Міністерство освіти і науки України

Національний авіаційний університет

Кафедра комп'ютерних систем та мереж

**Звіт про виконання модульної контрольної роботи № 1**

**з дисципліни:**

"Програмування"

**Варіант 6**

Виконав:

Студент групи КС-133Б  
Курко М.О

Прийняла:  
Вавіленкова А.І

Київ 202

1.

Комп’ютерна програма – набір інструкцій виражений у формі для зчитування комп’ютером, які приводять його у дію для виконання певної роботи для досягнення заданого результату.

Структура найпростішої С-програми – программа на С має таку структуру:

* Директиви препроцесора.
* Визначення функцій та глобальних об’єктів.

Наприклад:

#include <stdio.h>

int main()

{

printf("Hello world! \n");

return 0;

}

Де:

int main() – Функція, яка є точкою старту абсолютно для всіх програм написаних на С.

В середині ф-ції ми бачимо printf("Hello world! \n");- це інструкція.

І на кінець #include <stdio.h> - це підключення файлу .h для функціонування ф-ції printf за допомогою директиви include

Написати декілька рядків програмного коду для виведення на екран квадратів чисел від 10 до 20 включно –

#include <stdio.h>

#include <windows.h>

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

for (int i = 10; i <= 20; i++)

{

int square = i \* i;

printf("Квадрат числа %d = %d\n", i, square);

}

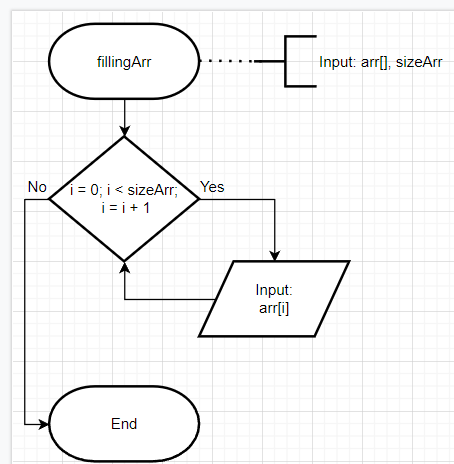
}

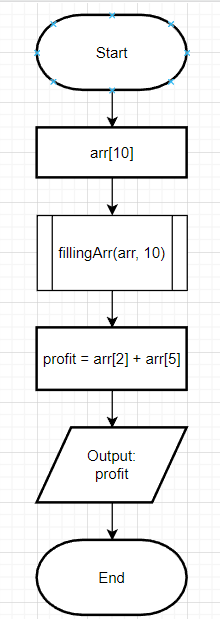
Розглянем деякі моменти програми:

#include <windows.h> - Підключення .h файлу для функціонування ф-цій SetConsoleCP та SetConsoleOutputCP - вони встановлюють мову для консолі в нашій програмі, в нашому випадку це українська.

for (int i = 10; i <= 20; i++) – цикл який пройде відрізок від 10 до 20 включно з кроком 1, відповідно в тілі циклу створюється змінна square яка зберігає в собі квадрат числа і, і відповідно ф-ція printf виводе це значення.

2. Написати код програми, якщо фінансова діяльність фірми задана значеннями її прибутків та збитків за останні 10 років: 77,3,-1,-7,15,31, -22,0,5,-3 (вводиться з клавіатури). Обчислити суму прибутків за третій та шостий роки. Намалювати схему алгоритму.





#define \_CRT\_SECURE\_NO\_WARNINGS

#include <Windows.h>

#include <stdio.h>

#define SIZE\_ARR 10

void fillingArr(int arr[], int sizeArr)

{

for (int i = 0; i < sizeArr; i++)

{

printf("Введіть прибуток за %d рік: ", (i+1));

int temp = 0;

scanf("%d", &temp);

arr[i] = temp;

}

}

int main(void)

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

int arr[SIZE\_ARR] = { 0 };

fillingArr(arr, SIZE\_ARR);

// рахунок в масивах починається з нуля, тому якщо ми хочемо отримати інформацію

// з 3 та 6 комірки масиву потрібно вказати індекси 2 та 5 відповідно

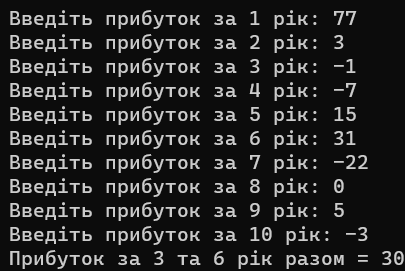
int profitOf3And6Years = arr[2] + arr[5];

printf("Прибуток за 3 та 6 рік разом = %d", profitOf3And6Years);

return 0;

}

Виведення в консоль має такий вигляд:



Код максимально простий, тому певно нам немає сенсу додатково пояснювати його.

3. Функції – це фрагмент кода до якого ми можемо звернутися за для того щоб він відпрацював.

Функції допомагають оптимізувати програмування, якщо вам потрібно використати якийсь фрагмент коду декілька раз його можна помістити в функції і звертатися до нього коли це буде потрібно. Також зазвичай весь код пишуть у функціях і лише викликають їх в основній функції main, наприклад як у минулому завданні. Це полегшує підтримку програми, так як вам не потрібно змінювати однаковий код в кожному місці де ви його використали а достатньо лиш винести його в функції і вам буде потрібно ввести лише змінну в цю функцію.

Функції не тільки можна робили в одному файлі з main а і можна зробити в окремому файлі і підключати їх в кожному проекті де вони нам потрібні.

Рекурсивні функції – це функції які викликають самі себе.

Рекурсивні функції досить важкі для розуміння та небезпечні при невірному використанню, однак це сильний інструмент який ми повинні розуміти і використовувати коли це має сенс.

Важливо вірно зазначити умову зупинки рекурсії, якщо це не зробити функція буде викликати себе вічно та ми отримаємо переповнення стеку.

Написати функцію, яка приймає три вказівники на цілі змінні та заміняє значення третьої з них добутком перших двох:

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <Windows.h>

#include <stdio.h>

int input(char letter)

{

printf("Введіть число %c : ", letter);

int num;

scanf("%d", &num);

return num;

}

void multiply1And2Into3(int \*a, int \*b, int \*c)

{

int temp = (\*a) \* (\*b);

\*c = temp;

}

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

int a = input('a');

int b = input('b');

int c = input('c');

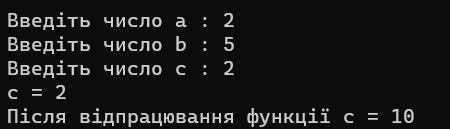
printf("c = %d\n", c);

multiply1And2Into3(&a, &b, &c);

printf("Після відпрацювання функції c = %d", c);

}

Вивід в консоль:



Є сенс зупинитися та розповісти про роботу ф-цій input та multiply1And2Into3.

Розпочнемо з першої, ця функція є досить простою, вона отримує літеру та далі вставляє її в стрічку при виведенні допоміжної інформації для користувача щоб було більш наглядно яку змінну ми в цей момент вводимо, відповідно функція після роботи повертає отримане число типу int. Важливо зауважити, що літера яку ми передаємо в ф-цію повинна бути або в одинарних лапках або ж у вигляді ASCII коду.

Ф-ція multiply1And2Into3 отримує вказівники на 3 числа, важливо зазначити що при передачі змінних в цю ф-ції потрібно подати їх за посиланням, тобто у такому вигляді: &x, де х це довільна змінна, далі значення змінної а та b перемножуються та кладуться в змінну temp після чого значення цеї змінної ми присвоюємо змінній с. Але потрібно бути уважним, тому що нам потрібно «розпакувати» змінну щоб отримати її значення за вказівником – це виконує операція \*, також, потрібно операцію «розпаковки» поставити в дужки щоб уникнути помилок в логікі програми.

4. Нехай фінансова діяльність кожної з n фірм задається у вигляді одновимірного масиву розмірністю m елементів (додатні елементи – це прибутки, від’ємні – це збитки). Необхідно визначити суму збитків по кожному підприємству у межах від -100 до -30, використовуючи функції та вказівники.

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <Windows.h>

#include <stdio.h>

#include <malloc.h>

int inputNumOfEnterprise()

{

printf("Введіть кількість підприємств: ");

int numOfEnterprise = 0;

scanf("%d", &numOfEnterprise);

return numOfEnterprise;

}

int inputYears()

{

printf("Введіть кількість років для аналізу: ");

int temp = 0;

scanf("%d", &temp);

return temp;

}

void initArray(int heigth, int weigth, int\*\* array)

{

for (int i = 0; i < weigth; i++)

{

array[i] = malloc(sizeof(\*\*array) \* heigth);

}

}

void fillingArray(int\*\* array, int heigth, int weigth)

{

for (int i = 0; i < weigth; i++)

{

for (int j = 0; j < heigth; j++)

{

printf("Введіть добуток %d компанії за %d місяць: ", (i + 1), (j + 1));

scanf("%d", &array[i][j]);

}

printf("\n");

}

}

void sumOfDamage(int\*\* arr, int\* outArr, int heigth, int weigth)

{

for (int i = 0; i < weigth; i++)

{

int temp = 0;

for (int j = 0; j < heigth; j++)

{

if ((arr[i][j] <= -30) && (arr[i][j] >= -100))

{

temp += arr[i][j];

}

}

outArr[i] = temp;

}

}

void outDynamicArray(int\* arr, int size)

{

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

{

printf("Сума збитків %d компанії які більші за -30 але менші за -100 = %d\n",(i + 1), arr[i]);

}

}

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

int yearsForAnalys = inputYears();

int numOfEnterprise = inputNumOfEnterprise();

int\*\* allEnterpriseStatistic = 0;

allEnterpriseStatistic = malloc(sizeof(\*allEnterpriseStatistic) \* numOfEnterprise);

int\* damageArr = 0;

damageArr = malloc(sizeof(\*damageArr) \* numOfEnterprise);

initArray(yearsForAnalys, numOfEnterprise, allEnterpriseStatistic);

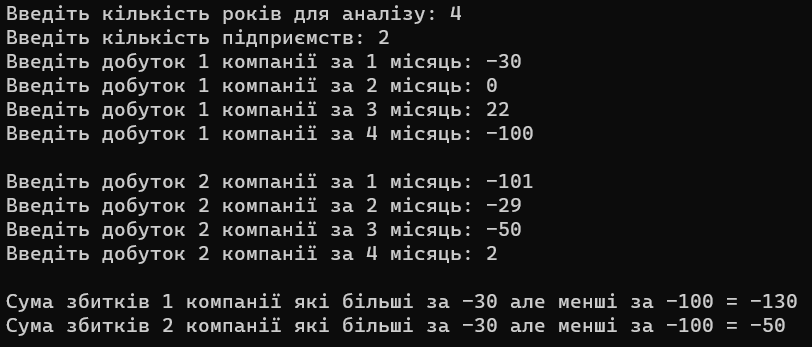
fillingArray(allEnterpriseStatistic, yearsForAnalys, numOfEnterprise);

sumOfDamage(allEnterpriseStatistic, damageArr, yearsForAnalys, numOfEnterprise);

outDynamicArray(damageArr, numOfEnterprise);

}

Вивід в консоль має такий вигляд:



Код досить примитивний, є сенс розглядати лише ф-ції: initArray , sumOfDamage .

Розпочнемо з першої, вона досить проста, по суті своїй вона просто створює масиви на кожний вказівник в нашому основному масиві вказівників, ось і вся робота цеї ф-ції, але сам інструмент разом нам дозволяє створювати динамічні масиви, в декількох словах дуже грубо це можна описати так :

* Створюємо динамічний масив вказівників
* Виділяємо пам'ять під нього за допомогою ф-ції malloc
* Викликаєм нашу ф-ції ініціалізації яка на кожний вказівник масиву створить свій динамічний масив певного заданого розміру типу int.

Друга ф-ція проходить циклом по всьому масиві та відповідно в кожному «стовпчику» сумує всі числа які меші за -30 але більші за -100 включно та записує їх спочатку в змінну temp а потім й у відповідну комірку одновимірного масиву, після переходу у наступний стовпчик змінна temp обнуляється.