МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Харківський національний університет імені В.Н. Каразіна

Навчально-науковий інститут комп’ютерних наук та штучного інтелекту

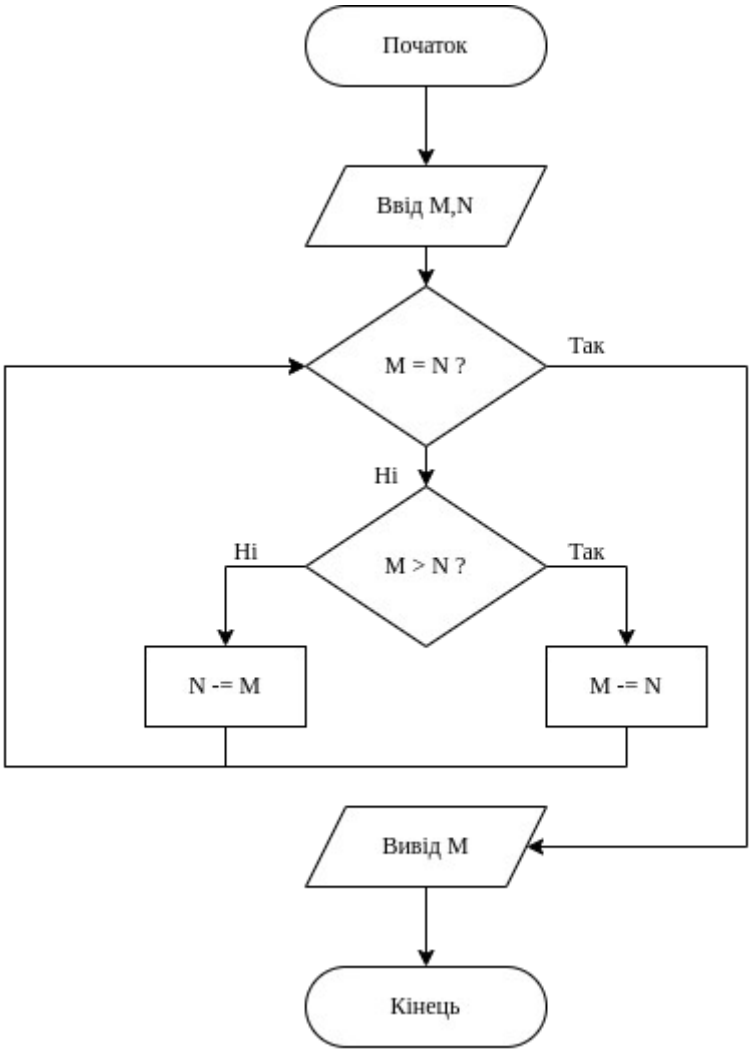
ПРАКТИЧНА РОБОТА №1

на тему: «Прості числа»

Виконав: студент 2 курсу групи КС22
Спеціальності 122 «Комп’ютерні науки»
Скрипняк Тарас Артемович
Прийняв: викладач
Олешко О. І.

Харків – 2024

Завдання 1. Реалізувати алгоритм Евкліда.
/home/bouncytorch/Repos/homework-c/pr1/tarik/1.png



Блок-схема

Пояснення:

1. Програма запитує у користувача два числа M і N і зберігає їх у змінних типу unsigned int (що означає невід'ємні цілі числа).
2. Це основна частина коду, яка використовує алгоритм Евкліда для знаходження НСД.
   - Цикл while продовжує виконуватись, поки числа M і N не стануть рівними.
   - Якщо M більше за N, то від M віднімається N (M -= N).
   - Якщо ж навпаки, N більше за M, то від N віднімається M (N -= M).
   - Таким чином, на кожній ітерації ми зменшуємо більше число на менше, поки вони не стануть рівними. У кінці циклу це число буде найбільшим спільним дільником.
3. Після завершення циклу, коли M дорівнює N, це значення є НСД двох початкових чисел. Програма виводить його на екран.

```c
#include <stdio.h>

int main() {
  unsigned int M, N;
  printf("Введіть число M: ");
  scanf("%d", &M);
  printf("Введіть число N: ");
  scanf("%d", &N);

  while (M != N) {
    if (M > N) M -= N;
    else N -= M;
  }

  printf("Дільник: %d\n", M);
  return 0;
}
```

Лістинг

```
bouncytorch@AORUS:~/Repos/homework-c/pr1/tarik$ ./a.out
Введіть число M: 69
Введіть число N: 240
Дільник: 3
```

Результат виконнаня

Завдання 2. Реалізувати решето Ератосфена.

Блок-схема

```c
#include <stdio.h>

int main() {
  int n;
  printf("Введіть ціле число більше 2: ");
  scanf("%d", &n);

  int primes[n + 1];

  // позначаємо всі числа як парні
  for (int i = 0; i <= n; i++) primes[i] = 1;

  // помічаємо нулем всі числа, що кратні на прості 2, 3, 5 та 7 за допомогою цього алгоритма
  for (int p = 2; p * p <= n; p++)
    // перевірка в раз того, що програма вже прошла через даний сет чисел кратних на два
    if (primes[p] == 1)
      for (int i = p * p; i <= n; i += p) primes[i] = 0;
```

```
  printf("Прості числа до %d:\n", n);
  // виводимо всі числа, помічені як прості
  for (int p = 2; p <= n; p++) if (primes[p]) printf("%d ", p);

  printf("\n");
  return 0;
}
```

Лістинг

```
bouncytorch@AORUS:~/Repos/homework-c/pr1/tarik$ ./a.out
Введіть ціле число більше 2: 1230
Прості числа до 1230:
2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 61 67 71 73 79 83 89 97 101 103 107 109 113 127 131 137 139
149 151 157 163 167 173 179 181 191 193 197 199 211 223 227 229 233 239 241 251 257 263 269 271 277 281 283
293 307 311 313 317 331 337 347 349 353 359 367 373 379 383 389 397 401 409 419 421 431 433 439 443 449 457
461 463 467 479 487 491 499 503 509 521 523 541 547 557 563 569 571 577 587 593 599 601 607 613 617 619 631
641 643 647 653 659 661 673 677 683 691 701 709 719 727 733 739 743 751 757 761 769 773 787 797 809 811 821
823 827 829 839 853 857 859 863 877 881 883 887 907 911 919 929 937 941 947 953 967 971 977 983 991 997
1009 1013 1019 1021 1031 1033 1039 1049 1051 1061 1063 1069 1087 1091 1093 1097 1103 1109 1117 1123 1129
1151 1153 1163 1171 1181 1187 1193 1201 1213 1217 1223 1229
```

Результат виконання

Завдання 3: Реалізувати решето Ератосфена.

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

// Функція для реалізації решета Аткіна
void atkinSieve(int n) {
  // Створюємо масив для позначення можливих простих чисел
  int *sieve = (int*)calloc(n + 1, sizeof(int));

  for (int x = 1; x * x <= n; x++) {
    for (int y = 1; y * y <= n; y++) {
      int num = 4 * x * x + y * y;
      if (num <= n && (num % 12 == 1 || num % 12 == 5)) {
        sieve[num] ^= 1;
      }

      num = 3 * x * x + y * y;
      if (num <= n && num % 12 == 7) {
        sieve[num] ^= 1;
      }

      num = 3 * x * x - y * y;
      if (x > y && num <= n && num % 12 == 11) {
        sieve[num] ^= 1;
      }
    }
  }

  // Позначаємо всі кратні квадратів як непотрібні
  for (int r = 5; r * r <= n; r++) {
    if (sieve[r]) {
      for (int i = r * r; i <= n; i += r * r) {
        sieve[i] = 0;
      }
    }
  }

  // Виводимо числа 2 і 3
  printf("2 ");
  if (n >= 3) printf("3 ");

  // Виводимо інші прості числа
  for (int i = 5; i <= n; i++)
    if (sieve[i])
      printf("%d ", i);

  printf("\n");

  // Звільнення пам'яті
  free(sieve);
```

```
}

int main() {
  int n;
  printf("Введіть межу для пошуку простих чисел: ");
  scanf("%d", &n);
  if (n < 2) return printf("Межа має бути більше двох");

  printf("Прості числа менші за %d:\n", n);
  atkinSieve(n);

  return 0;
}
```

Лістинг

```
bouncytorch@AORUS:~/Repos/homework-c/pr1/tarik$ ./a.out
Введіть межу для пошуку простих чисел: 1230
Прості числа менші за 1230:
2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 61 67 71 73 79 83 89 97 101 103 107 109 113 127 131 137 139
149 151 157 163 167 173 179 181 191 193 197 199 211 223 227 229 233 239 241 251 257 263 269 271 277 281 283
293 307 311 313 317 331 337 347 349 353 359 367 373 379 383 389 397 401 409 419 421 431 433 439 443 449 457
461 463 467 479 487 491 499 503 509 521 523 541 547 557 563 569 571 577 587 593 599 601 607 613 617 619 631
641 643 647 653 659 661 673 677 683 691 701 709 719 727 733 739 743 751 757 761 769 773 787 797 809 811 821
823 827 829 839 853 857 859 863 877 881 883 887 907 911 919 929 937 941 947 953 967 971 977 983 991 997
1009 1013 1019 1021 1031 1033 1039 1049 1051 1061 1063 1069 1087 1091 1093 1097 1103 1109 1117 1123 1129
1151 1153 1163 1171 1181 1187 1193 1201 1213 1217 1223 1229
```

Результат виконання