

## Домашнє завдання №28\_2

У домашньому завданні 28\_3 потрібно буде виконати домашнє завдання №27\_1 повторно як альтернативну низькорівневу реалізацію домашнього завдання №28\_1 мовою С. В якості домашнього завдання 28\_2 пропонується також виконати домашнє завдання №27\_1 повторно як альтернативну низькорівневу реалізацію домашнього завдання №28\_1 мовою С, але за допомогою бібліотеки libuv. Рушій платформи NodeJS побудований на основі цієї бібліотеки, яка була створена для заміни libeio(імплементує Tread Pool) та libev(імплементує Event Loop). *(Рушій JavaScript для NodeJS це V8, але рушієм подійно-орієнтованої парадигми у NodeJS є саме libuv).*

*\* коментар: це завдання аналогічне №27\_1, №27\_2 та №27\_3 і є повністю тотожне до завдань №28\_1 та №28\_3; таким чином можна порівняти різні засоби програмування; далі наводиться приклад повністю виконаного завдання; для компіляції і запуску можна використати <https://repl.it/languages/c> , нижче також показаний спосіб виконання наведеного прикладу коду за допомогою цього засобу, а також за допомогою Visual Studio .*

### Вибір варіанту

Варіант завдання відповідає варіанту домашнього завдання №27_1
--

### Спосіб виконання наведеного прикладу коду за допомогою

<https://repl.it/languages/c>

На рисунках 1, 2, 3, 4, 5 та 6 послідовно показаний спосіб виконання наведеного прикладу коду за допомогою <https://repl.it/languages/c> . На рисунках 7 та 8 показані спроби введення ключа ліцензії.

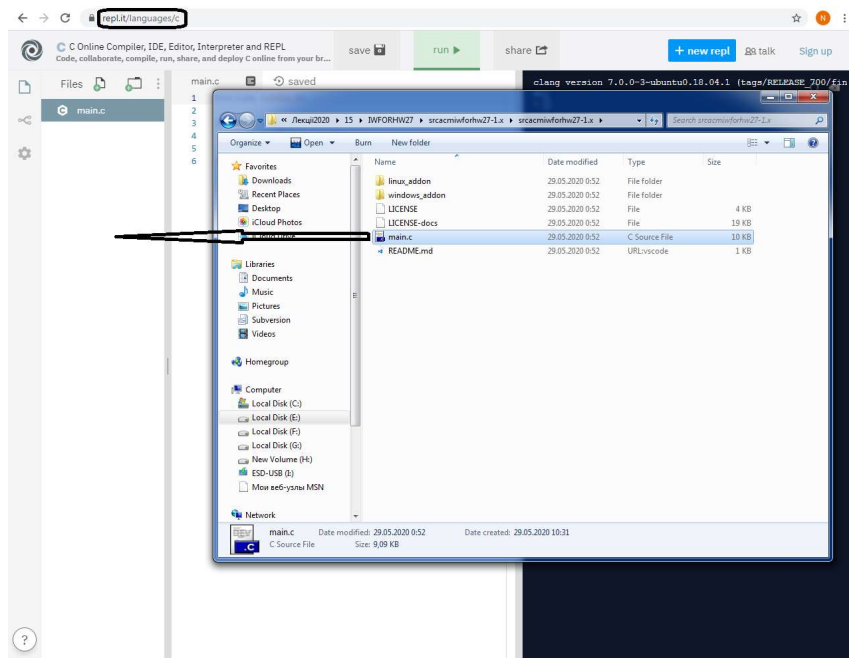


Рис. 1. Перетягування main.c

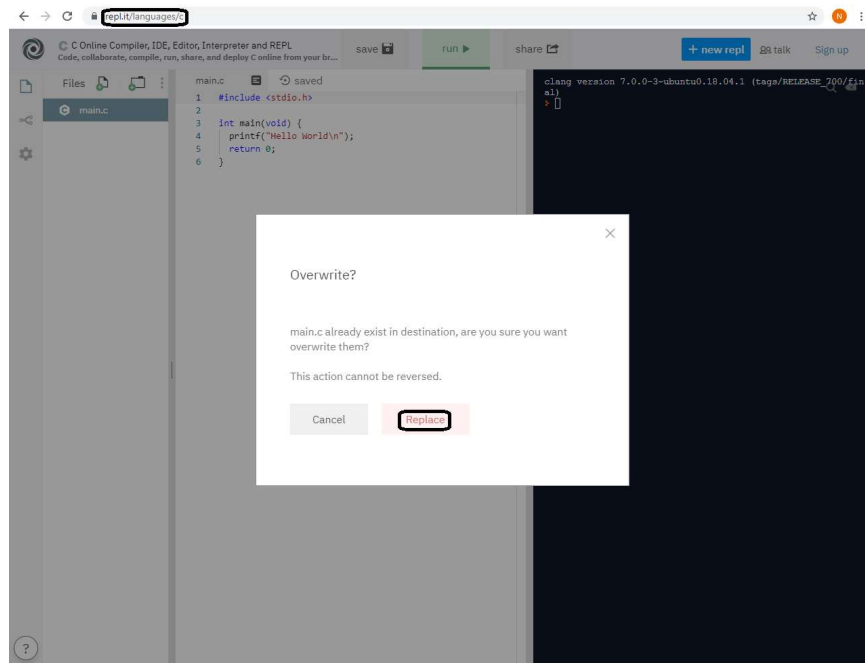
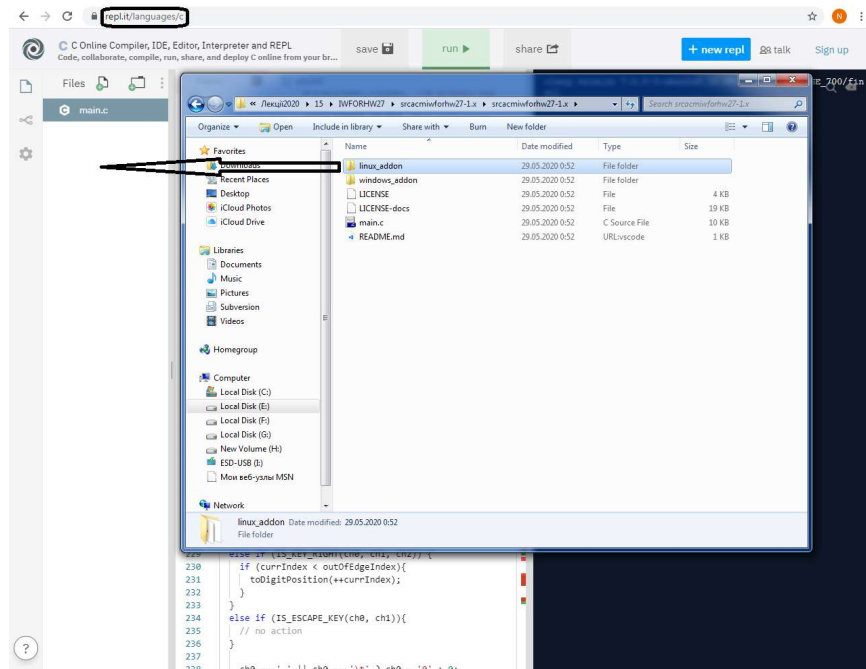


Рис. 2. Підтвердження перезапису файлу main.c

Якщо перезаписати файл не вдається, то можна замінити вміст файлу. Далі потрібно скопіювати файли бібліотеки (тека linux\_addon).



*Рис. 3. Перетягування файлів бібліотеки для Linux*

Якщо виконувати компіляцію натисканням кнопки у інтерфейсі засобу <https://repl.it/languages/c> , то при компіляції не буде включено бібліотеку для динамічного завантаження бібліотек(–ldl), тому компіляцію потрібно виконати вручну. Приклад команди для компіляції міститься в файлі main.cpp. Цю команду потрібно скопіювати (рис. 4) та вставити у віртуальну консоль засобу <https://repl.it/languages/c> (рис. 5), після чого натиснути Enter.

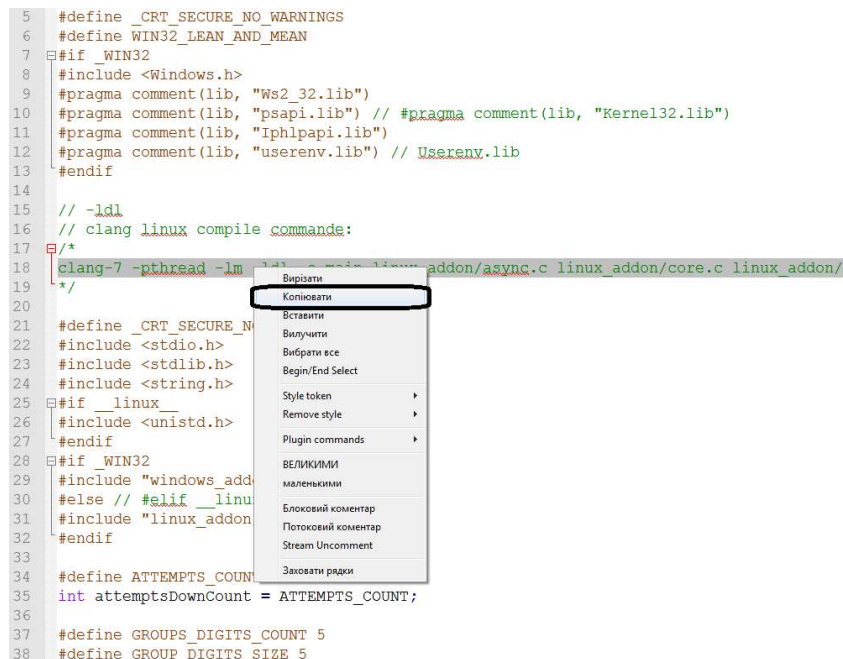


Рис. 4. Копіювання команди компіляції з файлу `main.c`

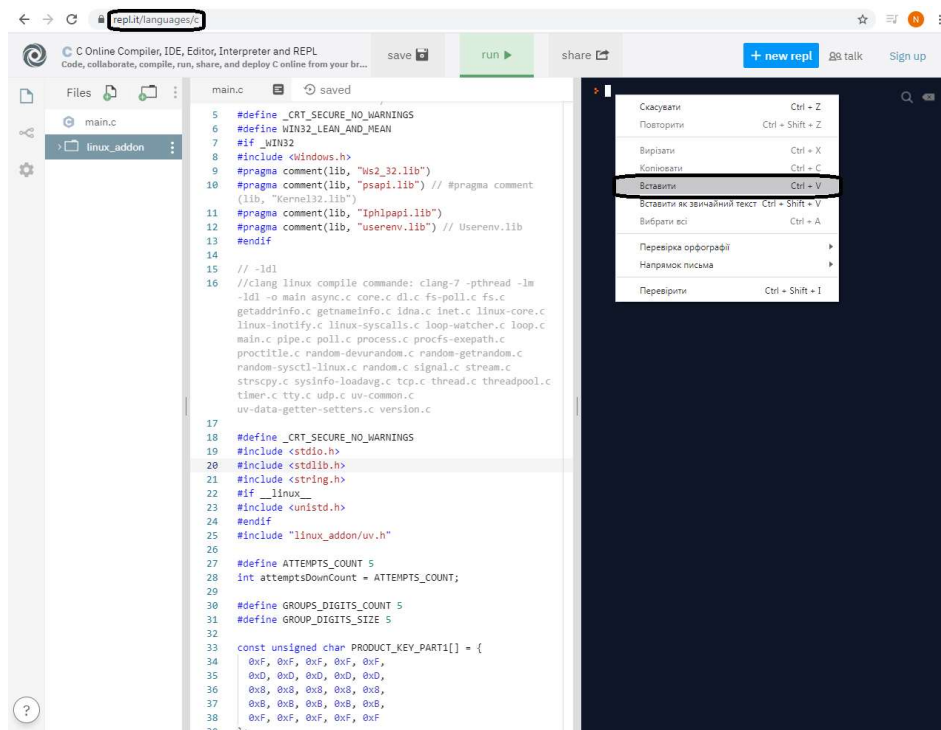
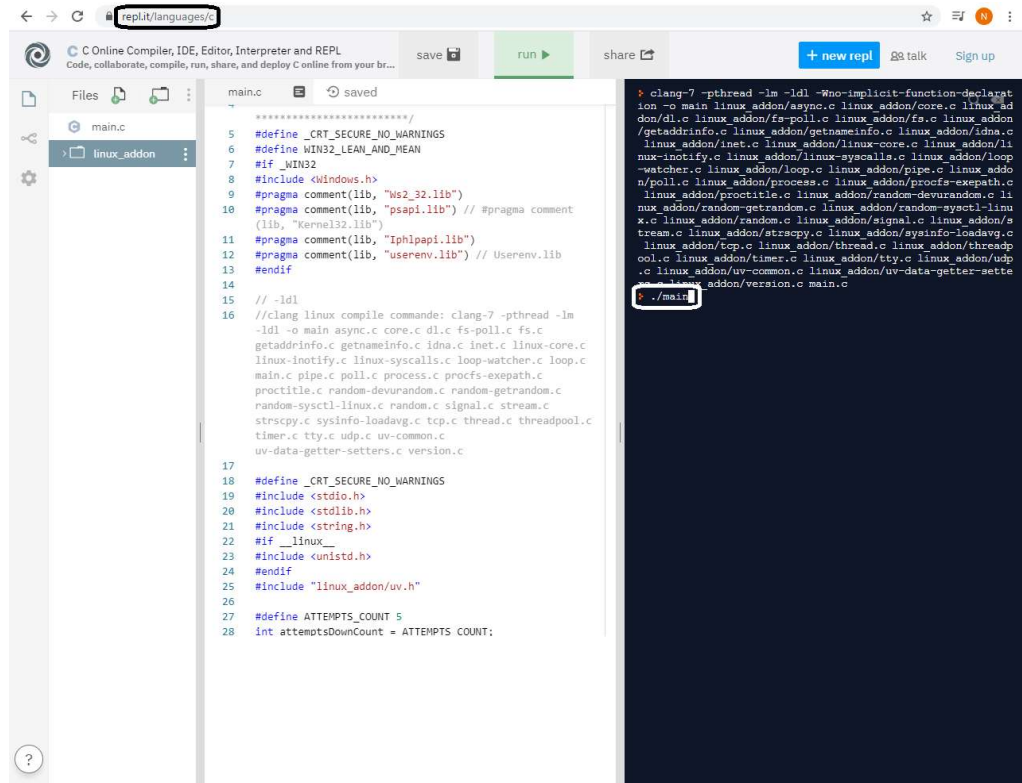


Рис. 5. Копіювання команди компіляції у віртуальну консоль

Після копіювання файлів у засіб <https://repl.it/languages/c> можливо не всі вони будуть одразу опрацьовані і при компіляції виникнуть помилки, тому потрібно трохи зачекати і повторно(можливо кілька разів) виконати компіляцію. Далі можна запустити скомпільовану програму за допомогою віртуальної консолі(ввести назву програми(рис. 6) і натиснути Enter):

```
./main
```



```

clang-7 -pthread -lm -ldl -Wno-implicit-function-declara
ion -o main linux_addon/async.c linux_addon/core.c linux_ad
don/dl.c linux_addon/fs-poll.c linux_addon/fs.c linux_addon
/getaddrinfo.c linux_addon/getnameinfo.c linux_addon/idna.c
linux_addon/inet.c linux_addon/linux-core.c linux_addon/li
nux-inotify.c linux_addon/linux-syscalls.c linux_addon/loop
-watcher.c linux_addon/loop.c linux_addon/pipe.c linux_addo
n/poll.c linux_addon/process.c linux_addon/proctitle.c
linux_addon/proctitle.c linux_addon/random-devurandom.c li
nux_addon/random-getrandom.c linux_addon/random-sysctl-linu
x.c linux_addon/random.c linux_addon/signal.c linux_addon/s
tream.c linux_addon/strncpy.c linux_addon/sysinfo-loadavg.c
linux_addon/tcp.c linux_addon/thread.c linux_addon/threadp
ool.c linux_addon/timer.c linux_addon/tty.c linux_addon/udp
.c linux_addon/uv-common.c linux_addon/uv-data-getter-sette
r.c linux_addon/version.c main.c
./main
  
```

Рис. 6. Запуск програми

```

Please, enter the product key:
11111-11111-11111-11111-11111
The product key is not correct

You have 4 attempts to try
Please, enter the product key:
11111-11111-11111-11111-11111
  
```

Рис. 7. Повідомлення про помилкове введення ключа ліцензії

```

Please, enter the product key:
11111-11111-11111-11111-11111
The product key is not correct

You have 4 attempts to try
Please, enter the product key:
11111-22222-33333-44444-55555
The product key is correct

1111122222333334444455555 (COMPLETE)
For exit press Cntl + C
  
```

Рис. 8. Повідомлення про коректне введення ключа ліцензії

## Спосіб виконання наведеного прикладу коду за допомогою Visual Studio

На рисунках 1, 2, 3, 5, 6, 7, 8 та 9 послідовно показаний спосіб виконання наведеного прикладу коду за допомогою Visual Studio. На рисунках 10 та 11 показані спроби введення ключа ліцензії.

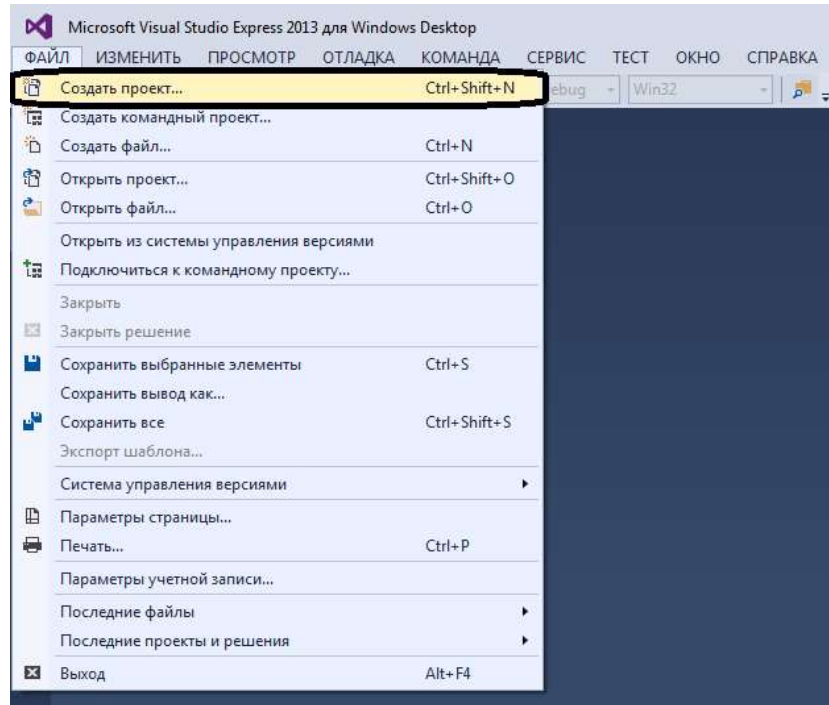


Рис. 1. Створення нового проекту у Visual Studio

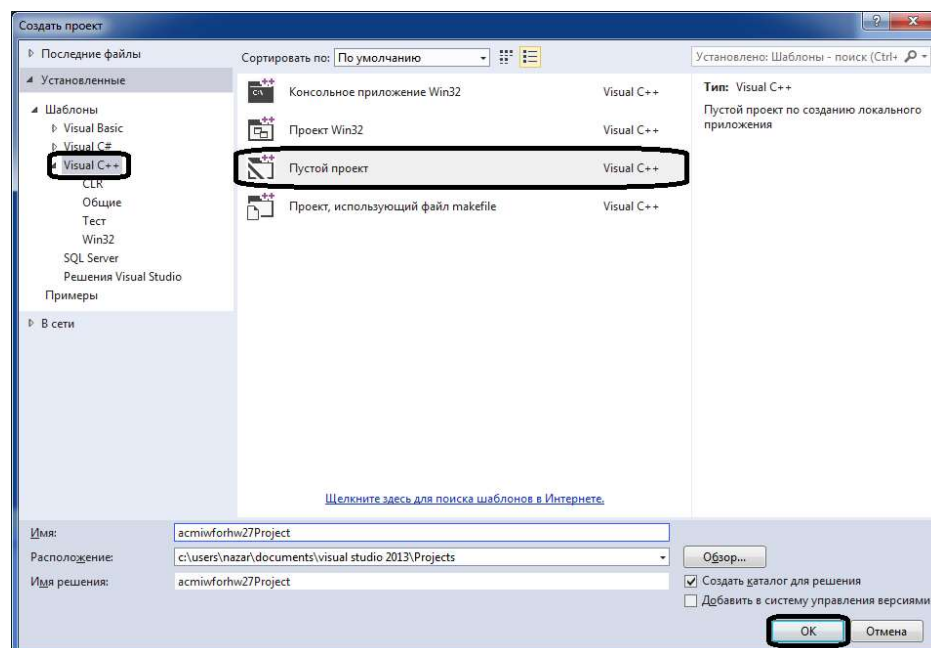


Рис. 2. Вибір пусого проекту при створенні у Visual Studio



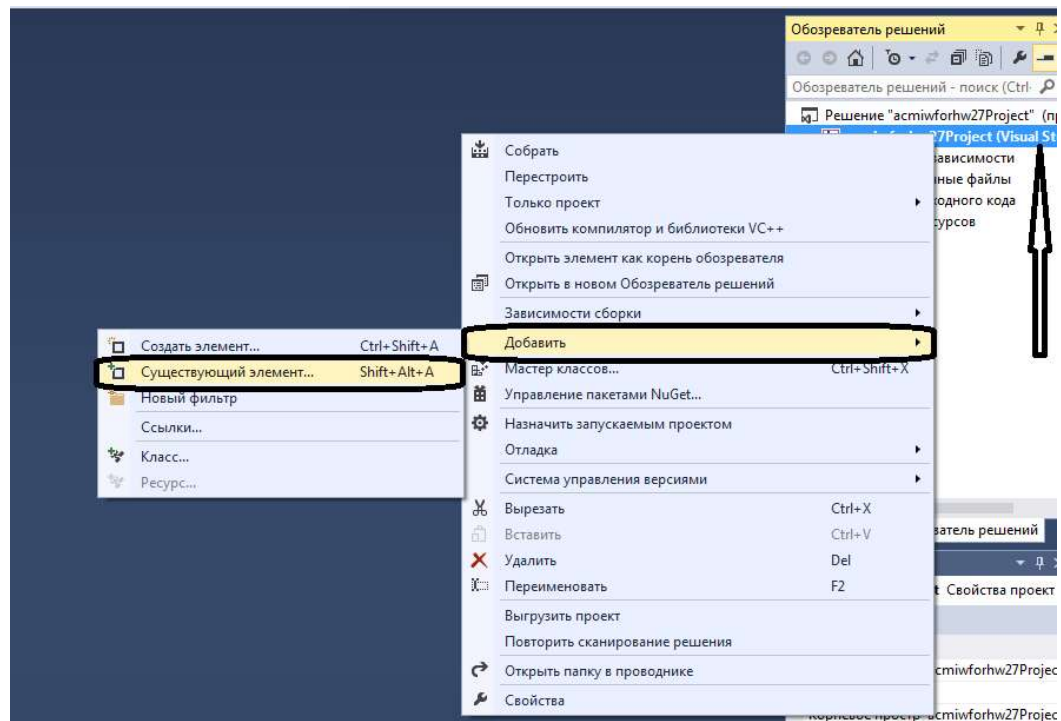


Рис. 3. Відкриття вікна для додавання існуючих елементів

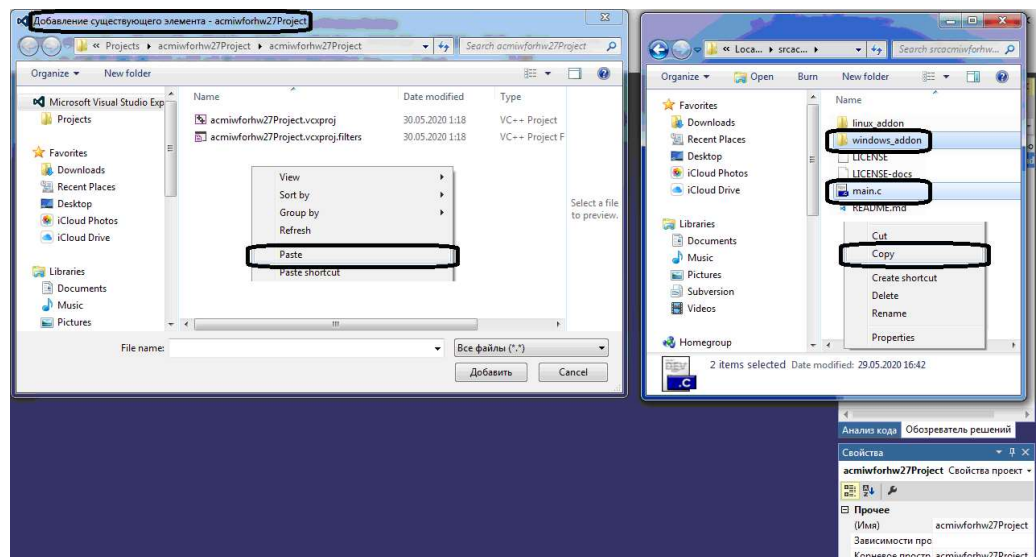


Рис. 4. Копіювання файлів у каталог проекту

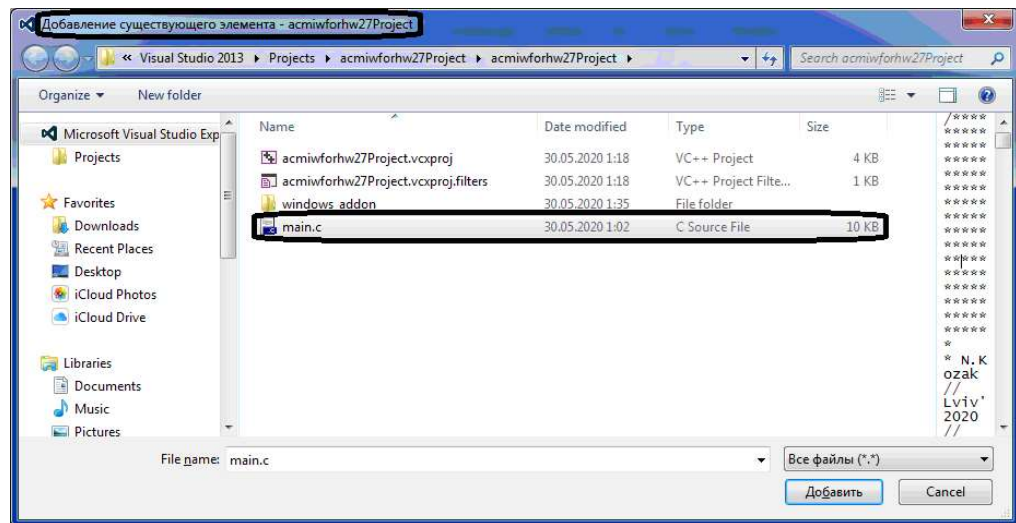


Рис. 5. Додавання файлу main.c у проект

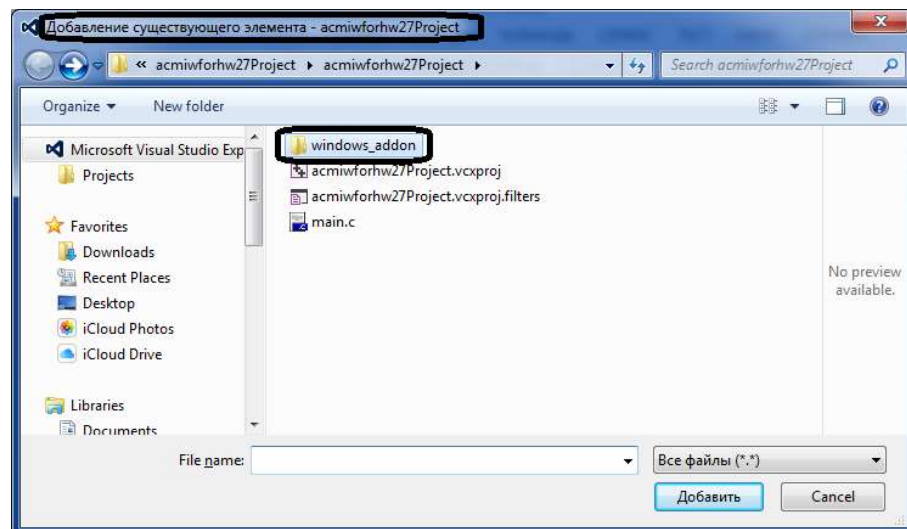


Рис. 6. Відкриття теки з файлами для Windows бібліотеки libuv



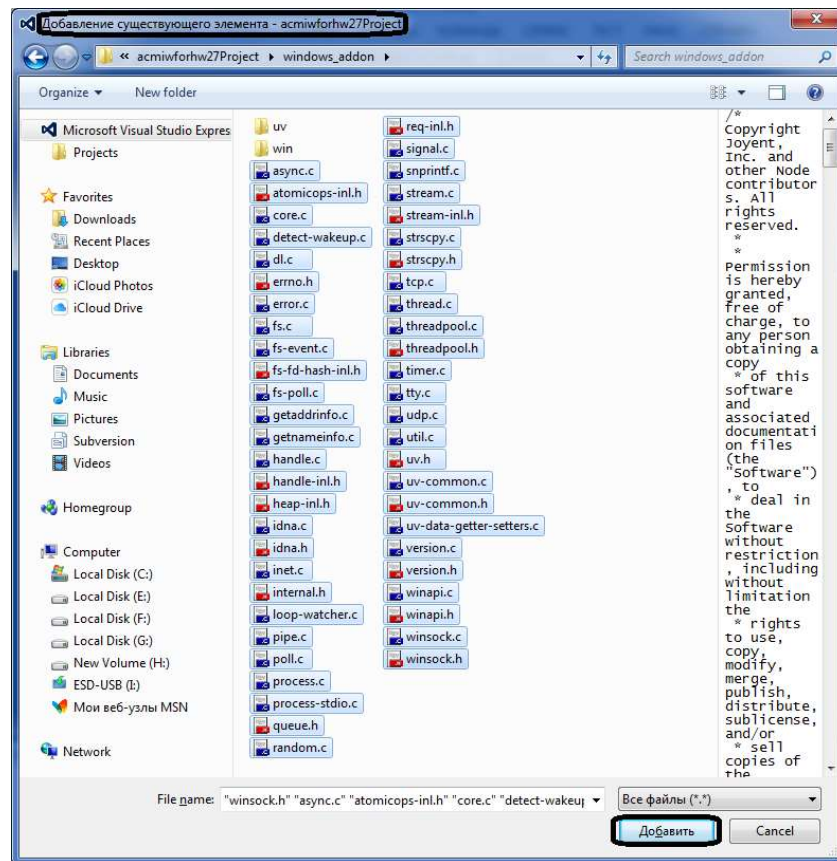


Рис. 7. Додавання файлів бібліотеки libuv

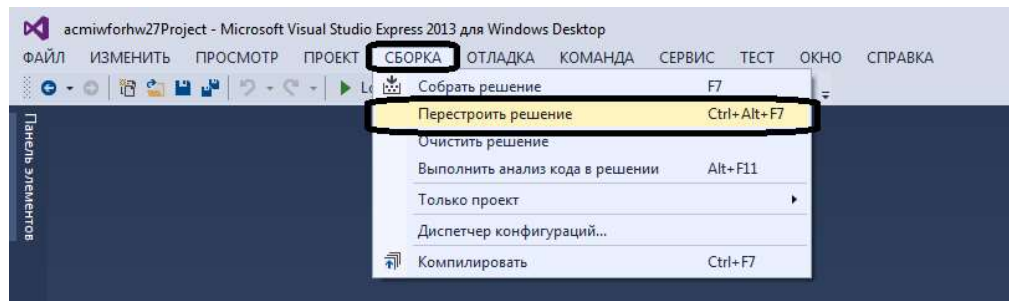


Рис. 8. Компіляція програми

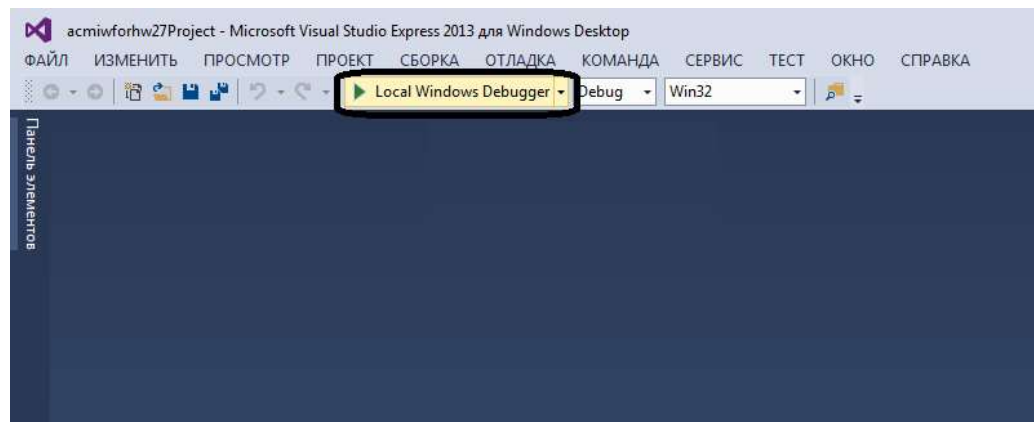


Рис. 9. Запуск програми

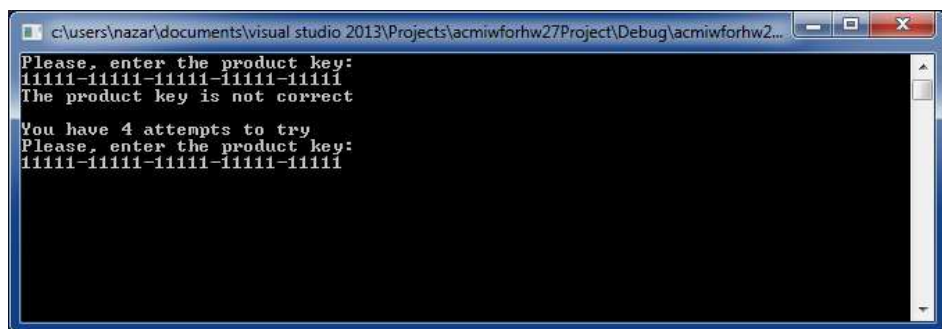


Рис. 10. Повідомлення про помилкове введення ключа ліцензії

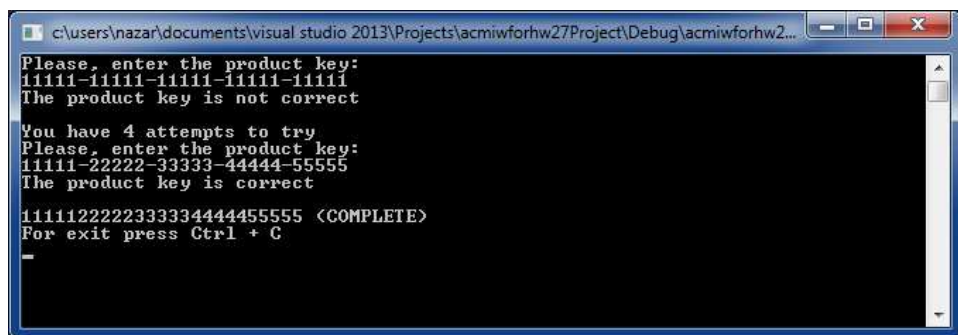


Рис. 11. Повідомлення про коректне введення ключа ліцензії

## Приклад коду

Наведений зразок коду реалізовує завдання для 5-ти максимально допустимих спроб введення ключа ліцензії.

Максимальна кількість спроб для введення ключа ліцензії	5
Макровизначення	<code>#define ATTEMPTS_COUNT 5</code>

Для коректного виконання коду за допомогою <https://repl.it/languages/c> віртуальну консоль з правого боку краще трохи розширити перед початком виконання коду, а у процесі виконання розмір консолі не змінювати.

*Лістинг*

```
#define _CRT_SECURE_NO_WARNINGS
#define WIN32_LEAN_AND_MEAN
#if _WIN32
#include <Windows.h>
#pragma comment(lib, "ws2_32.lib")
#pragma comment(lib, "psapi.lib") // #pragma comment(lib, "Kernel32.lib")
#pragma comment(lib, "Iphlpapi.lib")
#pragma comment(lib, "userenv.lib") // Userenv.lib
#endif

// -ldl
// clang linux compile commande:
/*
clang-7 -pthread -lm -ldl -Wno-implicit-function-declaration -o main linux_addon/async.c
linux_addon/core.c linux_addon/dl.c linux_addon/fs-poll.c linux_addon/fs.c
linux_addon/getaddrinfo.c linux_addon/getnameinfo.c linux_addon/idna.c linux_addon/inet.c
linux_addon/linux-core.c linux_addon/linux-inotify.c linux_addon/linux-syscalls.c
linux_addon/loop-watcher.c linux_addon/loop.c linux_addon/pipe.c linux_addon/poll.c
linux_addon/process.c linux_addon/procfs-exepath.c linux_addon/proctitle.c
linux_addon/random-devurandom.c linux_addon/random-getrandom.c linux_addon/random-sysctl-
linux.c linux_addon/random.c linux_addon/signal.c linux_addon/stream.c linux_addon/strscpy.c
linux_addon/sysinfo-loadavg.c linux_addon/tcp.c linux_addon/thread.c
linux_addon/threadpool.c linux_addon/timer.c linux_addon/tty.c linux_addon/udp.c
linux_addon/uv-common.c linux_addon/uv-data-getter-setters.c linux_addon/version.c main.c
*/

#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#if __linux__
#include <unistd.h>
#endif
#if _WIN32
#include "windows_addon/uv.h"
#else // #elif __linux__
#include "linux_addon/uv.h"
#endif

#define ATTEMPTS_COUNT 5
int attemptsDownCount = ATTEMPTS_COUNT;

#define GROUPS_DIGITS_COUNT 5
#define GROUP_DIGITS_SIZE 5

const unsigned char PRODUCT_KEY_PART1[] = {
    0xF, 0xF, 0xF, 0xF, 0xF,
```

```

        0xD, 0xD, 0xD, 0xD, 0xD,
        0x8, 0x8, 0x8, 0x8, 0x8,
        0xB, 0xB, 0xB, 0xB, 0xB,
        0xF, 0xF, 0xF, 0xF, 0xF
    };

    const unsigned char PRODUCT_KEY_PART2[] = {
        0xE, 0xE, 0xE, 0xE, 0xE,
        0xF, 0xF, 0xF, 0xF, 0xF,
        0xB, 0xB, 0xB, 0xB, 0xB,
        0xF, 0xF, 0xF, 0xF, 0xF,
        0xA, 0xA, 0xA, 0xA, 0xA
    };

#define DIGITS_COUNT (GROUPS_DIGITS_COUNT * GROUP_DIGITS_SIZE)

#define TYPED_FULL_RAW_MODE

#define IS_KEY_UP(CH0, CH1, CH2) (CH0 == 0x1b && CH1 == '[' && CH2 == 'A')
#define IS_KEY_DOWN(CH0, CH1, CH2) (CH0 == 0x1b && CH1 == '[' && CH2 == 'B')
#define IS_KEY_LEFT(CH0, CH1, CH2) (CH0 == 0x1b && CH1 == '[' && CH2 == 'D')
#define IS_KEY_RIGHT(CH0, CH1, CH2) (CH0 == 0x1b && CH1 == '[' && CH2 == 'C')
#define IS_ESCAPE_KEY(CH0, CH1) (CH0 == 0x1b && CH1 == 0x1b)
#define IS_KEY_DELETE(CH0, CH1, CH2, CH3) (CH0 == 0x1b && CH1 == '[' && CH2 == '3') // &&
CH3 == '^')
#ifdef _WIN32
#define IS_KEY_BACKSPACE(CH0) (CH0 == 8)
#else // #elif __linux__
#define IS_KEY_BACKSPACE(CH0) (CH0 == 127)
#endif
#ifdef TYPED_FULL_RAW_MODE
#define IS_KEY_ENTER(CH0) (CH0 == 13)
#else
#define IS_KEY_ENTER(CH0) (CH0 == 10)
#endif
#define IS_KEY_CTRL_C(CH0) (CH0 == 3)

int outOfEdgeIndex = 0;
int currIndex = 0;
unsigned char data[DIGITS_COUNT] = { 0 };

char checkProductKey(unsigned char * productKey){
    unsigned int index;
    for (index = 0; index < DIGITS_COUNT; ++index){
        if (productKey[index] ^ PRODUCT_KEY_PART1[index] ^ PRODUCT_KEY_PART2[index]){
            return 0;
        }
    }

    return ~0;
}

void toDigitPosition(unsigned int currIndex){
    int positionAddon;
#ifdef _WIN32
#else // #elif __linux__
    char temp[16];
#endif
#ifdef _WIN32
    CONSOLE_SCREEN_BUFFER_INFO cbsi;
    COORD pos;
    HANDLE hConsoleOutput = GetStdHandle(STD_OUTPUT_HANDLE);
    GetConsoleScreenBufferInfo(hConsoleOutput, &cbsi);
    pos = cbsi.dwCursorPosition;
#endif
}

```

```

        positionAddon = currIndex / GROUP_DIGITS_SIZE;
        positionAddon && positionAddon >= GROUPS_DIGITS_COUNT ? --positionAddon : 0;
    #if _WIN32
        currIndex += positionAddon;
        pos.X = currIndex;
        SetConsoleCursorPosition(hConsoleOutput, pos);
    #else // #elif __linux__
        write(STDOUT_FILENO, "\033[64D", 5);
        if(currIndex += positionAddon){
            sprintf(temp, "\033[%dC", currIndex);
            write(STDOUT_FILENO, temp, strlen(temp));
        }
    #endif
}

void printProductKey(unsigned char * productKey, unsigned int outOfEdgeIndex){
    unsigned int index;
    unsigned char value;
    for (index = 0; index < DIGITS_COUNT && index < outOfEdgeIndex; ++index){
        value = productKey[index];
        value > 9 ? (value += 'A' - 10) : (value += '0');
    #if _WIN32
        printf("%c", value);
    #else // #elif __linux__
        write(STDOUT_FILENO, &value, 1);
    #endif
    }
}

void printFormattedProductKey(unsigned char * productKey, unsigned int outOfEdgeIndex){
    unsigned int index;
    unsigned char value;
    for (index = 0; index < DIGITS_COUNT && index < outOfEdgeIndex; ++index){
        value = productKey[index];
        value > 9 ? (value += 'A' - 10) : (value += '0');
    #if _WIN32
        printf("%c", value);
    #else // #elif __linux__
        write(STDOUT_FILENO, &value, 1);
    #endif
        if (!((index + 1) % GROUP_DIGITS_SIZE) && (index + 1) < DIGITS_COUNT){
    #if _WIN32
        printf("-");
    #else // #elif __linux__
        write(STDOUT_FILENO, "-", 1);
    #endif
        }
    }
}

void inputHandler(int ch0, int ch1, int ch2, int ch3){
    char chstr[2] = { 0 };
    char * hexDigitScanfPattern = (char*)"[%0-9abcdefABCDEF]"; // /[%0-9A-Fa-f]/g

    if (!attemptsDownCount){
        return;
    }
    if (IS_KEY_ENTER(ch0)) {
        if (checkProductKey(data)) {
    #if _WIN32
        printf("\nThe product key is correct\n\n");
    #else // #elif __linux__
        write(STDOUT_FILENO, "\nThe product key is correct\n\n", 29);
    #endif
        printProductKey(data, outOfEdgeIndex);

```

```

#if _WIN32
    printf(" (COMPLETE)", 11);
    printf("\nFor exit press Ctrl + C\n");
#else // #elif __linux__
    write(STDOUT_FILENO, " (COMPLETE)", 11);
    write(STDOUT_FILENO, "\nFor exit press Ctrl + C\n", 25);
#endif

    attemptsDownCount = 0;
}
else{
#if _WIN32
    printf("\nThe product key is not correct\n");
    printf("\nYou have %d attempts to try\n", --attemptsDownCount);
#else // #elif __linux__
    write(STDOUT_FILENO, "\nThe product key is not correct\n", 32);
    printf("\nYou have %d attempts to try\n", --attemptsDownCount);
#endif

    if (attemptsDownCount){
#if _WIN32
        printf("Please, enter the product key:\n");
#else // #elif __linux__
        write(STDOUT_FILENO, "Please, enter the product key:\n", 31);
#endif

        printFormattedProductKey(data, outOfEdgeIndex);
        toDigitPosition(currIndex);
    }
    else{
#if _WIN32
        printf("The product key is not entered\n");
        printf("For exit press Ctrl + C\n");
#else // #elif __linux__
        write(STDOUT_FILENO, "The product key is not entered\n", 31);
        write(STDOUT_FILENO, "For exit press Ctrl + C\n", 24);
#endif

    }
}
}
else if (IS_KEY_BACKSPACE(ch0)) {
    if (currIndex){
        --currIndex;
        toDigitPosition(currIndex);
        data[currIndex] = 0;
#if _WIN32
        printf("0");
#else // #elif __linux__
        write(STDOUT_FILENO, "0", 1);
#endif

        toDigitPosition(currIndex);
    }
}
else if (IS_KEY_DELETE(ch0, ch1, ch2, ch3)) {
    toDigitPosition(currIndex);
    data[currIndex] = 0;
#if _WIN32
    printf("0");
#else // #elif __linux__
    write(STDOUT_FILENO, "0", 1);
#endif

    toDigitPosition(currIndex);
}
else if (IS_KEY_LEFT(ch0, ch1, ch2)) {
    if (currIndex){
        toDigitPosition(--currIndex); // got to 1.5
    }
}
}

```



```

else if (IS_KEY_RIGHT(ch0, ch1, ch2)) {
    if (currIndex < outOfEdgeIndex){
        toDigitPosition(++currIndex);
    }
}
else if (IS_ESCAPE_KEY(ch0, ch1)){
    // no action
}

ch0 == ' ' || ch0 == '\t' ? ch0 = '0' : 0;

//char chstr_[2] = { 0 };
//char * hexDigitScanfPattern = (char*)"[%0-9abcdefABCDEF]"; // /[%0-9A-Fa-f]/g

if (currIndex < DIGITS_COUNT && ch0 && sscanf((char*)&ch0, hexDigitScanfPattern,
chstr_) > 0) {
    data[currIndex] = (unsigned char)strtol(chstr_, NULL, 16);
#ifdef _WIN32
    printf("%X", data[currIndex]);
#else // #elif __linux__
    sprintf(chstr_, "%X", data[currIndex] );
    write(STDOUT_FILENO, chstr_, 1);
#endif
    if (outOfEdgeIndex <= currIndex){
        outOfEdgeIndex = currIndex + 1;
    }
    if (currIndex + 1 < DIGITS_COUNT) {
        ++currIndex;
        if (currIndex != DIGITS_COUNT && !(currIndex % 5)) {
#ifdef _WIN32
            printf("-");
#else // #elif __linux__
            write(STDOUT_FILENO, "-", 1);
#endif
        }
    }
    if (currIndex + 1 == DIGITS_COUNT){
        toDigitPosition(currIndex);
    }
}

uv_loop_t mainLoop;

void terminateHandler(int ch0, int ch1, int ch2, int ch3){
    if (IS_KEY_CTRL_C(ch0)) {
        uv_stop(&mainLoop);
    }
}

static void alloc_buffer(uv_handle_t *handle, size_t suggested_size, uv_buf_t *buf)
{
    static char buffer[1 << 16];
    *buf = uv_buf_init(buffer, 1 << 16);
}

static void read_stdin(uv_stream_t *stream, ssize_t nread, const uv_buf_t* buf){
    int ch0 = nread > 0 ? buf->base[0] : 0;
    int ch1 = nread > 1 ? buf->base[1] : 0;
    int ch2 = nread > 2 ? buf->base[2] : 0;
    int ch3 = nread > 3 ? buf->base[3] : 0;

    inputHandler(ch0, ch1, ch2, ch3);
    terminateHandler(ch0, ch1, ch2, ch3);
}

```

```
int main(){
    uv_tty_t input;
    uv_loop_init(&mainLoop);

    uv_tty_init(&mainLoop, &input, 0/* = stdin*/, 1);
    uv_tty_set_mode(&input, UV_TTY_MODE_RAW); //

    uv_read_start((uv_stream_t *)&input, alloc_buffer, read_stdin);

    #if _WIN32
        printf("Please, enter the product key:\n");
    #else // #elif __linux__
        write(STDOUT_FILENO, "Please, enter the product key:\n", 31);
    #endif

    /* Run main loop */
    uv_run(&mainLoop, UV_RUN_DEFAULT);

    uv_tty_reset_mode();

    return 0;
}
```