КПІ ім. Ігоря Сікорського

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

Звіт до комп'ютерного практикуму з курсу «Основи програмування»

Прийняв
асистент кафедри ІПІ
Ахаладзе А. Е.
«1» листопада 2024 р.

Виконав
студент групи ІП-43
Дутов І. А.

Київ 2024

Комп'ютерний практикум №3

Тема: *Програмування розгалужених алгоритмів*

Завдання:
Написати програму для обчислення числа $y = \sqrt[k]{x}$ із заданою точністю ε.

Текст програми

../src/main.c

```c
#include "input/input.h"
#include "root/root.h"

#include <float.h>
#include <limits.h>
#include <math.h>
#include <stdio.h>
#include <string.h>

void process_arguments(int argc, char *argv[], int *is_quiet_mode,
                       int *is_repeat_mode, int *is_help_mode);
void print_help();
void print_demo();

int main(int argc, char *argv[]) {
  int is_quiet_mode, is_repeat_mode, is_help_mode;
  is_quiet_mode = is_repeat_mode = is_help_mode = 0;

  process_arguments(argc, argv, &is_quiet_mode, &is_repeat_mode, &is_help_mode)
      ;

  if (is_help_mode) {
    print_help();
    return 0;
  }

  print_demo();
  do {

    long double x, e;
    int k, decimal_places;
    x = e = k = decimal_places = 0;

    while (read_long_double(&x, "підкореневий вираз x", "x", MAX_CHARS_X, 1,
                            is_quiet_mode, MAX_VAL_X, -MAX_VAL_X, 1, 1) != 0 ||
           (x < 0 && check_long_double_meets_restrictions(&x, "x", -MIN_VAL_X,
                                                          -MAX_VAL_X, 1, 1)) ||
           (x > 0 && check_long_double_meets_restrictions(&x, "x", MAX_VAL_X,
                                                          MIN_VAL_X, 1, 1)))
      ;

    while (read_int(&k, "показник кореня k", "k", MAX_CHARS_K, 1, is_quiet_mode
        ,
                    MAX_VAL_K, MIN_VAL_K, 1, 1) != 0 ||
           (x < 0 && k % 2 == 0 &&
            display_error("Неможливо порахувати корінь парного показнику з "
                          "відємного' числа.")))
      ;
```

```
    printf(GREEN "За замовчуванням тип вводу точності - через точність e.\n"
                 "Якщо ви бажаєте обрати ввід через кількість знаків після "
                 "коми D, введіть 0" RESET "\n");

    while (read_precision_or_decimal_places(&e, &decimal_places, 1,
                                            is_quiet_mode, MAX_VAL_E,
                                            MIN_VAL_E) != 0)
      ;

    const long double result = approximate_kth_root(
        x, k, e, decimal_places, is_quiet_mode, 0, DEFAULT_ITERATION_LIMIT);
    if (result == INFINITY) {
      display_error(
          "Послідовність не зійшлась через неточність в розрахунках.");
    } else {
      display_success("Кінцевий результат у: %.*Lf.", decimal_places, result);
    }
  } while (is_repeat_mode);
  return 0;
}

void process_arguments(int argc, char *argv[], int *is_quiet_mode,
                       int *is_repeat_mode, int *is_help_mode) {
  for (int i = 0; i < argc; i++) {
    if (strcmp("-q", argv[i]) == 0 || strcmp("--quiet", argv[i]) == 0) {
      *is_quiet_mode = 1;
    } else if (strcmp("-r", argv[i]) == 0 || strcmp("--repeat", argv[i]) == 0)
      {
      *is_repeat_mode = 1;
    } else if (strcmp("-h", argv[i]) == 0 || strcmp("--help", argv[i]) == 0) {
      *is_help_mode = 1;
    }
  }
}
void print_help() {
  printf("Програму можна запускати з такими параметрами:\n");
  printf("-q або --quiet: тихий«» режим менше( виводу)\n");
  printf("-r або --repeat: програма нескінченно повторюється\n");
  printf("-h або --help: вивести цю памятку'\n");
}

void print_demo() {
  printf("Вас вітає Find The KeX — програма для знаходження\n"
         "кореню kго-показника з x з точністю e\n");

  printf("Наші обмеження такі:\n");

  printf("\n%Lg ≤ x ≤ %Lg або %Lg ≤ x ≤ %Lg. ", MIN_VAL_X, MAX_VAL_X,
         -MAX_VAL_X, -MIN_VAL_X);
  printf("Довжина x в символах ≤ %u.\n", MAX_CHARS_X);

  printf("%d ≤ k ≤ %d. ", MIN_VAL_K, MAX_VAL_K);
  printf("Довжина k в символах ≤ %u.\n", MAX_CHARS_K);

  printf(
      "\Можнаn вводити щось одне: кількість знаків після коми D або точність "
      "e.\n");
  printf("%u ≤ D ≤ %u\n", MIN_DECIMAL_PLACES, MAX_DECIMAL_PLACES);
  printf("%Lg < e < %Lg\n", MIN_VAL_E, MAX_VAL_E);
}
```

../src/input/input.h

```c
#ifndef INPUT_H
#define INPUT_H

#define RED "\033[31m"
#define GREEN "\033[32m"
#define YELLOW "\033[33m"
#define RESET "\033[0m"

#define MAX_VAL_X (long double)1e15
#define MIN_VAL_X (long double)1e-15
#define MAX_CHARS_X (15 + 6 + 3) // 'e', '-', and '.'

#define MAX_VAL_K 1000000
#define MIN_VAL_K -1000000
#define MAX_CHARS_K 7

#define MIN_DECIMAL_PLACES 1
#define MAX_DECIMAL_PLACES 15
#define MAX_CHARS_DECIMAL_PLACES 2

#define MAX_VAL_E (long double)1
#define MIN_VAL_E (long double)1e-15
#define MAX_CHARS_E (15 + 3) // 'e', '-', and '.'

#define TOLERANCE (long double)1e-20

#define MAX_LONG_DOUBLE_DECIMAL_PLACES 308
#define MAX_CHARS_LONG_DOUBLE (MAX_LONG_DOUBLE_DECIMAL_PLACES + 3)
#define MAX_CHARS_INT (10 + 3)
#define MAX_SIGNIFICANT_DIGITS_LONG_DOUBLE 15

int display_error(const char *format, ...)
    __attribute__((format(printf, 1, 2)));
int display_warning(const char *format, ...)
    __attribute__((format(printf, 1, 2)));
int display_success(const char *format, ...)
    __attribute__((format(printf, 1, 2)));

void replace_commas_with_dots(char *string);
int validate_input_precision(const char *input, int max_significant_digits);

void set_nonblocking_input();
void reset_input_mode();

int read_long_double(long double *value, const char *full_name,
                     const char *short_name, int max_char_count,
                     int is_restricted, int is_quiet_mode,
                     long double max_value, long double min_value,
                     int is_max_included, int is_min_included);

int read_precision_or_decimal_places(long double *precision,
                                     int *decimal_places, int is_restricted,
                                     int is_quiet_mode,
                                     long double max_value_precision,
                                     long double min_value_precision);

int read_int(int *value, const char *full_name, const char *short_name,
             int max_char_count, int is_restricted, int is_quiet_mode,
```

```
                int max_value, int min_value, int is_max_included,
                int is_min_included);

int check_long_double_meets_restrictions(long double *value, const char *name,
                                         long double max_value,
                                         long double min_value,
                                         int is_max_included,
                                         int is_min_included);

int check_int_meets_restrictions(int *value, const char *name, int max_value,
                                 int min_value, int is_max_included,
                                 int is_min_included);

#endif
```

../src/input/input.c

```
#include "input.h"

#include <math.h>
#include <stdarg.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

void replace_commas_with_dots(char *string) {
  while (*string) {
    if (*string == ',') {
      *string = '.';
    }
    string++;
  }
}

int validate_input_precision(const char *input, int max_significant_digits) {
  int significant_digits = 0;

  for (size_t i = 0; i < strlen(input); i++) {
    char c = input[i];

    if (c >= '0' && c <= '9') {
      significant_digits++;
    }
  }

  if (significant_digits > max_significant_digits) {
    display_warning("Ввід перевищує максимальну дозволену кількість значущих "
                    "цифр %d. Розрахунки можуть бути неточними",
                    max_significant_digits);
    return 1;
  }

  return 0;
}

int display_error(const char *format, ...) {
  va_list args;
  va_start(args, format);
```

```
  printf("\n" RED "ПОМИЛКА! ");
  vprintf(format, args);
  printf(RESET "\n");

  va_end(args);
  return 1;
}

int display_warning(const char *format, ...) {
  va_list args;
  va_start(args, format);

  printf("\n" YELLOW "УВАГА! ");
  vprintf(format, args);
  printf(RESET "\n");

  va_end(args);
  return 1;
}

int display_success(const char *format, ...) {
  va_list args;
  va_start(args, format);

  printf("\n" GREEN "ПЕРЕМОГА! ");
  vprintf(format, args);
  printf(RESET "\n");

  va_end(args);
  return 1;
}

int check_long_double_meets_restrictions(long double *value, const char *name,
                                         long double max_value,
                                         long double min_value,
                                         int is_max_included,
                                         int is_min_included) {

  if (is_min_included) {
    if (*value < min_value - TOLERANCE) {
      display_error("%s має бути більшийрівний- %Lg.", name, min_value);
      return 1;
    }
  } else {
    if (*value <= min_value + TOLERANCE) {
      display_error("%s має бути більший за %Lg.", name, min_value);
      return 1;
    }
  }

  if (is_max_included) {
    if (*value > max_value + TOLERANCE) {
      display_error("%s має бути меншийрівний- %Lg.", name, max_value);
      return 1;
    }
  } else {
    if (*value >= max_value - TOLERANCE) {
      display_error("%s має бути менший за %Lg.", name, max_value);
      return 1;
    }
```

```
  }

  return 0;
}

int read_long_double(long double *value, const char *full_name,
                     const char *short_name, int max_char_count,
                     int is_restricted, int is_quiet_mode,
                     long double max_value, long double min_value,
                     int is_max_included, int is_min_included) {
  if (is_quiet_mode == 1) {
    printf("%s: ", short_name);
  } else if (is_restricted == 1) {
    printf("Введіть %s від( %Lg до %Lg): ", full_name, min_value, max_value);
  } else {
    printf("Введіть %s: ", full_name);
  }

  char input[max_char_count + 2];
  char *endptr;

  if (!fgets(input, max_char_count + 2, stdin)) {
    display_error("Не вдалося прочитати ввід для %s.", full_name);
    return 1;
  }

  if (input[strlen(input) - 1] != '\n') {
    display_error("Довжина %s в символах має бути меншою за %u.", short_name,
                  max_char_count);
    int ch;
    while ((ch = getchar()) != '\n' && ch != EOF)
      ;
    return 1;
  }

  replace_commas_with_dots(input);

  *value = strtold(input, &endptr);
  if (endptr == input || *endptr != '\n') {
    display_error("%s має бути числом і не містити додаткових символів!",
                  short_name);
    return 1;
  }

  if (is_restricted == 1 && check_long_double_meets_restrictions(
                                value, short_name, max_value, min_value,
                                is_max_included, is_min_included) == 1) {
    return 1;
  }

  validate_input_precision(input, MAX_SIGNIFICANT_DIGITS_LONG_DOUBLE);

  return 0;
}

int read_precision_or_decimal_places(long double *precision,
                                     int *decimal_places, int is_restricted,
                                     int is_quiet_mode,
                                     long double max_value_precision,
                                     long double min_value_precision) {
```

```
  while (read_long_double(precision, "точність e", "e", MAX_CHARS_E, 0,
                          is_quiet_mode, 0, 0, 0, 0) != 0)
    ;

  if (*precision == 0) {
    while (read_int(decimal_places, "кількість знаків після коми D", "D",
                    MAX_CHARS_DECIMAL_PLACES, is_restricted, is_quiet_mode,
                    MAX_DECIMAL_PLACES, MIN_DECIMAL_PLACES, 1, 1) != 0)
      ;
    *precision = powl(10.0L, (long double)(-*decimal_places));
    return 0;
  }

  if (is_restricted) {
    if (check_long_double_meets_restrictions(precision, "e",
                                             max_value_precision,
                                             min_value_precision, 0, 1) != 0) {
      return 1;
    }
  }

  *decimal_places = (int)-log10l(*precision);
  return 0;
}

int check_int_meets_restrictions(int *value, const char *name, int max_value,
                                 int min_value, int is_max_included,
                                 int is_min_included) {

  if (is_min_included) {
    if (*value < min_value) {
      display_error("%s має бути більшийрівний- %d.", name, min_value);
      return 1;
    }
  } else {
    if (*value <= min_value) {
      display_error("%s має бути більший за %d.", name, min_value);
      return 1;
    }
  }

  if (is_max_included) {
    if (*value > max_value) {
      display_error("%s має бути меншийрівний- %d.", name, max_value);
      return 1;
    }
  } else {
    if (*value >= max_value) {
      display_error("%s має бути менший за %d.", name, max_value);
      return 1;
    }
  }

  return 0;
}

int read_int(int *value, const char *full_name, const char *short_name,
             int max_char_count, int is_restricted, int is_quiet_mode,
             int max_value, int min_value, int is_max_included,
             int is_min_included) {
```

```
  if (is_quiet_mode == 1) {
    printf("%s: ", short_name);
  } else if (is_restricted == 1) {
    printf("Введіть %s від( %d до %d): ", full_name, min_value, max_value);
  } else {
    printf("Введіть %s: ", full_name);
  }

  char input[max_char_count + 2];
  char *endptr;

  if (!fgets(input, max_char_count + 2, stdin)) {
    display_error("Не вдалося прочитати ввід для %s.\n", full_name);
    return 1;
  }
  if (input[strlen(input) - 1] != '\n') {
    display_error("Довжина %s в символах має бути меншою за %d.\n", short_name,
                  max_char_count);
    int ch;
    while ((ch = getchar()) != '\n' && ch != EOF)
      ;
    return 1;
  }

  replace_commas_with_dots(input);

  long double input_num = strtold(input, &endptr);
  if (endptr == input || *endptr != '\n') {
    display_error("%s має бути числом і не містити додаткових символів!\n",
                  short_name);
    return 1;
  }

  if (floor(input_num) != input_num) {
    display_error("%s має бути цілим числом.\n", short_name);
    return 1;
  }

  *value = (int)input_num;

  if (is_restricted == 1 &&
      check_int_meets_restrictions(value, short_name, max_value, min_value,
                                   is_max_included, is_min_included)) {
    return 1;
  }

  return 0;
}
```

../src/root/root.h

```
#ifndef ROOT_H
#define ROOT_H

#define DEFAULT_ITERATION_LIMIT 1000000

long double approximate_kth_root(long double x, int k, long double e_target,
                                 int decimal_places, int is_quiet_mode,
```

```
                                    int is_return_delta, int iteration_limit);
#endif
```

../src/root/root.c

```
#include "root.h"
#include "..//input/input.h"

#include <limits.h>
#include <math.h>
#include <stdio.h>

long double approximate_kth_root(long double x, int k, long double e_target,
                                 int decimal_places, int is_quiet_mode,
                                 int is_return_delta, int iteration_limit) {

  if (k == 0) {
    if (!is_quiet_mode)
      display_error("Кореня нульового степеня не існує, спробуйте ще раз.");
    return INFINITY;
  }

  if (k % 2 == 0 && x < 0) {
    display_error(
        "Кореня парного степеня з відємного' числа не існує, спробуйте ще "
        "раз");
    return INFINITY;
  }

  if (!is_quiet_mode) {
    printf("%10s %d\n", "", k);
    printf("Обчислюємо √%.*Lg ...\n", decimal_places, x);
  }

  int i = 0;

  long double y = (x >= 0) ? 1.0L : -1.0L;
  long double d = 0L;

  do {
    if (k > 0) {
      d = (x / powl(y, k - 1) - y) / k;
    } else {
      d = (1.0L / x / powl(y, -k - 1) - y) / -k;
    }
    y += d;

    if (!is_quiet_mode) {
      i++;
      printf("Ітерація %-*u   Δ = %-*.*Lf   y_%u = %-*.*Lf\n", MAX_CHARS_K, i,
             MAX_CHARS_X, decimal_places, d, i, MAX_CHARS_X, decimal_places, y)
                 ;
    }
    if (i > iteration_limit) {
      return INFINITY;
    }
  } while (fabsl(d) >= e_target && !isinf(d) && !isnan(d));

  if (is_return_delta) {
```

```
    return d;
  }

  return y;
}
```

../meson.build

```
project('op-lab-3', 'c')

# Compiler flags and include directories
c_args = ['-I.', '-Iunity/src', '-Isrc', '-g', '-lm']
inc_dirs = include_directories('.', 'unity/src', 'src')

# Main program sources
main_sources = [
  'src/input/input.c',
  'src/root/root.c',
  'src/main.c'
]

# Test program sources
test_sources = [
  'test/input/check_input_test.c',
  'test/input/custom_input_test.c',
  'test/root/root_test.c',
  'test/test.c',
  'unity/src/unity.c',
  'src/input/input.c',
  'src/root/root.c',
]

# Define the main executable
executable('main.out', main_sources,
  c_args: c_args,
  include_directories: inc_dirs,
  link_args: ['-lm']
)

# Define the test executable
executable('test.out', test_sources,
  c_args: c_args,
  include_directories: inc_dirs,
  link_args: ['-lm']
)
```

Введені та одержані результати

```
Running test 1
Вас вітає Find The KeX — програма для знаходження
кореню k-го показника з x з точністю e
Наші обмеження такі:

1e-15 ≤ x ≤ 1e+15 або -1e+15 ≤ x ≤ -1e-15. Довжина x в символах ≤ 24.
-1000000 ≤ k ≤ 1000000. Довжина k в символах ≤ 7.

Можна вводити щось одне: кількість знаків після коми D або точність e.
1 ≤ D ≤ 15
1e-15 < e < 1
Введіть підкореневий вираз x (від -1e+15 до 1e+15): abc

ПОМИЛКА! x має бути числом і не містити додаткових символів!
Введіть підкореневий вираз x (від -1e+15 до 1e+15): 1e20

ПОМИЛКА! x має бути менший-рівний 1e+15.
Введіть підкореневий вираз x (від -1e+15 до 1e+15): 1e-20

ПОМИЛКА! x має бути більший-рівний 1e-15.
Введіть підкореневий вираз x (від -1e+15 до 1e+15): 17.8
Введіть показник кореня k (від -1000000 до 1000000): 14
За замовчуванням тип вводу точності - через точність e.
Якщо ви бажаєте обрати ввід через кількість знаків після коми D, введіть 0
Введіть точність e: 1e-15
          14
Обчислюємо √17.8 ...
Ітерація 1         Δ = 1.200000000000000          y_1 = 2.200000000000000
Ітерація 2         Δ = -0.157097900164278         y_2 = 2.042902099835722
Ітерація 3         Δ = -0.145803797961997         y_3 = 1.897098301873725
Ітерація 4         Δ = -0.135198614646896         y_4 = 1.761899687226829
Ітерація 5         Δ = -0.125043603417235         y_5 = 1.636856083809594
Ітерація 6         Δ = -0.114818600120066         y_6 = 1.522037483689527
Ітерація 7         Δ = -0.103312376998746         y_7 = 1.418725106690781
Ітерація 8         Δ = -0.087859842542129         y_8 = 1.330865264148652
Ітерація 9         Δ = -0.064119775059567         y_9 = 1.266745489089085
Ітерація 10        Δ = -0.031688200389591         y_10 = 1.235057288699494
Ітерація 11        Δ = -0.006492493653537         y_11 = 1.228564795045956
Ітерація 12        Δ = -0.000232654139487         y_12 = 1.228332140906470
Ітерація 13        Δ = -0.000000286864753         y_13 = 1.228331854041716
Ітерація 14        Δ = -0.000000000000435         y_14 = 1.228331854041281
Ітерація 15        Δ = 0.000000000000000          y_15 = 1.228331854041281

ПЕРЕМОГА! Кінцевий результат y: 1.228331854041281.
```

```
Running test 2
Вас вітає Find The KeX — програма для знаходження
кореню k-го показника з x з точністю e
Наші обмеження такі:

1e-15 ≤ x ≤ 1e+15 або -1e+15 ≤ x ≤ -1e-15. Довжина x в символах ≤ 24.
-1000000 ≤ k ≤ 1000000. Довжина k в символах ≤ 7.

Можна вводити щось одне: кількість знаків після коми D або точність e.
1 ≤ D ≤ 15
1e-15 < e < 1
Введіть підкореневий вираз x (від -1e+15 до 1e+15): -78
Введіть показник кореня k (від -1000000 до 1000000): 1e7

ПОМИЛКА! k має бути менший-рівний 1000000.
Введіть показник кореня k (від -1000000 до 1000000): -1e7

ПОМИЛКА! k має бути більший-рівний -1000000.
Введіть показник кореня k (від -1000000 до 1000000): abc

ПОМИЛКА! k має бути числом і не містити додаткових символів!

Введіть показник кореня k (від -1000000 до 1000000): -2

ПОМИЛКА! Неможливо порахувати корінь парного показнику з від'ємного числа.
Введіть показник кореня k (від -1000000 до 1000000): -3
За замовчуванням тип вводу точності - через точність e.
Якщо ви бажаєте обрати ввід через кількість знаків після коми D, введіть 0
Введіть точність e: 1e-15
          -3
Обчислюємо √-78 ...
Ітерація 1         Δ = 0.329059829059829          y_1 = -0.670940170940171
Ітерація 2         Δ = 0.214153437915051          y_2 = -0.456786733025120
Ітерація 3         Δ = 0.131780960366077          y_3 = -0.325005772659043
Ітерація 4         Δ = 0.067877482733177          y_4 = -0.257128289925866
Ітерація 5         Δ = 0.021071949496169          y_5 = -0.236056340429697
Ітерація 6         Δ = 0.001992963127577          y_6 = -0.234063377302120
Ітерація 7         Δ = 0.000017065669388          y_7 = -0.234046311632732
Ітерація 8         Δ = 0.000000001244417          y_8 = -0.234046310388315
Ітерація 9         Δ = 0.000000000000000          y_9 = -0.234046310388315

ПЕРЕМОГА! Кінцевий результат y: -0.234046310388315.
```

```
Running test 3
Вас вітає Find The KeX — програма для знаходження
кореню k-го показника з x з точністю e
Наші обмеження такі:

1e-15 ≤ x ≤ 1e+15 або -1e+15 ≤ x ≤ -1e-15. Довжина x в символах ≤ 24.
-1000000 ≤ k ≤ 1000000. Довжина k в символах ≤ 7.

Можна вводити щось одне: кількість знаків після коми D або точність e.
1 ≤ D ≤ 15
1e-15 < e < 1
Введіть підкореневий вираз x (від -1e+15 до 1e+15): 234.786432
Введіть показник кореня k (від -1000000 до 1000000): -123
За замовчуванням тип вводу точності - через точність e.
Якщо ви бажаєте обрати ввід через кількість знаків після коми D, введіть 0
Введіть точність e: 2

ПОМИЛКА! e має бути менший за 1.
Введіть точність e: 1e-20

ПОМИЛКА! e має бути більший-рівний 1e-15.
Введіть точність e: 1e-15
          -123
Обчислюємо √234.786432 ...
Ітерація 1         Δ = -0.008095453740645         y_1 = 0.991904546259355
Ітерація 2         Δ = -0.007970918448360         y_2 = 0.983933627810995
Ітерація 3         Δ = -0.007749659577064         y_3 = 0.976183968233932
Ітерація 4         Δ = -0.007280977162385         y_4 = 0.968902991071547
Ітерація 5         Δ = -0.006243359209996         y_5 = 0.962659631861551
Ітерація 6         Δ = -0.004231118484195         y_6 = 0.958428513377356
Ітерація 7         Δ = -0.001638415130880         y_7 = 0.956790098246476
Ітерація 8         Δ = -0.000196709603394         y_8 = 0.956593388643082
Ітерація 9         Δ = -0.000002508791311         y_9 = 0.956590879851770
Ітерація 10        Δ = -0.000000000401444         y_10 = 0.956590879450327
Ітерація 11        Δ = -0.000000000000000         y_11 = 0.956590879450327

ПЕРЕМОГА! Кінцевий результат y: 0.956590879450327.
```

```
Running test 4
Вас вітає Find The KeX — програма для знаходження
кореню k-го показника з x з точністю e
Наші обмеження такі:

1e-15 ≤ x ≤ 1e+15 або -1e+15 ≤ x ≤ -1e-15. Довжина x в символах ≤ 24.
-1000000 ≤ k ≤ 1000000. Довжина k в символах ≤ 7.

Можна вводити щось одне: кількість знаків після коми D або точність e.
1 ≤ D ≤ 15
1e-15 < e < 1
Введіть підкореневий вираз x (від -1e+15 до 1e+15): 512
Введіть показник кореня k (від -1000000 до 1000000): 9
За замовчуванням тип вводу точності - через точність e.
Якщо ви бажаєте обрати ввід через кількість знаків після коми D, введіть 0
Введіть точність e: 0
Введіть кількість знаків після коми D (від 1 до 15): 5
          9
Обчислюємо √512 ...
Ітерація 1         Δ = 56.77778                   y_1 = 57.77778
Ітерація 2         Δ = -6.41975                   y_2 = 51.35802
Ітерація 3         Δ = -5.70645                   y_3 = 45.65158
Ітерація 4         Δ = -5.07240                   y_4 = 40.57918
Ітерація 5         Δ = -4.50880                   y_5 = 36.07038
Ітерація 6         Δ = -4.00782                   y_6 = 32.06256
Ітерація 7         Δ = -3.56251                   y_7 = 28.50006
Ітерація 8         Δ = -3.16667                   y_8 = 25.33338
Ітерація 9         Δ = -2.81482                   y_9 = 22.51856
Ітерація 10        Δ = -2.50206                   y_10 = 20.01650
Ітерація 11        Δ = -2.22406                   y_11 = 17.79244
Ітерація 12        Δ = -1.97694                   y_12 = 15.81551
Ітерація 13        Δ = -1.75728                   y_13 = 14.05823
Ітерація 14        Δ = -1.56203                   y_14 = 12.49620
Ітерація 15        Δ = -1.38847                   y_15 = 11.10774
Ітерація 16        Δ = -1.23419                   y_16 = 9.87354
Ітерація 17        Δ = -1.09706                   y_17 = 8.77648
Ітерація 18        Δ = -0.97516                   y_18 = 7.80132
Ітерація 19        Δ = -0.86681                   y_19 = 6.93451
Ітерація 20        Δ = -0.77049                   y_20 = 6.16402
Ітерація 21        Δ = -0.68486                   y_21 = 5.47916
Ітерація 22        Δ = -0.60873                   y_22 = 4.87043
Ітерація 23        Δ = -0.54098                   y_23 = 4.32945
Ітерація 24        Δ = -0.48059                   y_24 = 3.84886
Ітерація 25        Δ = -0.42647                   y_25 = 3.42239
Ітерація 26        Δ = -0.37724                   y_26 = 3.04515
Ітерація 27        Δ = -0.33066                   y_27 = 2.71449
Ітерація 28        Δ = -0.28231                   y_28 = 2.43218
Ітерація 29        Δ = -0.22378                   y_29 = 2.20840
Ітерація 30        Δ = -0.14482                   y_30 = 2.06358
Ітерація 31        Δ = -0.05628                   y_31 = 2.00730
Ітерація 32        Δ = -0.00719                   y_32 = 2.00011
Ітерація 33        Δ = -0.00011                   y_33 = 2.00000
Ітерація 34        Δ = -0.00000                   y_34 = 2.00000

ПЕРЕМОГА! Кінцевий результат y: 2.00000.
```

Теоретичні розрахунки

№	x	k	Приблизне значення	Точне значення	Похибка
1	17.8	14	1.228331854041281	1.228331854041281	0.000000000000000
2	-73	-3	-0.239272275595000	-0.239272275594637	0.000000000000363
3	234.786432	-123	0.956590879450327	0.956590879450327	0.000000000000000
4	512	9	2.000000000000000	2.000000000000000	0.000000000000000

Висновки: Теоретичні розрахунки відповідають отриманим. Точність достатньо збережена. Програма працює коректно. Програма вирішує поставлене завдання.