**Практична робота №6. Розгляд прикладів програм лінійної, розгалуженої та циклічної структури, побудова блок-схем**

**Мета:** навчитися складати алгоритми програм лінійної, розгалуженої та циклічної структури**.**

**Завдання.**

1. Ознайомтеся з теоретичною частиною.
2. Намалюйте блок-схему до задачі розглянутій в теоретичній частині (рівняння *ax*+*b* =0) для різних пар a,b.
3. Зробіть постановку задачі з визначеннямвхідних та вихідних даних, розробіть блок-схему для наступних двох розрахунків та напишіть програму.

 якщо 

Запис  означає, х може приймати значення з діапазону [-1;2]. тобто х ≥ -1 та х ≤2. Запис  означає, що х змінюється з кроком 0.2.

 якщо    

1. Надайте відповідь на запитання: чи можна об’єднати ці розрахунки в одному циклі?

***Блок - схеми можна розробити в будь-якому наявному у вас редакторі, зокрема у Word, використовуючи графічні примітиви, які надаються в меню Вставка→Фігури. В крайньому випадку намалюйте в зошиті, сфотографуйте і вставте в файл з виконанням завдання як малюнок.***

Результати у вигляді текстового файлу надсилати на електронну адресу викладача [**t.i.lumpova@gmail.com**](mailto:t.i.lumpova@gmail.com)

Файл повинен мати назву в такому форматі:

**АР<Номер групи><Номер лекції / практичної / лабораторної [літера позначення типу роботи L – лекція, P – практична, R – лабораторна]<Прізвище англійською>**. Наприклад, **АРPTBD-2104P**buts.doc.

**Тему в заголовку листа записати**

**АР <Номер групи>-><Номер лекції / практичної / лабораторної [літера позначення типу роботи L – лекція, P – практична, R – лабораторна]<Прізвище англійською>**

**Строк виконання 13.03.2021**

**ТЕОРЕТИЧНА ЧАСТИНА**

**Проектування програми**.

Найчастіше алгоритм створюють, поступово уточнюючи поняття, пов'язані із задачею, і необхідні дії. Тоді кажуть, що розробку ведуть **згори донизу**.

Напишемо програму, що розв'язує рівняння *ax*+*b* =0.

**Уточнення постановки задачі**. Визначимо вхідні й вихідні дані програми. *Вхід*: коефіцієнти рівняння – два дійсних числа *a* та *b*. *Вихід*: кількість розв'язків; якщо розв'язок один – то саме цей розв'язок.

**Математичний аналіз задачі**. За умови *a* ≠0 рівняння має один розв'язок - *b*/*a*.

Тому потрібна перевірка, що вхідні дані коректні, тобто, що *a* ≠0.

**Проектування програми**.

У загальному вигляді алгоритм такий:

1. Отримати вхідні дані.

2. Обробити вхідні дані.

3. Вивести результат обробки.

Уточнимо кожен із кроків алгоритму.

"Отримати вхідні дані".

1.1. Вивести запрошення на введення даних.

1.2. Увести коефіцієнти рівняння в дійсні змінні **a** та **b**.

"Обробити вхідні дані". На основі аналізу задачі, якщо *a* ≠ 0, то кількість розв'язків дорівнює 1, а розв'язком є -*b*/*a*. Розв'язок запам'ятаємо в дійсній змінній **x**.

"Вивести результат обробки".

3.1. Вивести рівняння, уведене користувачем.

3.2. За допомогою значень змінної **x** вивести розв'язок, якщо він є, інакше вивести повідомлення про відсутність рішення.

Нарешті, можна кодувати.

**//програма, що розв'язує рівняння ax+b=0**

**#include <iostream>**

**using namespace std;**

**int main()**

**{**

**double a=0, b=0; //** коефіцієнти рівняння

**double x; // розв'язок рівняння**

**// отримати вхідні дані**

**cout<<"Enter coefficients a and b of " << "equation ax+b=0 (2 reals)\n";**

**cin>>a>>b;**

**// Підготовити виведення результату**

**cout<<"Equation "<<a<<"x+"<< b <<"=0 has ";**

**// обробити введені дані**

**if (a!=0)**

**{x=(-b)/a;**

**// повідомити результат**

**cout <<" solution " << x<<"\n";**

**}**

**else cout<<" no solution\n"; // повідомити результат**

**system("pause");**

**return 0;**

**}**

У цій програмі кожен фрагмент коду задає певні дії для отримання необхідного результату, тобто має своє призначення, або свій **обов'язок**. На перший погляд, програму можна зробити коротшою: якби обчислювати й відразу виводити розв'язок. Однак тоді код обробки даних був би *перевантажений обов'язками*, тобто відповідав за кілька різних функцій (тут – обчислення й виведення на екран).

Якщо кожен фрагмент коду має своє, персональне призначення, то це, по-перше, робить загальну структуру програми прозорішою і, по-друге, полегшує модифікацію окремих частин програми. У наведеному прикладі можна забажати змінити вихідне текстове повідомлення, і це не вплине на алгоритм обчислення результату. Отже, відокремлення обробки від виведення результатів цілком обґрунтоване.

Ускладнимо розглянуту задачу. Нехай розв’язок потрібно знайти не для однієї пари чисел *a* та *b*, а для якоїсь заздалегідь невідомої кількості їх комбінацій. В цьому випадку значення *a* та *b* потрібно вводити в циклі та запитувати можливість припинення. Для цього програма може бути модифікована таким чином.

**//програма, що розв'язує рівняння ax+b=0 для різних пар a,b**

**#include <iostream>**

**using namespace std;**

**int main()**

**{**

**double a=0, b=0; // коефіцієнти рівняння**

**double x; // розв'язок рівняння**

**int IsContinue=1; // Ознака продовження**

**while (IsContinue == 1)**

**{**

**// отримати вхідні дані**

**cout<<"Enter coefficients a and b of " << "equation ax+b=0 (2 reals)\n";**

**cin>>a>>b;**

**// Підготовити виведення результату**

**cout<<"Equation "<<a<<"x+"<< b <<"=0 has ";**

**// обробити введені дані**

**if (a!=0)**

**{x=(-b)/a;**

**// повідомити результат**

**cout <<" solution " << x<<"\n";**

**}**

**else cout<<" no solution\n"; // повідомити результат**

**cout <<" Enter 1 for continue \n ";**

**cin>> IsContinue;**

**}**

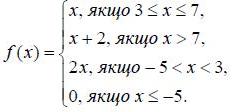
**system("pause");**

**return 0;**

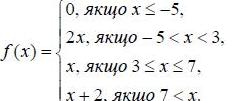
**}**

**Використання умовних операторів**

**Розглянемо приклад**. Написати фрагмент коду, що за дійсним *x* обчислює значення *f*(*x*) і присвоює його дійсній змінній *y*.



Перепишемо формулу обчислення *f*(*x*) у еквівалентному вигляді.



Якщо код писати безпосередньо за формулою, то такий фрагмент коду:

**if (x<=-5) y=0;**

**if (-5<x && x<3) y=2\*x;**

**if (3<=x && x<=7) y=x;**

**if (7<x) y=x+2;**

Цей код є правильним, але *неоптимальним* за кількістю виконуваних операцій. Якщо значенням змінної **x** є **-9.0**, то обчислюються всі чотири умови, хоча з погляду математики зрозуміло, що за істинності умови **x<=-5** решта умов хибні. Отже, модифікуємо фрагмент коду.

**if (x<=-5) y=0;**

**else if (-5<x && x<3) y=2\*x;**

**else if (3<=x && x<=7) y=x;**

**else if (7<x) y=x+2;**

Тепер за значення **-9.0** обчислюється тільки перша умова. Нехай значенням змінної **x** є **0.1**. Тоді вираз **x<=-5** є хибним, і обчислюється вираз **-5<x&&x<3**. Однак **x<=-5** хибний, тому **-5<x** є істинним! Отже, значенням виразу -**5<x&&x<3** є значення **x<3**. Міркуючи так само далі, отримуємо ще один варіант.

**if (x<=-5) y=0;**

**else if (x<3) y=2\*x; // тут значення -5<x істинне**

**else if (x<=7) y=x; // тут значення 3<=x істинне**

**else y=x+2; // тут значення 7<x істинне**

Зауважимо: наступний фрагмент коду для нашої задачі є *помилковим*.

**if (x<=-5) y=0;**

**if (x<3) y=2\*x;**

**if (x<=7) y=x;**

**else y=x+2;**

Якщо значенням **x** є **1.0**, то умова **x<3** істинна, тому спочатку виконується присвоювання **y=2\*x**. Проте потім перевіряється умова **x<=7**, виявляється істинною, і виконується **y=x**, що, вочевидь, є помилковим.

Дуже часто інструкції розгалуження є частиною інших розгалужень, тому їх записують "східцями", зсуваючи вкладену інструкцію праворуч, наприклад, таким чином:

**if (*умова1*)**

**if (*умова2*)**

***інструкція1***

**else**

***інструкція2***

**else ...**

Інколи виникають довгі ланцюги розгалужень, в яких за словами **else** йдуть наступні розгалуження з **if** на початку. Краще записувати їх у такому вигляді:

**if (*умова*)**

***інструкція***

**else if (*умова*)**

***інструкція***

**else if (*умова*)**

**...**