

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

Факультет прикладної математики

Кафедра прикладної математики

Звіт

до лабораторної роботи №1

із дисципліни «Програмування»

на тему

БАЗОВІ ТИПИ ДАНИХ, УВЕДЕННЯ-ВИВЕДЕННЯ, БІТОВІ ОПЕРАЦІЇ,
ОПЕРАЦІЇ ЗСУВУ

Виконав:

студент групи КМ-93

Пиндиківський Т. Р.

Керівник:

асистент Дрозденко О. М.

ЗМІСТ

МЕТА РОБОТИ.....	3
ПОСТАНОВКА ЗАДАЧІ.....	4
ОСНОВНА ЧАСТИНА	5
ВИСНОВКИ	10
ДОДАТОК А	11

МЕТА РОБОТИ

Здобути практичні навички роботи з типами даних мови програмування C, бітовими операціями, використання функцій стандартного введення-виведення.

ПОСТАНОВКА ЗАДАЧІ

Завдання 1:

1. Обчислити об'єм тора, утвореного обертанням кола радіуса r навколо вісі, яка відступає на відстань R від центру, за формулою $V = 2\pi^2 Rr^2$.

Завдання 2:

2. Обчислити радіус уписаного в трикутник кола.

Завдання 3: Програма упакування.

Дескриптор сегмента для системи віртуальної пам'яті подається у вигляді:

№ розрядів	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
значення	F	F	F	F	F	F	F	F	R	W	L	L	L	L	L	L

де F . . F — номер блока, з якого починається сегмент;

R — доступ для читання;

W — доступ для запису;

L . . L — розмір сегмента в блоках.

Завдання 4: Програма розпакування.

ОСНОВНА ЧАСТИНА

Завдання 1 :

Для виконання програми спочатку імпортуються наступні модулі :

- ***stdio.h*** – файл заголовку для стандартних операцій введення/виведення;
- ***math.h*** – бібліотека мови C, що містить функціонал для базових математичних операцій;

Змінні, що використовуються під час виконання завдання:

- ***bigger_radius*** (char) – рядкова величина, що дорівнює більшому радіусу тора;
- ***smaller_radius*** (char) – рядкова величина, що дорівнює меншому радіусу тора;
- ***bigger_radius_fl*** (float) - числове значення більшого радіусу тора;
- ***smaller_radius_fl*** (float) - числове значення меншого радіусу тора;

Хід виконання завдання:

1. Вводяться дані змінних *bigger_radius* та *smaller_radius*. Їхнє введення відбувається у тілі циклу *do-while*, де умовою виходу з циклу є отримання числових значень радіусів, які не є від'ємними.
2. Відбувається конвертація значень змінних *bigger_radius* та *smaller_radius* у *bigger_radius_fl*, *smaller_radius_fl*, які вже приймають числові значення типу float.
3. Згідно отриманих вхідних значень та формули обчислення об'єму тора знаходиться потрібний результат (використовується математична функція *pow()*), та виводиться у консоль.

Завдання 2 :

Для виконання програми спочатку імпортуються наступні модулі :

- ***stdio.h*** – файл заголовку для стандартних операцій введення/виведення;
- ***math.h*** – бібліотека мови C, що містить функціонал для базових математичних операцій;

Змінні, що використовуються під час виконання завдання:

- *first_side* (float) – числове значення першої сторони трикутника;
- *second_side* (float) – числове значення другої сторони трикутника;
- *third_side* (float) – числове значення третьої сторони трикутника;

Хід виконання завдання:

1. Вводяться дані змінних *first_side*, *second_side*, *third_side*. Введення кожного значення відбувається у тілі циклу *do-while*, де умовою виходу з циклу є отримання числового значення, яке не є від'ємним.
2. Перевіряється можливість існування трикутника при заданих параметрах : якщо трикутник не існує, то дані вводяться спочатку. Це продовжується до моменту, поки не буде отримано правильні значення.
3. Згідно отриманих вхідних значень та формули обчислення площі трикутника за трьома сторонами (використовується математична функція *sqrt()*) знаходиться шукане значення радіуса кола, вписаного в трикутник та виводиться отримане значення.

Завдання 3 :

Для виконання програми спочатку імпортуються наступні модулі :

- ***stdio.h*** – файл заголовку для стандартних операцій введення/виведення;

Змінні, що використовуються під час виконання завдання:

- *block_number[100]* (str) – рядкова величина, що визначає номер блока, з якого починається сегмент;
- *segment_size[100]* (str) – рядкова величина, що визначає розмір сегмента;
- *read_access[100]* (str) – рядкова величина, що визначає доступ до зчитування даних;
- *write_access[100]* (str) – рядкова величина, що визначає доступ до запису даних;

- *block_number_int* (int) - цілочисельна величина, що визначає номер блока, з якого починається сегмент;
- *segment_size_int* (int) – цілочисельне значення, що визначає розмір сегмента;
- *read_access_int* (int) – ціле число, що визначає доступ до зчитування даних;
- *write_access_int* (int) - ціле число, що визначає доступ до запису даних;
- *unitstatecode* (unsigned int) – ціле число, що визначає код стану дескриптора.

Хід виконання завдання:

1. Почергово вводяться дані про параметри стану дескриптора сегмента для системи віртуальної пам'яті: номер блока, розмір сегменту, доступ до зчитування та запису даних. Процес введення значення кожного елемента продовжується, поки користувач не введе дані правильного формату, а саме значення цілого типу з конкретного проміжку чисел. Цей процес виконується з використанням циклу *do-while()*.
2. Спочатку у змінній *unitstatecode* виділяється 8 молодших розрядів, завдяки виконанню побітового логічного І з константою FF та побітовому зсуву на 8 розрядів вліво.
3. Занесення в код стану ознаки доступу до зчитування даних : виконується логічне І з 1 та побітовий зсув на 7 розрядів вліво.
4. Занесення в код стану ознаки доступу до запису даних : виконується логічне І з 1 та побітовий зсув на 6 розрядів вліво.
5. Занесення в код стану розміру сегмента : виконується логічне І з 3F та логічне або з попереднім значення коду дескриптора для збереження попередніх значень.
6. В результаті застосування побітових логічних та операцій зсуву, отримаємо фінальний результат – код стану дескриптора у 16-ій системі числення.

Завдання 4 :

Для виконання програми спочатку імпортуються наступні модулі :

- ***stdio.h*** – файл заголовку для стандартних операцій введення/виведення;

Змінні, що використовуються під час виконання завдання:

- ***block_number*** (int) – числова величина, що визначає номер блока, з якого починається сегмент;
- ***segment_size*** (int) – числова величина, що визначає розмір сегмента;
- ***read_access*** (int) – числова величина, що визначає доступ до зчитування даних;
- ***write_access*** (int) – числова величина, що визначає доступ до запису даних;
- ***unitstatecode*** (unsigned int) – ціле число, що визначає код стану дескриптора.

Хід виконання завдання:

1. Вводиться код стану дескриптора у шістнадцятковій формі у тілі циклу do-while, поки не буде отримано значення правильного формату.
2. Виділення номера блоку сегмента дескриптора : виконується побітовий зсув на 8 розрядів вправо, а потім логічне І з шістнадцятковим числом FF.
3. Виділення стану доступу до читання даних дескриптора : виконується побітовий зсув на 7 розрядів вправо, а потім логічне І з числом 1.
4. Виділення стану доступу до запису даних дескриптора : виконується побітовий зсув на 6 розрядів вправо, а потім логічне І з числом 1.

5. Виділення розміра блоку сегмента дескриптора : виконується логічне І з шістнадцятковим числом 3F.
6. В результаті виводиться значення кожного окремого параметра коду стану дескриптора у консоль.

ВИСНОВКИ

На цій лабораторній роботі було здобуто практичні навички роботи з типами даних мови програмування C, бітовими операціями, використанням функцій стандартного введення-виведення.

ДОДАТОК А

1. Програмна реалізація задачі №1

```
#include<stdio.h>
#include<math.h>
//#include"isnumber.h"
void program1 ()
{
    char bigger_radius[100], smaller_radius[100];
    float bigger_radius_fl=0, smaller_radius_fl=0;
    float volume=0;

    do
    {
        printf("\n\nEnter the value of smaller radius : ");
        scanf("%s", &smaller_radius);
        getchar();
        if((!(isnumber(smaller_radius)) || (isnumber(smaller_radius) &&
!ispositivenumber(smaller_radius))))
            printf("\n\nYou've entered a wrong value of radius\n\n");

        } while(!(isnumber(smaller_radius)) || (isnumber(smaller_radius) &&
!ispositivenumber(smaller_radius))));

        smaller_radius_fl=atof(smaller_radius);

    do
    {
        printf("\n\nEnter the value of bigger radius : ");
        scanf("%s", &bigger_radius);
        getchar();
        if(!(isnumber(bigger_radius)) || (isnumber(bigger_radius) &&
!ispositivenumber(bigger_radius)))
            printf("\n\nYou've entered a wrong value of radius\n\n");
        } while(!(isnumber(bigger_radius)) || (isnumber(bigger_radius) &&
!ispositivenumber(bigger_radius))));

        bigger_radius_fl=atof(bigger_radius);
        volume=2*pow(M_PI, 2)*bigger_radius_fl*smaller_radius_fl;

        printf("\n\nThe volume of thor figure is : %f\n\n", volume);
    }
```

```
C:\Users\taras\Desktop\laboratory1_best\menu.exe
Taras Pyndykivskiy
Laboratory work #1
15th variant

Press the number of task you want to test ( 1 - 4 ) : 1

Task 1 : Find the volume of the thor using formula :

$$V=2\pi^2Rr^2$$


Enter the value of smaller radius : -9

You've entered a wrong value of radius

Enter the value of smaller radius : 53

Enter the value of bigger radius : 3

The volume of thor figure is : 3138.534180

If you want to continue testing program, press Space bar ...
```

Рис.1 – Тестування завдання №1

2. Програмна реалізація задачі №2

```
#include<stdio.h>
#include<math.h>
//#include"isnumber.h"
float enter_data(int side_number)
{
    char side_length[100];
    do
    {
        switch(side_number)
        {
            case 1:
                printf("\n\nEnter the length of the first side of
triangle : ");
```

```

        break;
    case 2:
        printf("\n\nEnter the length of the second side of
triangle : ");
        break;
    case 3:
        printf("\n\nEnter the length of the third side of
triangle : ");
        break;
    default:
        printf("Error");
        break;
    }
    scanf("%s", &side_length);
    getchar();
    if(!(isnumber(side_length)) || (isnumber(side_length) &&
(!ispositivenumber(side_length))) || (atof(side_length)==0))
    {
        if((!ispositivenumber(side_length) &&
isnumber(side_length)) || (atof(side_length)==0))
            printf("\n\nThe length of side of triangle can't be
less or equal to zero!!!");
        if(!(isnumber(side_length)))
            printf("\n\nYou've entered a wrong value of side
length");
        continue;
    }

    while(!isnumber(side_length) || (isnumber(side_length) &&
(!ispositivenumberr(side_length))) || (atof(side_length)==0)) ;

    return atof(side_length);
}

void program2()
{
    float first_side=0, second_side=0, third_side=0;

    first_side=enter_data(1);
    second_side=enter_data(2);
    third_side=enter_data(3);

    while((first_side+second_side<=third_side)|| (second_side+third_side<=
first_side)|| (first_side+third_side<=second_side))
    {
        printf("\n\nThe triangle with such sides can't exist !!!\n\n");
        first_side=enter_data(1);
        second_side=enter_data(2);
        third_side=enter_data(3);
    }

    float area=0, half_perimeter=0, radius=0;

    half_perimeter=(first_side+second_side+third_side)/2;

    area=sqrt(half_perimeter*(half_perimeter-first_side)*(half_perimeter-
second_side)*(half_perimeter-third_side));

    radius=area/half_perimeter;

```

```

        printf("\n\nThe radius of circle, built inside the triangle with
sides %.2f, %.2f, %.2f is : %.2f\n\n", first_side, second_side, third_side,
radius);
}

```

```

C:\Users\taras\Desktop\laboratory1_best\menu.exe
Taras Pyndykivskiy
Laboratory work #1
15th variant

Press the number of task you want to test ( 1 - 4 ) : 2

Task 2 : Find the radius of circle, built inside the triangle.

Enter the length of the first side of triangle : 4

Enter the length of the second side of triangle : 9

Enter the length of the third side of triangle : 7

The radius of circle, built inside the triangle with sides 4.00, 9.00, 7.00 is : 1.34

If you want to continue testing program, press Space bar ...

```

Рис.2 – Тестування завдання №2

3. Програмна реалізація задачі №3, 4

```

#include<stdio.h>
//#include"isnumber.h"

void program3_packing()
{
    char block_number[100], segment_size[100];
    char read_access[100], write_access[100];
    unsigned int unitstatecode;

    do
    {
        printf("\n\nEnter the number of block ( 0 - 255 ): ");
        scanf("%s", &block_number);
        getchar();
    } while((!ispositivenumber(block_number) || (
ispositivenumber(block_number) && (atoi(block_number)<0 ||
atoi(block_number)>255 ))) || (!isintnumber(block_number)));
    do
    {
        printf("\n\nEnter the state of read access ( 0 - 1 ): ");
        scanf("%s", &read_access);
        getchar();

    }while((!ispositivenumber(read_access) || (
ispositivenumber(read_access) && (atoi(read_access)<0 ||
atoi(read_access)>1 ))) || (!isintnumber(read_access)));
}

```

```

do
{
    printf("\n\nEnter the state of write access ( 0 - 1 ): ");
    scanf("%s", &write_access);
    getchar();
} while((!ispositivenumber(write_access) || (
ispositivenumber(write_access) && (atoi(write_access)<0 ||
atoi(write_access)>1 ))) || (!isintnumber(write_access)));
do
{
    printf("\n\nEnter the segment size ( 0 - 63 ): ");
    scanf("%s", &segment_size);
    getchar();
} while((!isnumber(segment_size) || ( isnumber(segment_size) && (
atoi(segment_size)<0 || atoi(segment_size)>63
)))) || (!isintnumber(segment_size)));

int block_number_int, segment_size_int;
int read_access_int, write_access_int;

block_number_int=atoi(block_number);
segment_size_int=atoi(segment_size);
read_access_int=atoi(read_access);
write_access_int=atoi(write_access);

unitstatecode=((unsigned int) block_number_int & 0xFF)<<8;
unitstatecode|=((unsigned int) read_access_int & 1)<<7;
unitstatecode|=((unsigned int) write_access_int & 1)<<6;
unitstatecode|=segment_size_int & 0x3F;

printf("\n\nDevice's state code : %04x", unitstatecode);
}

void program3_unpacking()
{
    char unitstatecode[100];
    do
    {
        printf("\n\nEnter hexadecimal number ( 0 - FFFF ) : ");
        scanf("%s", &unitstatecode);
        getchar();
    } while(!ishexadecimalnumber(unitstatecode) ||
(ishexadecimalnumber(unitstatecode) && strlen(unitstatecode)>4));

    int unitstatecode_int;
    sscanf(unitstatecode, "%x", &unitstatecode_int);

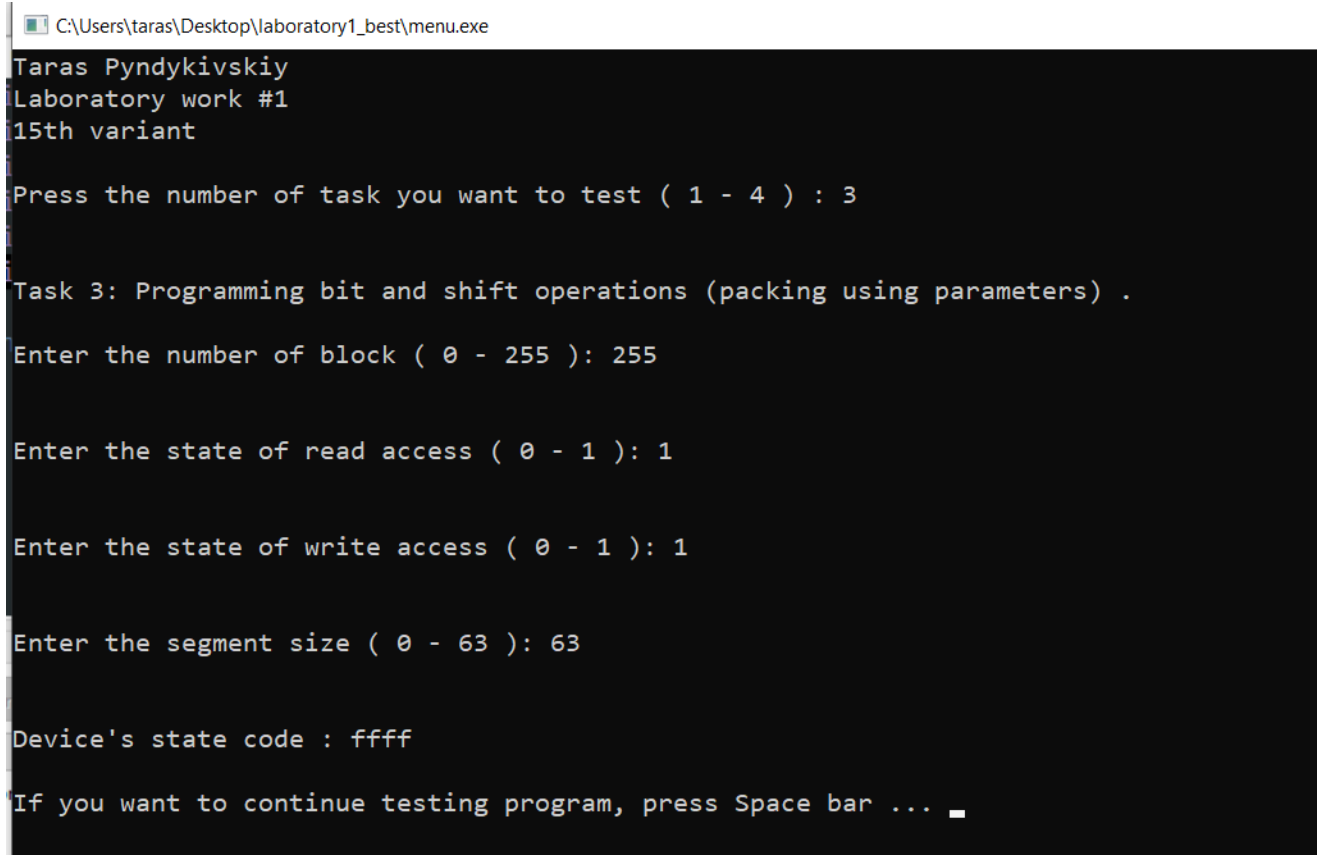
    unsigned char block_number, segment_size;
    unsigned char read_access, write_access;

    block_number=(unitstatecode_int >> 8) & 0xFF;
    read_access=(unitstatecode_int >> 7) & 1;
    write_access=(unitstatecode_int >> 6) & 1;
    segment_size=unitstatecode_int & 0x3F;

    printf("\n\nThe number of block is : %d", block_number);
    printf("\n\nThe state of read access : %d", read_access);
    printf("\n\nThe state of write access : %d", write_access);
    printf("\n\nThe size of segment is : %d", segment_size);

```

}



```
C:\Users\taras\Desktop\laboratory1_best\menu.exe
Taras Pyndykivskiy
Laboratory work #1
15th variant

Press the number of task you want to test ( 1 - 4 ) : 3

Task 3: Programming bit and shift operations (packing using parameters) .

Enter the number of block ( 0 - 255 ): 255

Enter the state of read access ( 0 - 1 ): 1

Enter the state of write access ( 0 - 1 ): 1

Enter the segment size ( 0 - 63 ): 63

Device's state code : ffff

If you want to continue testing program, press Space bar ...
```

Рис.3 – Тестування завдання №3

C:\Users\taras\Desktop\laboratory1_best\menu.exe

Taras Pyndykivskiy
Laboratory work #1
15th variant

Press the number of task you want to test (1 - 4) : 4

Task 4: Programming bit and shift operations (unpacking) .

Enter hexadecimal number (0 - FFFF) : AAA

Enter hexadecimal number (0 - FFFF) : FFFF

The number of block is : 255

The state of read access : 1

The state of write access : 1

The size of segment is : 63

If you want to continue testing program, press Space bar ... ☐

Рис.4 – Тестування завдання №4