

Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Ульяновский государственный технический университет»
Кафедра «Вычислительная техника»

Организация вычислительных машин и систем

Лабораторная работа №2
«Определение конфигурации компьютера программными средствами»

Выполнил
Студент группы ИВТбд-21
Ведин В. А.
Проверил(а):
ст. преподаватель кафедры «ВТ»
Лылова А.В.

Ульяновск
2024

ОГЛАВЛЕНИЕ

ЗАДАНИЕ	3
ВЫБОР ПРОГРАММНОГО ПРОДУКТА	6
ВЫБОР ЯЗЫКА ПРОГРАММИРОВАНИЯ	7
АЛГОРИТМ ПРОГРАММЫ	8
Код программы на «С»:	8
Дизассемблированный код:	10
ТЕСТЫ ПРОГРАММЫ И СРАВНИТЕЛЬНЫЙ АНАЛИЗ	14
ИТОГИ	16
СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ	17

ЗАДАНИЕ

Требуется написать программу на языке C, которая определяет наличие тип установленного манипулятора «мышь» и считать данные с этого устройства, например, сообщать пользователю о наличии данных, поступающих с мышки. Обращение будет осуществляться через прерывание INT 33h, обработчик которого устанавливает драйвер мыши в операционной системе MS-DOS. Этот обработчик выполняет все операции, связанные с обслуживанием мыши:

1. сброс мыши и установка драйвера в исходное состояние;
2. включение и выключение курсора мыши;
3. установка курсора в определенное место экрана;
4. определение текущих координат курсора и текущего состояния клавиш;
5. определение координат курсора и состояния клавиш в момент нажатия на клавишу и в момент отпускания клавиши;
6. определение области на экране, в пределах которой может перемещаться курсор;
7. определение области на экране, в пределах которой курсор не будет виден;
8. определение формы графического и текстового курсоров;
9. определение величины перемещения мыши в сотых долях дюйма;
10. подключение к драйверу процедуры, определенной в программе, получающей управление при нажатии на заданную клавишу или при перемещении мыши;
11. запоминание и восстановление состояния драйвера;
12. управление эмуляцией светового пера;
13. управление скоростью движения курсора;
14. указание или определение используемой страницы видеопамяти;
15. управление драйвером мыши.

Для данной работы нам понадобился пункт 4.

Соответственно, нам потребовалось определить положение курсора и определить нажаты ли клавиши, следовательно, нам потребуются следующая функция прерывания INT 33h: 03h

03h – это функция прерывания INT 33h, которая возвращает текущие координаты курсора мыши и состояние клавиш. На входе мы получаем AX = 0003h, а на выходе: BX = состояние клавиш мыши; CX = координата X курсора DX = координата Y курсора. В регистре BX могут быть установлены следующие флаги, которые показаны в Таблице 1.

Таблица 1. Флаги регистра BX.

Установленный бит регистра BX	Клавиша, которая была нажата
0	Левая
1	Правая
2	Средняя

Для графических режимов координаты располагаются в различных диапазонах, в зависимости от текущего режима видеоадаптера. Конкретные примеры показаны в Таблице 2.

Таблица 2. Диапазоны координат курсора мыши.

Размер экрана в пикселях	Номер режима	Диапазон координат по оси X	Диапазон координат по оси Y
320x200	4,5	0-638	0-199
640x200	6	0-639	0-199
320x200	0Dh	0-638	0-199
640x200	0Eh	0-639	0-199
640x350	0Fh	0-639	0-349

Однако, в ходе написания программы была использована библиотека «windows.h», которая предоставляет доступ к нужной нам информации через API Windows, который, в свою очередь, предоставляет нужную нам информацию выше описанным способом.

ВЫБОР ПРОГРАММНОГО ПРОДУКТА

Для тестирования характеристик мыши ЭВМ была выбрана программа AIDA64

AIDA64 – самая популярная программа для определения и анализа аппаратного и программного обеспечения компьютера и подключаемых устройств, тестирования производительности и стабильности, а также мониторинга состояния ключевых узлов компьютера.

Это профессионально разработанный продукт является безоговорочным лидером в своем классе. Программа выросла из таких известных утилит как Everest и AIDA16/32 и имеет богатую историю развития. На данный момент база AIDA64 включает более 150000 разнообразных устройств.

Плюсы данной программы:

- AIDA64 предоставляет полные данные обо всём компьютере и внешних устройствах, в том числе и недокументированные параметры, которые обычно скрыты;
- Программа включает в себя набор тестов компьютера и позволяет сравнить полученный результат с эталонным;
- В программе присутствуют функции мониторинга в реальном времени
- Интуитивно понятный интерфейс на русском языке
- Приложение доступно не только для ЭВМ, но и для мобильных устройств на операционных системах Android и IOS

Из недостатков приложения AIDA64 можно выделить, что программа платная и ее пробный период составляет 30 дней.

ВЫБОР ЯЗЫКА ПРОГРАММИРОВАНИЯ

Для тестирования характеристик мыши ЭВМ из множества языков программирования выбор был сделан в сторону «С».

«С» - это язык, который часто называют «средне уровнем» или даже «низко уровнем» языком программирования, т.к. он сочетает элементы языков высокого уровня с функциональностью и производительностью ассемблера и работает близко к аппаратной части компьютера, что и послужило основным аргументом в сторону выбора данного языка. Сейчас «С» используется чаще всего в системном программировании, в частности, создание операционных систем, драйверов, различных утилит, антивирусов и т.д.

Из плюсов языка программирования «С» можно выделить:

- Универсальность – один и тот же код может быть скомпилирован почти на каждой платформе (при наличии для нее компилятора);
- Высокая скорость выполнения
- Компактность, небольшой размер выходных скомпилированных файлов;
- Малое потребление оперативной памяти;
- Простой синтаксис и небольшое количество возможностей языка упрощают изучение.

К недостаткам данного языка относится:

- Необходимость ручного выделения и освобождения памяти;
- Необходимость контроля за размерами массива;
- Необходимость контроля за типами переменных;
- Необходимость работы с указателями, что порождает множество возможностей для ошибок программиста;
- Необходимость работы с макросами.

АЛГОРИТМ ПРОГРАММЫ

Код программы на «С»:

```
#include <windows.h>
#include <stdio.h>
int main() {
    SetConsoleOutputCP(CP_UTF8); //Устанавливаем кодировку вывода
    UINT deviceCount; //Количество устройств ввода
    PRAWINPUTDEVICELIST pRawInputDeviceList; //Список устройств ввода
    //Получение количества (если первый аргумент функции = NULL) устройств
    //ввода, если функция возвращает 0,
    //то функция выполнена успешно, иначе выводится сообщение об ошибке
    if (GetRawInputDeviceList(NULL, &deviceCount, sizeof(RAWINPUTDEVICELIST)) !=
0) {
        printf("Ошибка при получении списка устройств\n");
        return 1;
    }
    //Выделение памяти под список устройств ввода, если память не выделена
    (NULL), то выводится сообщение об ошибке
    if ((pRawInputDeviceList =
(PRAWINPUTDEVICELIST)malloc(sizeof(RAWINPUTDEVICELIST) * deviceCount)) == NULL)
{
        printf("Ошибка при выделении памяти\n");
        return 1;
    }
    //Запись списка устройств ввода и проверка на ошибки при записи, если
    функция возвращает -1, то функция сработала
    //неверно и выводится сообщение об ошибке
    if (GetRawInputDeviceList(pRawInputDeviceList, &deviceCount,
sizeof(RAWINPUTDEVICELIST)) == -1) {
        printf("Ошибка при записи списка устройств\n");
        free(pRawInputDeviceList);
        return 1;
    }
    //Проходимся по списку устройств ввода
    for (UINT i = 0; i < deviceCount; ++i) {
        //Если устройство является мышью
        if (pRawInputDeviceList[i].dwType == RIM_TYPEMOUSE) {
            void *deviceHandle; //Указатель на устройство или же его дескриптор
            deviceHandle = (void*) pRawInputDeviceList[i].hDevice; //Присваиваем
дескриптор устройства
            UINT dataSize; // Размер строки-информации об устройстве
            //Получаем информацию об устройстве и выделяем память под строку-
информацию об устройстве
            GetRawInputDeviceInfo(deviceHandle, RIDI_DEVICENAME, NULL,
&dataSize);
            char* deviceName = (char*)malloc(dataSize);
            //Если функция GetRawInputDeviceInfo с аргументом deviceName (имя
устройства) возвращает отрицательное
            // число, то выводится сообщение об ошибке
            //Если вместо deviceName указать NULL, как выше, то мы получим как
раз таки размер
            // строки-информации об устройстве
            if (GetRawInputDeviceInfo(deviceHandle, RIDI_DEVICENAME, deviceName,
&dataSize) < 0)
                printf("Ошибка при получении имени устройства\n");
            //Выводим информацию об устройстве, т.е. Vendor ID и Product ID
            else {
```



```

        printf("Полный ID мыши: %s\n", deviceName);
        printf("Vendor ID и Product ID мыши: ");
        for (int j = 8; j < 25; ++j)
            printf("%c", deviceName[j] == '&' ? ' ' : deviceName[j]);
        printf("\n");
    }
    //Освобождаем память, которую мы выделяли
    free(deviceName);
    free(pRawInputDeviceList);
}

//Получаем системную информацию о мыши, в данном случае количество кнопок
мыши и наличие колеса мыши
//GetSystemMetrics - функция, которая получает информацию из системного
реестра Windows, однако
//функция не делает это напрямую, она использует API Windows, чтобы получить
информацию
int buttons = GetSystemMetrics(SM_CMOUSEBUTTONS);
boolean mouseWheel = GetSystemMetrics(SM_MOUSEWHEELPRESENT);
//Вывод информацию, которую мы получили выше
printf("Количество кнопок: %d\n", buttons);
printf("Колесо мыши: %s\n\n", mouseWheel ? "Да" : "Нет");
//Бесконечный цикл, который проверяет состояние кнопок мыши и координаты курсора
while(1) {
    //GetAsyncKeyState - функция, которая возвращает состояние клавиши, если
клавиша нажата, то функция возвращает
//ненулевое число, иначе 0
    SHORT leftButtonState = GetAsyncKeyState(VK_LBUTTON);
    if (leftButtonState != 0)
        printf("Левая кнопка мыши нажата!\n");
    SHORT rightButtonState = GetAsyncKeyState(VK_RBUTTON);
    if (rightButtonState != 0)
        printf("Правая кнопка мыши нажата!\n");
    SHORT middleButtonState = GetAsyncKeyState(VK_MBUTTON);
    if (middleButtonState != 0)
        printf("Средняя кнопка мыши нажата!\n");
    SHORT xButton1State = GetAsyncKeyState(VK_XBUTTON1);
    if (xButton1State != 0)
        printf("Кнопка X1 мыши нажата!\n");
    SHORT xButton2State = GetAsyncKeyState(VK_XBUTTON2);
    if (xButton2State != 0)
        printf("Кнопка X2 мыши нажата!\n");
    //GetCursorPos - функция, которая получает координаты курсора
    POINT cursorPos;
    if (GetCursorPos(&cursorPos))
        printf("Координаты курсора: x = %ld, y = %ld\n", cursorPos.x,
cursorPos.y);
    // GetAsyncKeyState(VK_ESCAPE) - функция, которая проверяет нажата ли
клавиша ESCAPE, если нажата, то
    // программа завершается
    if (GetAsyncKeyState(VK_ESCAPE)) {
        printf("Выход из программы\n");
        break;
    }
    //Задержка в 200 миллисекунд, чтобы не перегружать процессор
    Sleep(200);
}
return 0;
}

```

Дизассемблированный код:

```
main:
    push    %rbp
    mov     %rsp, %rbp
    sub     $0x60, %rsp
    call    0x7ff6ac8f1cb7 <__main>
    lea     -0x34(%rbp), %rax
    mov     $0x10, %r8d
    mov     %rax, %rdx
    mov     $0x0, %ecx
    mov     0xeb7e(%rip), %rax          # 0x7ff6ac9003b8
<__imp_GetRawInputDeviceList>
    call    *%rax
    test    %eax, %eax
    je      0x7ff6ac8f1859 <main+69>
    lea     0x97b9(%rip), %rax          # 0x7ff6ac8fb000
    mov     %rax, %rcx
    call    0x7ff6ac8f15e0 <printf>
    mov     $0x1, %eax
    jmp     0x7ff6ac8f1b9e <main+906>
    mov     -0x34(%rbp), %eax
    mov     %eax, %eax
    shl     $0x4, %rax
    mov     %rax, %rcx
    call    0x7ff6ac8f9558 <malloc>
    mov     %rax, -0x10(%rbp)
    cmpq    $0x0, -0x10(%rbp)
    jne     0x7ff6ac8f188e <main+122>
    lea     0x97cc(%rip), %rax          # 0x7ff6ac8fb048
    mov     %rax, %rcx
    call    0x7ff6ac8f15e0 <printf>
    mov     $0x1, %eax
    jmp     0x7ff6ac8f1b9e <main+906>
    lea     -0x34(%rbp), %rdx
    mov     -0x10(%rbp), %rax
    mov     $0x10, %r8d
    mov     %rax, %rcx
    mov     0xeb12(%rip), %rax          # 0x7ff6ac9003b8
<__imp_GetRawInputDeviceList>
    call    *%rax
    cmp     $0xffffffff, %eax
    jne     0x7ff6ac8f18d2 <main+190>
    lea     0x97cc(%rip), %rax          # 0x7ff6ac8fb080
    mov     %rax, %rcx
    call    0x7ff6ac8f15e0 <printf>
    mov     -0x10(%rbp), %rax
    mov     %rax, %rcx
    call    0x7ff6ac8f94f8 <free>
    mov     $0x1, %eax
    jmp     0x7ff6ac8f1b9e <main+906>
    movl    $0x0, -0x4(%rbp)
    jmp     0x7ff6ac8f19f4 <main+480>
    mov     -0x4(%rbp), %eax
    shl     $0x4, %rax
    mov     %rax, %rdx
    mov     -0x10(%rbp), %rax
    add     %rdx, %rax
    mov     0x8(%rax), %eax
    test    %eax, %eax
    jne     0x7ff6ac8f19f0 <main+476>
    mov     -0x4(%rbp), %eax
```

```

    shl     $0x4,%rax
    mov     %rax,%rdx
    mov     -0x10(%rbp),%rax
    add     %rdx,%rax
    mov     (%rax),%rax
    mov     %rax,-0x28(%rbp)
    lea     -0x38(%rbp),%rdx
    mov     -0x28(%rbp),%rax
    mov     %rdx,%r9
    mov     $0x0,%r8d
    mov     $0x20000007,%edx
    mov     %rax,%rcx
    mov     0xea7e(%rip),%rax          # 0x7ff6ac9003b0
<__imp_GetRawInputDeviceInfoA>
    call    *%rax
    mov     -0x38(%rbp),%eax
    mov     %eax,%eax
    mov     %rax,%rcx
    call    0x7ff6ac8f9558 <malloc>
    mov     %rax,-0x30(%rbp)
    lea     -0x38(%rbp),%rcx
    mov     -0x30(%rbp),%rdx
    mov     -0x28(%rbp),%rax
    mov     %rcx,%r9
    mov     %rdx,%r8
    mov     $0x20000007,%edx
    mov     %rax,%rcx
    mov     0xea4a(%rip),%rax          # 0x7ff6ac9003b0
<__imp_GetRawInputDeviceInfoA>
    call    *%rax
    lea     0x9759(%rip),%rax          # 0x7ff6ac8fb0c8
    mov     %rax,%rcx
    call    0x7ff6ac8f15e0 <printf>
    movl    $0x8,-0x8(%rbp)
    jmp     0x7ff6ac8f19c3 <main+431>
    mov     -0x8(%rbp),%eax
    movslq  %eax,%rdx
    mov     -0x30(%rbp),%rax
    add     %rdx,%rax
    movzbl  (%rax),%eax
    cmp     $0x26,%al
    je      0x7ff6ac8f19a9 <main+405>
    mov     -0x8(%rbp),%eax
    movslq  %eax,%rdx
    mov     -0x30(%rbp),%rax
    add     %rdx,%rax
    movzbl  (%rax),%eax
    movsbl  %al,%eax
    jmp     0x7ff6ac8f19ae <main+410>
    mov     $0x20,%eax
    mov     %eax,%edx
    lea     0x9734(%rip),%rax          # 0x7ff6ac8fb0eb
    mov     %rax,%rcx
    call    0x7ff6ac8f15e0 <printf>
    addl    $0x1,-0x8(%rbp)
    cmpl    $0x18,-0x8(%rbp)
    jle     0x7ff6ac8f1980 <main+364>
    lea     0x971e(%rip),%rax          # 0x7ff6ac8fb0ee
    mov     %rax,%rcx
    call    0x7ff6ac8f15e0 <printf>
    mov     -0x30(%rbp),%rax
    mov     %rax,%rcx
    call    0x7ff6ac8f94f8 <free>

```

```

mov     -0x10(%rbp),%rax
mov     %rax,%rcx
call    0x7ff6ac8f94f8 <free>
addl    $0x1,-0x4(%rbp)
mov     -0x34(%rbp),%eax
cmp     %eax,-0x4(%rbp)
jb      0x7ff6ac8f18de <main+202>
mov     $0xfde9,%ecx
mov     0xe83c(%rip),%rax          # 0x7ff6ac900248 <__imp_SetConsoleOutputCP>
call    *%rax
mov     $0x2b,%ecx
mov     0xe9a6(%rip),%rax          # 0x7ff6ac9003c0 <__imp_GetSystemMetrics>
call    *%rax
mov     %eax,-0x14(%rbp)
mov     $0x4b,%ecx
mov     0xe995(%rip),%rax          # 0x7ff6ac9003c0 <__imp_GetSystemMetrics>
call    *%rax
mov     %al,-0x15(%rbp)
mov     -0x14(%rbp),%eax
mov     %eax,%edx
lea     0x96b4(%rip),%rax          # 0x7ff6ac8fb0f0
mov     %rax,%rcx
call    0x7ff6ac8f15e0 <printf>
cmpl    $0x0,-0x15(%rbp)
je      0x7ff6ac8f1a53 <main+575>
lea     0x96c6(%rip),%rax          # 0x7ff6ac8fb117
jmp     0x7ff6ac8f1a5a <main+582>
lea     0x96c2(%rip),%rax          # 0x7ff6ac8fb11c
mov     %rax,%rdx
lea     0x96bf(%rip),%rax          # 0x7ff6ac8fb123
mov     %rax,%rcx
call    0x7ff6ac8f15e0 <printf>
mov     $0x1,%ecx
mov     0xe928(%rip),%rax          # 0x7ff6ac9003a0 <fthunk>
call    *%rax
mov     %ax,-0x18(%rbp)
cmpl    $0x0,-0x18(%rbp)
je      0x7ff6ac8f1a94 <main+640>
lea     0x96b4(%rip),%rax          # 0x7ff6ac8fb140
mov     %rax,%rcx
call    0x7ff6ac8f15e0 <printf>
mov     $0x2,%ecx
mov     0xe900(%rip),%rax          # 0x7ff6ac9003a0 <fthunk>
call    *%rax
mov     %ax,-0x1a(%rbp)
cmpl    $0x0,-0x1a(%rbp)
je      0x7ff6ac8f1abc <main+680>
lea     0x96bc(%rip),%rax          # 0x7ff6ac8fb170
mov     %rax,%rcx
call    0x7ff6ac8f15e0 <printf>
mov     $0x4,%ecx
mov     0xe8d8(%rip),%rax          # 0x7ff6ac9003a0 <fthunk>
call    *%rax
mov     %ax,-0x1c(%rbp)
cmpl    $0x0,-0x1c(%rbp)
je      0x7ff6ac8f1ae4 <main+720>
lea     0x96cc(%rip),%rax          # 0x7ff6ac8fb1a8
mov     %rax,%rcx
call    0x7ff6ac8f15e0 <printf>
mov     $0x5,%ecx
mov     0xe8b0(%rip),%rax          # 0x7ff6ac9003a0 <fthunk>
call    *%rax
mov     %ax,-0x1e(%rbp)

```

```

cmpw    $0x0,-0x1e(%rbp)
je      0x7ff6ac8f1b0c <main+760>
lea     0x96dc(%rip),%rax      # 0x7ff6ac8fb1e0
mov     %rax,%rcx
call    0x7ff6ac8f15e0 <printf>
mov     $0x6,%ecx
mov     0xe888(%rip),%rax      # 0x7ff6ac9003a0 <fthunk>
call    *%rax
mov     %ax,-0x20(%rbp)
cmpw    $0x0,-0x20(%rbp)
je      0x7ff6ac8f1b34 <main+800>
lea     0x96dc(%rip),%rax      # 0x7ff6ac8fb208
mov     %rax,%rcx
call    0x7ff6ac8f15e0 <printf>
lea     -0x40(%rbp),%rax
mov     %rax,%rcx
mov     0xe866(%rip),%rax      # 0x7ff6ac9003a8 <__imp_GetCursorPos>
call    *%rax
test    %eax,%eax
je      0x7ff6ac8f1b62 <main+846>
mov     -0x3c(%rbp),%edx
mov     -0x40(%rbp),%eax
mov     %edx,%r8d
mov     %eax,%edx
lea     0x96d6(%rip),%rax      # 0x7ff6ac8fb230
mov     %rax,%rcx
call    0x7ff6ac8f15e0 <printf>
mov     $0x1b,%ecx
mov     0xe832(%rip),%rax      # 0x7ff6ac9003a0 <fthunk>
call    *%rax
test    %ax,%ax
je      0x7ff6ac8f1b8b <main+887>
lea     0x96ec(%rip),%rax      # 0x7ff6ac8fb268
mov     %rax,%rcx
call    0x7ff6ac8f15e0 <printf>
mov     $0x0,%eax
jmp     0x7ff6ac8f1b9e <main+906>
mov     $0xc8,%ecx
mov     0xe6c1(%rip),%rax      # 0x7ff6ac900258 <__imp_Sleep>
call    *%rax
jmp     0x7ff6ac8f1a6c <main+600>
add     $0x60,%rsp
pop     %rbp
ret

```

ТЕСТЫ ПРОГРАММЫ И СРАВНИТЕЛЬНЫЙ АНАЛИЗ

```
Полный ID мыши: \\?\HID#VID_1532&PID_0098&MI_00#7&d895776&0&0000#{378de44c-56ef-11d1-bc8c-00a0c91405dd}
Vendor ID и Product ID мыши: VID_1532 PID_0098
Количество кнопок: 5
Колесо мыши: Да

Левая кнопка мыши нажата!
Координаты курсора: x = -264, y = 533
Координаты курсора: x = -704, y = 375
Координаты курсора: x = -651, y = 376
Средняя кнопка мыши нажата!
Координаты курсора: x = -675, y = 359
Правая кнопка мыши нажата!
Координаты курсора: x = -591, y = 381
Кнопка X2 мыши нажата!
Координаты курсора: x = -495, y = 383
Кнопка X1 мыши нажата!
Координаты курсора: x = -513, y = 495
Координаты курсора: x = -858, y = 541
Средняя кнопка мыши нажата!
Координаты курсора: x = -865, y = 534
Левая кнопка мыши нажата!
Координаты курсора: x = -901, y = 508
Координаты курсора: x = -753, y = 525
Координаты курсора: x = -688, y = 525
Левая кнопка мыши нажата!
Координаты курсора: x = -684, y = 555
Координаты курсора: x = -684, y = 555
Выход из программы

Process finished with exit code 0
```

Рис 1. Результаты работы программы










 Свойства устройства	
 Описание драйвера	USB Input Device
 Дата драйвера	28.01.2021
 Версия драйвера	10.0.15063.0
 Поставщик драйвера	Razer Inc
 INF-файл	oem27.inf
 INF Section	RzDev_Device.NT
 Аппаратный ID	USB\VID_1532&PID_0098&REV_0200&MI_01
 Сведения о размещении	0000.0014.0000.003.000.000.000.000

Рис 2. Информация о мышке полученная через AIDA 64

Type USB	Vendor ID 1532
-------------	-------------------

Vendor Details

Razer USA, Ltd

Рис 3. Полученные данные о компании-создателя устройства с интернет ресурса

Различий между нашей программой и AIDA64 не наблюдается, если не учитывать различия в полноте данных. Алгоритм, написанный на «С» предоставил краткую, но достоверную и, в большинстве случаев, самую полезную информацию о компьютерной мыши. Помимо это, мы сверились с интернет источником, который показал, что программа выдала точно верное устройство.

Однако, AIDA64 не смогла показать нам информацию о наличии кнопок и наличии колесика у мышки, однако данную информацию мы можем получить просто, взглянув на подключенную мышь, и использовать простые арифметические правила подсчёта.

ИТОГИ

По итогам данной работы был написан алгоритм на языке «С», который получает информацию о подключенном манипуляторе мышь. Также, в ходе данной работы были получены навыки работы с Windows API и его функциями. Помимо этого, был проведен сравнительный анализ между выбранной программой AIDA64 и нашим алгоритмом и была проведена сводка с интернет-ресурсом, чтобы удостовериться в правильности работы нашего алгоритма.

СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ

1. Device Hunt [Электронный ресурс] – Режим доступа
<https://devicehunt.com/> Дата обращения: 26.03.24