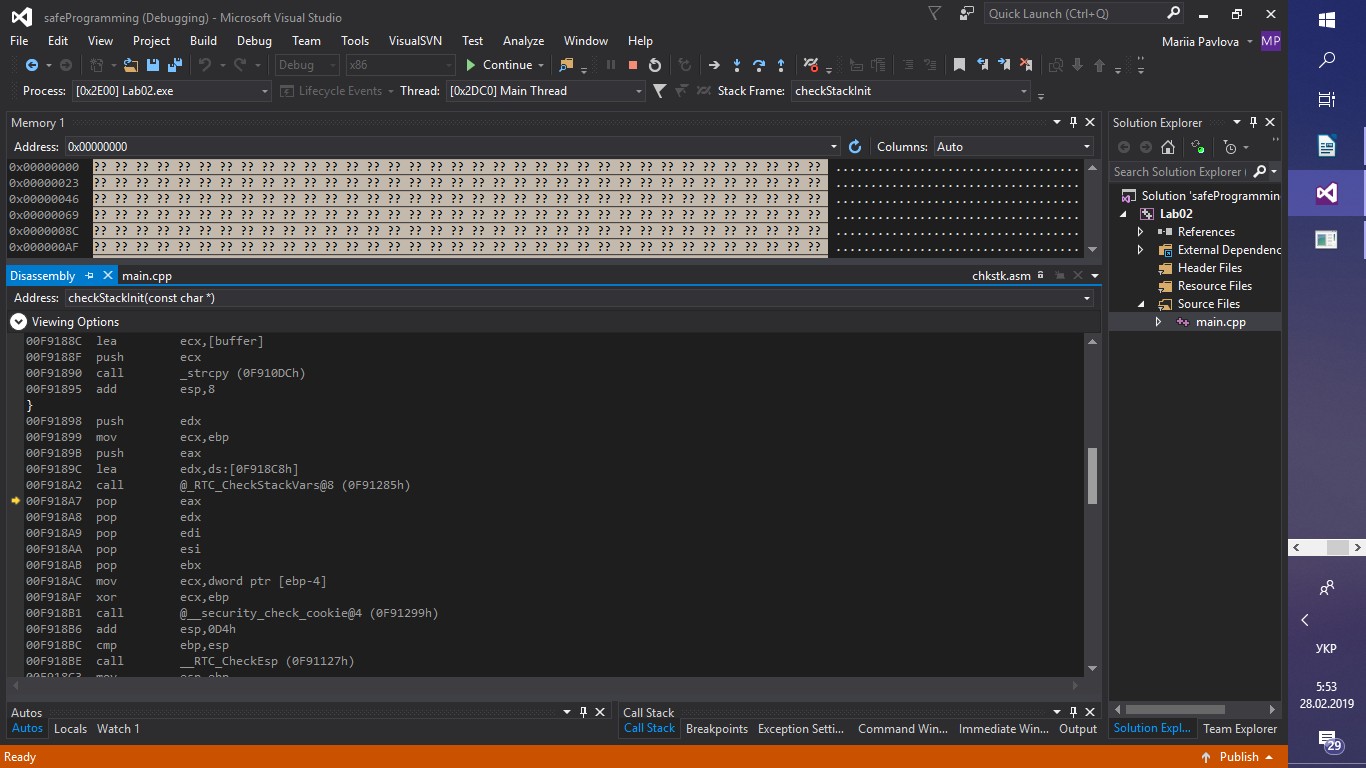


Рисунок 5 — Виключна ситуація в асемблерному коді

При використанні безпечної функції у випадку переповнення стеку маємо наступне попередження (рис.6).

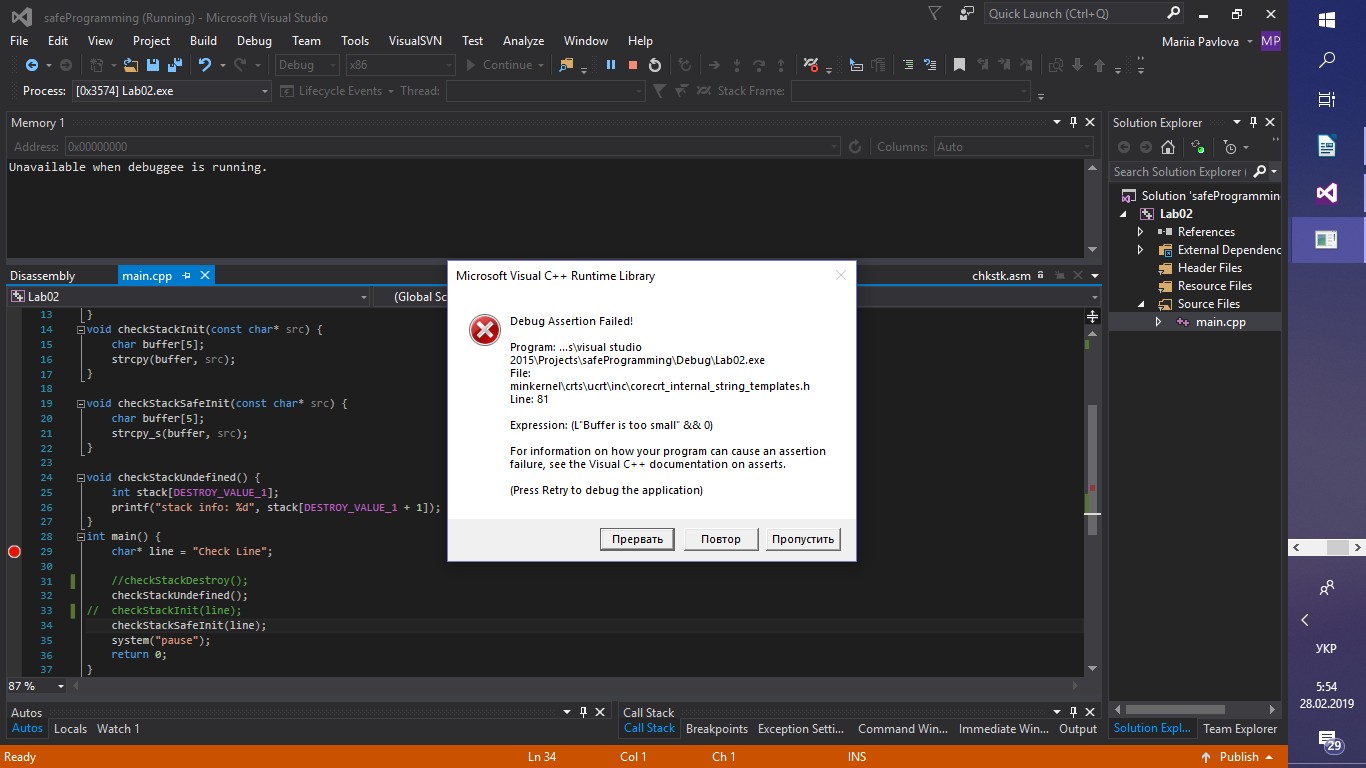


Рисунок 5 — Попередження

Далі спробуємо працювати із параметром /GS. Такий запуск програми одразу ж попереджає про небезпечну функцію (рис.6).

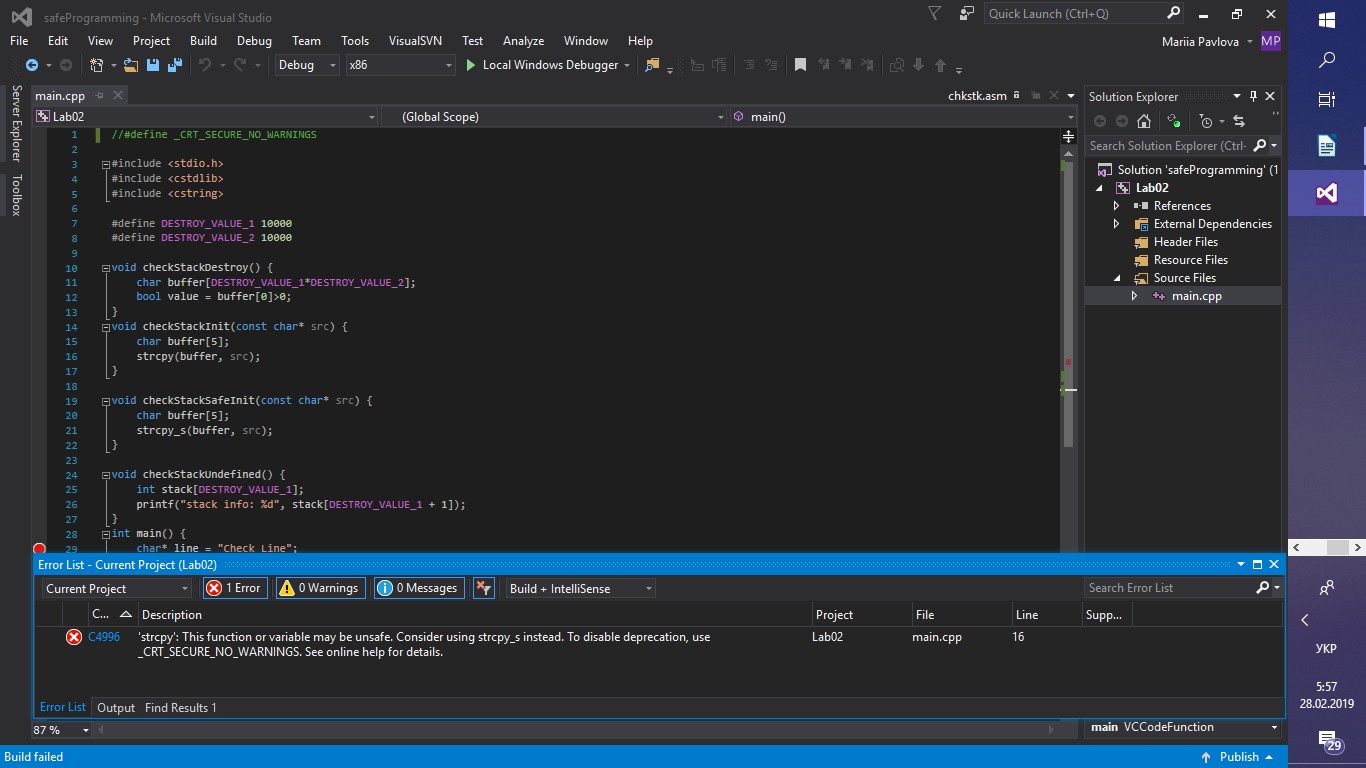


Рисунок 6 — Попередження

Далі робота програми ідентична попереднім запускам.

У наступній частині лабораторної роботи перевіремо роботу зі стеком у Eclipse. У цій сфері розробки параметр -fno-stack-protector є за замовчуванням.

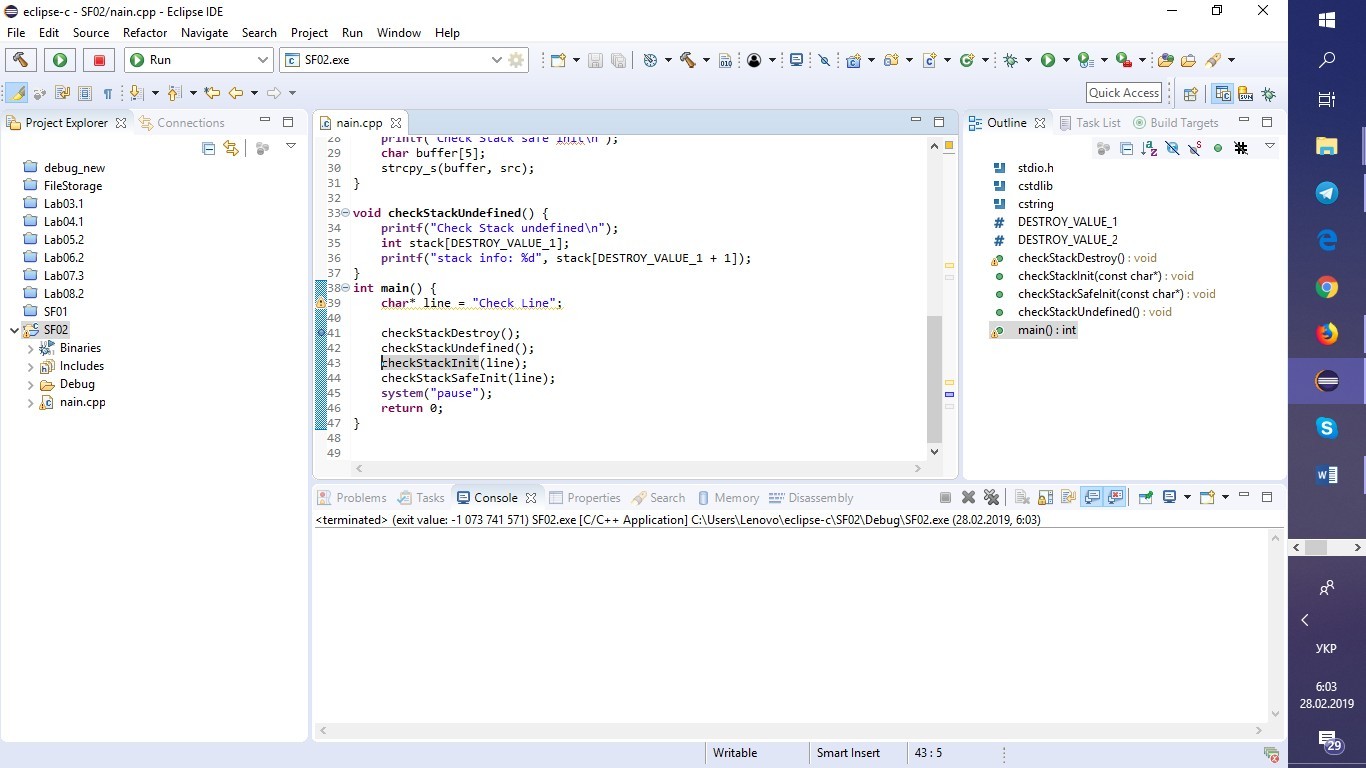


Рисунок 7 — Термінований вихід з програми

При запуску функції, що звертається до неіснуючих елементів стеку маємо виведення сміття з пам’яті, до якої звернулись, а також термінований вихід з програми. (рис.8).

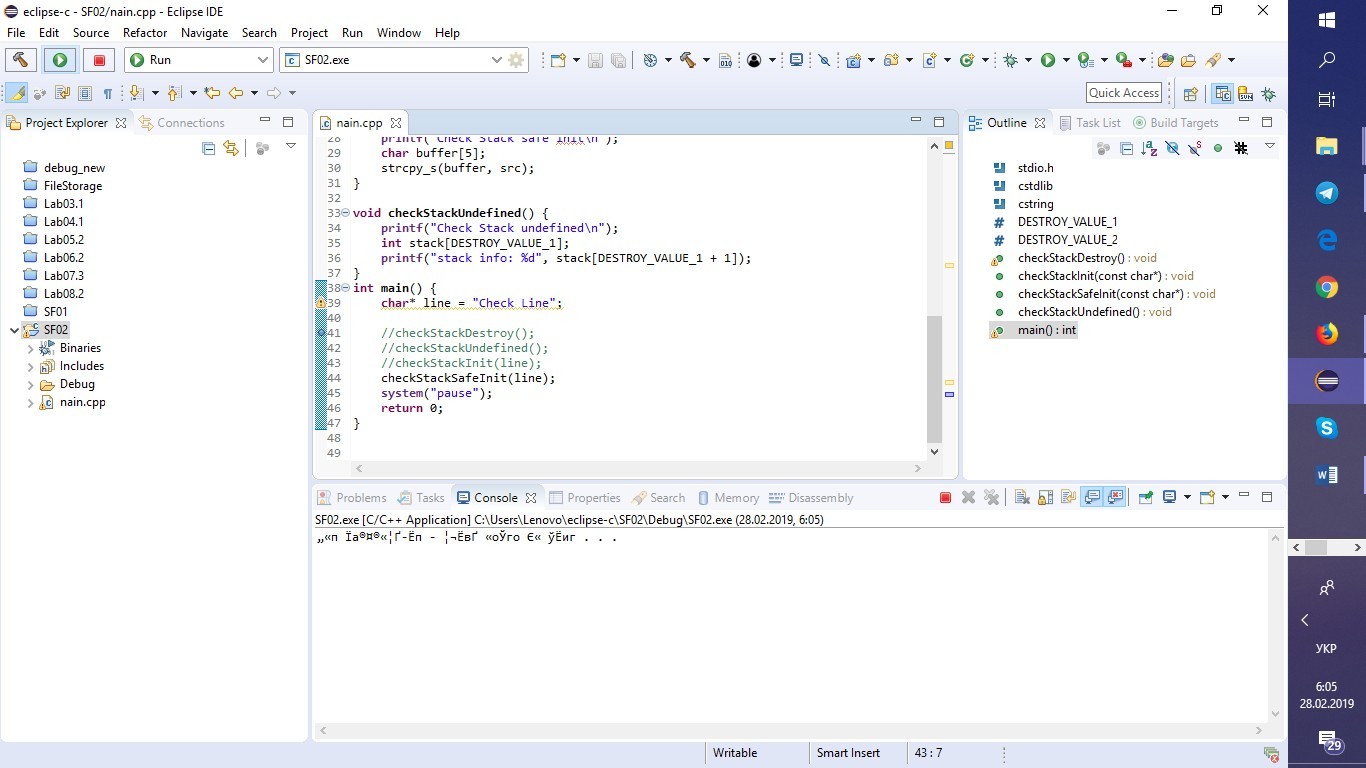


Рисунок 8 — Результат роботи програми

ВИСНОВКИ

В результаті виконання лабораторної роботи було продемонстровано руйнування стеку, його причини та наслідки.