ĐẠI HỌC ĐÀ NẴNG
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA CÔNG NGHỆ THÔNG TIN

BÀI TẬP KỸ THUẬT LẬP TRÌNH

SINH VIÊN THỰC HIỆN:

Tên: Nguyễn Văn Mạnh	**Lớp: 20T1**	**Nhóm: 2**
Tên: Ngô Văn Toàn	**Lớp: 20TCLC_DT1**	**Nhóm: 2**
Tên: Đinh Văn Quốc Đạt	**Lớp: 25T_DT1**	**Nhóm: 2**
Tên: Phan Tấn Đạt	**Lớp: 25T_DT1**	**Nhóm: 2**

GIẢNG VIÊN HƯỚNG DẪN: ThS. Nguyễn Văn Nguyên

Đà Nẵng, 12/2025

MỤC LỤC

MỤC LỤC HÌNH ẢNH

1. Bài 1

1.1. Tên đề bài

Viết chương trình nhập vào độ dài 3 cạnh của một tam giác, kiểm tra xem đó có phải là 3 cạnh của một tam giác không ? Cho biết đó là tam giác gì ?

1.2. Thuật toán

1.2.1. Phát biểu thuật toán

- Nhập 3 cạnh a, b, c (Với điều kiện a, b, c lớn hơn 0)
- Kiểm tra điều kiện tồn tại tam giác: tổng hai cạnh bất kỳ phải lớn hơn cạnh còn lại. Nếu không thỏa, kết luận không phải tam giác.
- Phân loại tam giác theo thứ tự:
    - Nếu cả ba cạnh bằng nhau là tam giác đều
    - Nếu có hai cạnh bằng nhau và thỏa định lý Pytago là Tam giác vuông cân
    - Nếu có hai cạnh bằng nhau là Tam giác cân
    - Nếu thỏa định lý Pytago ($a^2 + b^2 = c^2$) là Tam giác vuông
    - Các trường hợp còn lại là Tam giác thường

1.2.2. Mã giả (pseudocode C)

```
FUNCTION nhap_canh (STRING ten) RETURN DOUBLE
  DECLARE DOUBLE x
  WHILE (1)
    OUTPUT ("Nhap canh ", ten, ": ")
    INPUT (x)
    IF (x > 0) THEN
      RETURN x
    ENDIF
    OUTPUT ("Gia tri khong hop le, phai > 0. Nhap lai.")
  ENDWHILE
ENDFUNCTION

FUNCTION la_tam_giac (DOUBLE a, DOUBLE b, DOUBLE c) RETURN INTEGER
  IF (a + b > c && a + c > b && b + c > a) THEN
    RETURN 1
  ENDIF
  RETURN 0
ENDFUNCTION

FUNCTION loai_tam_giac (DOUBLE a, DOUBLE b, DOUBLE c)
```

```
        IF (ABS(a - b) < EPS && ABS(b - c) < EPS) THEN
            OUTPUT ("Tam giac deu")
        ELSE IF (ABS(a - b) < EPS || ABS(a - c) < EPS || ABS(b - c) < EPS) THEN
            DECLARE DOUBLE a2, b2, c2
            a2 = a * a
            b2 = b * b
            c2 = c * c
            IF (ABS(a2 + b2 - c2) < EPS || ABS(a2 + c2 - b2) < EPS || ABS(b2 + c2 -
a2) < EPS) THEN
                OUTPUT ("Tam giac vuong can")
            ELSE
                OUTPUT ("Tam giac can")
            ENDIF
        ELSE
            DECLARE DOUBLE a2, b2, c2
            a2 = a * a
            b2 = b * b
            c2 = c * c
            IF (ABS(a2 + b2 - c2) < EPS || ABS(a2 + c2 - b2) < EPS || ABS(b2 + c2 -
a2) < EPS) THEN
                OUTPUT ("Tam giac vuong")
            ELSE
                OUTPUT ("Tam giac thuong")
            ENDIF
        ENDIF
    ENDFUNCTION

    FUNCTION nhap_3_canh (DOUBLE *a, DOUBLE *b, DOUBLE *c)
        *a = nhap_canh("a")
        *b = nhap_canh("b")
        *c = nhap_canh("c")
    ENDFUNCTION

    FUNCTION main ()
        DECLARE DOUBLE a, b, c
        nhap_3_canh(&a, &b, &c)
        IF (la_tam_giac(a, b, c) == 0) THEN
```

```
        OUTPUT ("Khong phai 3 canh cua tam giac")
        RETURN 0
    ENDIF
    OUTPUT ("Day la 3 canh cua tam giac")
    loai_tam_giac(a, b, c)
    RETURN 0
ENDFUNCTION
```

1.3. Chương trình nguồn:

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

#define EPS 1e-9

double nhap_canh(const char *ten) {
    double x;
    while (1) {
        printf("Nhap canh %s: ", ten);
        if (scanf("%lf", &x) == 1 && x > 0)
            return x;
        printf("Gia tri khong hop le, phai > 0. Nhap lai.\n");
        while (getchar() != '\n');
    }
}

int la_tam_giac(double a, double b, double c) {
    if (a + b > c && a + c > b && b + c > a) {
        return 1;
    }
    return 0;
}

void loai_tam_giac(double a, double b, double c) {
    if (fabs(a - b) < EPS && fabs(b - c) < EPS) {
        printf("Tam giac deu\n");
    } else if (fabs(a - b) < EPS || fabs(a - c) < EPS || fabs(b - c) < EPS) {
        double a2 = a * a, b2 = b * b, c2 = c * c;
```

```
            if (fabs(a2 + b2 - c2) < EPS || fabs(a2 + c2 - b2) < EPS || fabs(b2 + c2 - a2)
< EPS) {
                printf("Tam giac vuong can\n");
            } else {
                printf("Tam giac can\n");
            }
        } else {
            double a2 = a * a, b2 = b * b, c2 = c * c;
            if (fabs(a2 + b2 - c2) < EPS || fabs(a2 + c2 - b2) < EPS || fabs(b2 + c2 - a2)
< EPS) {
                printf("Tam giac vuong\n");
            } else {
                printf("Tam giac thuong\n");
            }
        }
    }

    void nhap_3_canh(double *a, double *b, double *c) {
        *a = nhap_canh("a");
        *b = nhap_canh("b");
        *c = nhap_canh("c");
    }

    int main(void) {
        double a, b, c;
        nhap_3_canh(&a, &b, &c);
        if (!la_tam_giac(a, b, c)) {
            printf("Khong phai 3 canh cua tam giac\n");
            return 0;
        }
        printf("Day la 3 canh cua tam giac\n");
        loai_tam_giac(a, b, c);
        return 0;
    }
```

1.4. Kết quả chương trình

```
Nhap canh a: 5
Nhap canh b: 5
Nhap canh c: 5
Day la 3 canh cua tam giac
Tam giac deu
```

Hình 1.1: Kết quả thực thi thành công với tam giác đều

```
Nhap canh a: 4
Nhap canh b: 4
Nhap canh c: 5
Day la 3 canh cua tam giac
Tam giac can
```

Hình 1.2: Kết quả thực thi với tam giác cân

```
Nhap canh a: 3
Nhap canh b: 4
Nhap canh c: 5
Day la 3 canh cua tam giac
Tam giac vuong
```

Hình 1.3: Kết quả thực thi với tam giác vuông

```
Nhap canh a: 4
Nhap canh b: 5
Nhap canh c: 6
Day la 3 canh cua tam giac
Tam giac thuong
```

Hình 1.4: Kết quả thực thi thành công

```
Nhap canh a: 1
Nhap canh b: 6
Nhap canh c: 7
Khong phai 3 canh cua tam giac
```

Hình 1.5: Kết quả thực thi với không phải là tam giác

1.5. Nhận xét

Thuật toán sử dụng hằng số epsilon ($\varepsilon = 10^{-9}$) để so sánh số thực, đảm bảo độ chính xác khi xử lý kiểu dữ liệu floating-point. Thứ tự phân loại tam giác được thiết kế hợp lý, ưu tiên kiểm tra tam giác đều trước để tránh nhầm lẫn với các trường hợp đặc biệt như tam giác cân, tam giác đều, tam giác vuông và tam giác vuông cân. Cơ chế validate dữ liệu đầu vào giúp chương trình hoạt động ổn định.

2. Bài 5

2.1. Tên đề bài

Cho các số tự nhiên m, n1,n2,..,nm (m ≥2). Hãy chương trình có sử dụng hàm tìm USCLN(n1,n2, …,nm) bằng cách sử dụng hệ thức: USCLN(n1, n2,…,nm) =USCLN(USCLN(n1,n2,…,nm-1),nm)

2.2. Thuật toán

2.2.1. Phát biểu thuật toán

- Kiểm tra trường hợp đặc biệt: nếu tất cả các số đều bằng 0 thì ƯSCLN không xác định.

- Tính ƯSCLN của hai số bằng thuật toán Euclid:
    - Lặp: chia a cho b, lấy phần dư
    - Gán: a = b, b = phần dư
    - Dừng khi b = 0, kết quả là a
- Tính ƯSCLN của nhiều số bằng cách áp dụng tính chất:
    - ƯSCLN(a, b, c) = ƯSCLN(ƯSCLN(a, b), c)
    - Lần lượt tính ƯSCLN với từng phần tử trong dãy

2.2.2. Mã giả (pseudocode C)

```
FUNCTION nhap_chuoi (STRING ten, STRING out)
  OUTPUT ("Nhap ", ten, ": ")
  INPUT (out)
ENDFUNCTION

FUNCTION la_so_tu_nhien (STRING s) RETURN INTEGER
  IF (s[0] == NULL) THEN RETURN 0 ENDIF
  FOR (i = 0; i < LENGTH(s); i++)
    IF (s[i] IS NOT DIGIT) THEN RETURN 0 ENDIF
  ENDFOR
  RETURN 1
ENDFUNCTION

FUNCTION to_ll (STRING s) RETURN LONG_LONG
  DECLARE LONG_LONG x = 0
  FOR (i = 0; i < LENGTH(s); i++)
    x = x * 10 + (s[i] - '0')
  ENDFOR
  RETURN x
ENDFUNCTION

FUNCTION uscln2 (LONG_LONG a, LONG_LONG b) RETURN LONG_LONG
  WHILE (b != 0)
    t = a % b
    a = b
    b = t
  ENDWHILE
  RETURN a
```

```
ENDFUNCTION

FUNCTION uscln_nhieu (ARRAY arr[], INTEGER n) RETURN LONG_LONG
   DECLARE LONG_LONG g = arr[0]
   FOR (i = 1; i < n; i++)
      g = uscln2(g, arr[i])
   ENDFOR
   RETURN g
ENDFUNCTION

FUNCTION main ()
   WHILE (1)
      nhap_chuoi("so luong phan tu m", buf)
      IF (la_so_tu_nhien(buf)) THEN
         m = to_ll(buf)
         IF (m >= 2) THEN BREAK ENDIF
      ENDIF
   ENDWHILE
   FOR (i = 0; i < m; i++)
      nhap_chuoi(ten, A[i])
   ENDFOR
   dem = 0
   FOR (i = 0; i < m; i++)
      IF (la_so_tu_nhien(A[i])) THEN
         hop_le[dem] = to_ll(A[i])
         dem = dem + 1
      ENDIF
   ENDFOR
   ketqua = uscln_nhieu(hop_le, dem)
   OUTPUT ("USCLN cua cac so tu nhien hop le = ", ketqua)
   RETURN 0
ENDFUNCTION
```

2.3. Chương trình nguồn:

```
#include <stdio.h>
#include <ctype.h>
#include <string.h>
```

```
void nhap_chuoi(const char *ten, char *out) {
    printf("Nhap %s: ", ten);
    scanf("%s", out);
}

int la_so_tu_nhien(const char *s) {
    int i = 0;
    if (s[0] == '\0') return 0;
    for (i = 0; s[i]; i++)
        if (!isdigit(s[i]))
            return 0;
    return 1;
}

long long to_ll(const char *s) {
    long long x = 0;
    int i;
    for (i = 0; s[i]; i++)
        x = x * 10 + (s[i] - '0');
    return x;
}

long long uscln2(long long a, long long b) {
    long long t;
    while (b != 0) {
        t = a % b;
        a = b;
        b = t;
    }
    return a;
}

long long uscln_nhieu(long long arr[], int n) {
    int i;
    long long g = arr[0];
    for (i = 1; i < n; i++)
        g = uscln2(g, arr[i]);
```

```
    return g;
}

int main() {
    char buf[100];
    int m = 0;
    while (1) {
        nhap_chuoi("so luong phan tu m", buf);
        if (la_so_tu_nhien(buf)) {
            m = (int)to_ll(buf);
            if (m >= 2) break;
        }
        printf("m phai la so tu nhien >= 2. Nhap lai.\n");
    }
    char A[200][100];
    printf("\n=== Nhap %d gia tri bat ky ===\n", m);
    int i;
    for (i = 0; i < m; i++) {
        char ten[50];
        sprintf(ten, "n%d", i + 1);
        nhap_chuoi(ten, A[i]);
    }
    long long hop_le[200];
    int dem = 0;
    for (i = 0; i < m; i++) {
        if (la_so_tu_nhien(A[i])) {
            hop_le[dem++] = to_ll(A[i]);
        }
    }
    if (dem == 0) {
        printf("\nKhong co so tu nhien hop le nao.\n");
        return 0;
    }
    int tat_ca_bang_0 = 1;
    for (i = 0; i < dem; i++) {
        if (hop_le[i] != 0) {
            tat_ca_bang_0 = 0;
```

```
            break;
        }
    }
    if (tat_ca_bang_0) {
        printf("\nTat ca so tu nhien = 0, USCLN khong xac dinh\n");
        return 0;
    }
    long long ketqua = uscln_nhieu(hop_le, dem);
    printf("\nUSCLN cua cac so tu nhien hop le = %lld\n", ketqua);
    return 0;
}
```

2.4. Kết quả chương trình

```
Nhap so luong phan tu m: 2

=== Nhap 2 gia tri bat ky ===
Nhap n1: 12
Nhap n2: 18

USCLN cua cac so tu nhien hop le = 6
```

Hình 2.1: Kết quả thực thi thành công với m bằng 2

```
Nhap so luong phan tu m: 4

=== Nhap 4 gia tri bat ky ===
Nhap n1: 7
Nhap n2: 13
Nhap n3: 19
Nhap n4: 29

USCLN cua cac so tu nhien hop le = 1
```

Hình 2.2: Kết quả thực thi với m lớn hơn 2

2.5. Nhận xét

Sử dụng thuật toán Euclid để tính ƯSCLN của 2 số, sau đó mở rộng cho nhiều số: USCLN(a,b,c) = USCLN(USCLN(a,b),c). Chương trình có kiểm tra đầu vào bằng chuỗi để lọc số tự nhiên hợp lệ.

3. Bài 9

3.1. Tên đề tài

Hãy viết chương trình có sử dụng hàm để tính giá trị của biểu thức

$$1 - \frac{1}{2} - \frac{1}{3} + \cdots + \frac{1}{999} - \frac{1}{1000}$$ theo các cách sau đây:

a. Tính từ trái sang phải

b. Tính từ phải sang trái

c. Tính tổng $S_1 = 1 + \frac{1}{3} + \cdots + \frac{1}{999}$ và $S_2 = 1 + \frac{1}{2} + \frac{1}{4} + \ldots + \frac{1}{1000}$

d. So sánh kết quả câu b và c. Giải thích tại sao kết quả khác nhau?

3.2. Thuật toán

3.2.1. Phát biểu thuật toán

- Câu a: Duyệt i từ 1 đến n, cộng 1/i nếu i lẻ, trừ 1/i nếu i chẵn. Cách này cộng số lớn trước nên sai số tích lũy nhiều hơn.
- Câu b: Duyệt i từ n về 1, thực hiện tương tự. Cách này cộng số nhỏ trước nên sai số ít hơn.
- Câu c: Tính riêng tổng các số hạng lẻ (S1) và tổng các số hạng chẵn (S2), sau đó trả về S1 - S2.

3.2.2. Mã giả (pseudocode C)

```
FUNCTION tinh_trai_sang_phai (INTEGER n) RETURN DOUBLE
   DECLARE DOUBLE tong = 0.0
   FOR (i = 1; i <= n; i++)
      IF (i % 2 == 0) THEN
         tong = tong - 1.0 / i
      ELSE
         tong = tong + 1.0 / i
      ENDIF
   ENDFOR
   RETURN tong
ENDFUNCTION

FUNCTION tinh_phai_sang_trai (INTEGER n) RETURN DOUBLE
   DECLARE DOUBLE tong = 0.0
   FOR (i = n; i >= 1; i--)
      IF (i % 2 == 0) THEN
         tong = tong - 1.0 / i
      ELSE
         tong = tong + 1.0 / i
      ENDIF
   ENDFOR
   RETURN tong
ENDFUNCTION

FUNCTION tinh_tach_s1_s2 (INTEGER n) RETURN DOUBLE
   DECLARE DOUBLE s1 = 0.0, s2 = 0.0
   FOR (i = 1; i <= n; i++)
      IF (i % 2 == 0) THEN
```

```
            s2 = s2 + 1.0 / i
        ELSE
            s1 = s1 + 1.0 / i
        ENDIF
    ENDFOR
    RETURN s1 - s2
ENDFUNCTION

FUNCTION xu_ly ()
    DECLARE INTEGER n = 10000
    DECLARE DOUBLE kq_a, kq_b, kq_c
    kq_a = tinh_trai_sang_phai(n)
    kq_b = tinh_phai_sang_trai(n)
    kq_c = tinh_tach_s1_s2(n)
    OUTPUT (kq_a, kq_b, kq_c)
    IF (kq_b == kq_c) THEN
        OUTPUT ("Ket qua bang nhau")
    ELSE
        OUTPUT ("Ket qua khac nhau do sai so floating point")
    ENDIF
ENDFUNCTION

FUNCTION main ()
    xu_ly()
    RETURN 0
ENDFUNCTION
```

3.3. Chương trình nguồn

```
#include <stdio.h>

double tinh_trai_sang_phai(int n) {
    int i;
    double tong = 0.0;
    for (i = 1; i <= n; i++) {
        if (i % 2 == 0)
            tong -= 1.0 / i;
        else
            tong += 1.0 / i;
```

```
    }
    return tong;
}

double tinh_phai_sang_trai(int n) {
    int i;
    double tong = 0.0;
    for (i = n; i >= 1; i--) {
        if (i % 2 == 0)
            tong -= 1.0 / i;
        else
            tong += 1.0 / i;
    }
    return tong;
}

double tinh_tach_s1_s2(int n) {
    int i;
    double s1 = 0.0;
    double s2 = 0.0;
    for (i = 1; i <= n; i++) {
        if (i % 2 == 0)
            s2 += 1.0 / i;
        else
            s1 += 1.0 / i;
    }
    return s1 - s2;
}

void xu_ly() {
    int n = 10000;
    double kq_a = tinh_trai_sang_phai(n);
    printf("a. Tong tinh tu trai sang phai: %.15lf\n", kq_a);
    double kq_b = tinh_phai_sang_trai(n);
    printf("b. Tong tinh tu phai sang trai: %.15lf\n", kq_b);
    double kq_c = tinh_tach_s1_s2(n);
    printf("c. Tong s1 - s2: %.15lf\n", kq_c);
```

```
    if (kq_b == kq_c)
        printf("d. Ket qua b va c bang nhau\n");
    else
        printf("d. Ket qua b va c khac nhau\n");
    printf("   Ly do khac nhau: do sai so cham phan (floating point) khi cong tru cac so nho theo thu tu khac nhau.\n");
}

int main() {
    xu_ly();
    return 0;
}
```

3.4. Kết quả

```
a. Tong tinh tu trai sang phai: 0.693097183059958
b. Tong tinh tu phai sang trai: 0.693097183059945
c. Tong s1 - s2: 0.693097183059954
d. Ket qua b va c khac nhau
   Ly do khac nhau: do sai so cham phan (floating point) khi cong tru cac so nho th
eo thu tu khac nhau.
```

Hình 3.1: Kết quả thực thi thành công

3.5. Nhận xét

Bài toán minh họa sai số cộng dồn trong số thực floating point. Thứ tự cộng các số nhỏ ảnh hưởng đến kết quả cuối cùng. Cách tính từ phải sang trái (cộng số nhỏ trước) thường chính xác hơn

4. Bài 14

4.1. Tên đề tài

Cho một tam giác có 3 cạnh là a,b,c. Hãy tính

a. Độ dài các chiều cao của tam giác

b. Độ dài 3 đường trung tuyến

c. Độ dài 3 đường phân giác

d. Bán kính đường tròn nội tiếp và ngoại tiếp.

Viết chương trình sử dụng hàm để giải bài toán này.

4.2. Thuật Toán

4.2.1. Phát biểu bài toán

- Nhập 3 cạnh a, b, c (kiểm tra > 0)
- Kiểm tra điều kiện tam giác a + b > c và a + c > b và b + c > a
- Tính diện tích bằng công thức Heron: $S = \sqrt{p(p-a)(p-b)(p-c)}$ với p là nửa chu vi. Từ diện tích suy ra các yếu tố khác.
- Các công thức tính toán:

- Chiều cao: $h_a = 2.S/a$
- Đường trung tuyến: $m_a = \sqrt{2b^2 + 2c^2 - a^2}/2$
- Đường phân giác: $l_a = \sqrt{bc(1 - a^2/(b + c)^2)}$
- Bán kính nội tiếp: $r = S/p$, ngoại tiếp: $R = abc/(4S)$

4.2.2. Mã giả (pseudocode C)

```
FUNCTION nhap_canh (STRING ten) RETURN DOUBLE
  DECLARE DOUBLE x
  WHILE (1)
    OUTPUT ("Nhap canh ", ten, ": ")
    INPUT (x)
    IF (x > 0) RETURN x
    OUTPUT ("Gia tri khong hop le, phai > 0. Nhap lai.")
  ENDWHILE
ENDFUNCTION

FUNCTION la_tam_giac (DOUBLE a, DOUBLE b, DOUBLE c) RETURN INTEGER
  RETURN (a + b > c) && (a + c > b) && (b + c > a)
ENDFUNCTION

FUNCTION dien_tich (DOUBLE a, DOUBLE b, DOUBLE c) RETURN DOUBLE
  DECLARE DOUBLE p = (a + b + c) / 2.0
  DECLARE DOUBLE S2 = p * (p - a) * (p - b) * (p - c)
  IF (S2 <= EPSILON) RETURN 0
  RETURN SQRT(S2)
ENDFUNCTION

FUNCTION tinh_chieu_cao (DOUBLE a, DOUBLE b, DOUBLE c)
  DECLARE DOUBLE S = dien_tich(a, b, c)
  IF (S == 0) RETURN
  OUTPUT ("Chieu cao ha = ", 2*S/a)
  OUTPUT ("Chieu cao hb = ", 2*S/b)
  OUTPUT ("Chieu cao hc = ", 2*S/c)
ENDFUNCTION
```

```
FUNCTION tinh_trung_tuyen (DOUBLE a, DOUBLE b, DOUBLE c)
   OUTPUT ("Trung tuyen ma = ", SQRT(2*b*b + 2*c*c - a*a)/2)
   OUTPUT ("Trung tuyen mb = ", SQRT(2*a*a + 2*c*c - b*b)/2)
   OUTPUT ("Trung tuyen mc = ", SQRT(2*a*a + 2*b*b - c*c)/2)
ENDFUNCTION

FUNCTION tinh_phan_giac (DOUBLE a, DOUBLE b, DOUBLE c)
   OUTPUT ("Phan giac la = ", SQRT(b*c * (1 - a*a/((b+c)*(b+c)))))
   OUTPUT ("Phan giac lb = ", SQRT(a*c * (1 - b*b/((a+c)*(a+c)))))
   OUTPUT ("Phan giac lc = ", SQRT(a*b * (1 - c*c/((a+b)*(a+b)))))
ENDFUNCTION

FUNCTION tinh_ban_kinh (DOUBLE a, DOUBLE b, DOUBLE c)
   DECLARE DOUBLE S = dien_tich(a, b, c)
   DECLARE DOUBLE p = (a + b + c) / 2.0
   IF (S == 0) RETURN
   OUTPUT ("Ban kinh noi tiep r = ", S/p)
   OUTPUT ("Ban kinh ngoai tiep R = ", a*b*c/(4*S))
ENDFUNCTION

FUNCTION xu_ly ()
   DECLARE DOUBLE a, b, c
   a = nhap_canh("a")
   b = nhap_canh("b")
   c = nhap_canh("c")
   IF (!la_tam_giac(a, b, c))
      OUTPUT ("Ba canh khong tao thanh tam giac.")
      RETURN
   ENDIF
   tinh_chieu_cao(a, b, c)
   tinh_trung_tuyen(a, b, c)
   tinh_phan_giac(a, b, c)
   tinh_ban_kinh(a, b, c)
ENDFUNCTION

FUNCTION main ()
   xu_ly()
```

```
    RETURN 0
ENDFUNCTION
```

4.3. Chương trình nguồn

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

#define EPSILON 1e-12

double nhap_canh(const char *ten) {
    double x;
    while (1) {
        printf("Nhap canh %s: ", ten);
        if (scanf("%lf", &x) == 1 && x > 0)
            return x;
        printf("Gia tri khong hop le, phai > 0. Nhap lai.\n");
        while (getchar() != '\n');
    }
}

int la_tam_giac(double a, double b, double c) {
    return (a + b > c) && (a + c > b) && (b + c > a);
}

double dien_tich(double a, double b, double c) {
    double p = (a + b + c) / 2.0;
    double S2 = p * (p - a) * (p - b) * (p - c);
    if (S2 <= EPSILON) return 0;
    return sqrt(S2);
}

void tinh_chieu_cao(double a, double b, double c) {
    double S = dien_tich(a, b, c);
    if (S == 0) {
        printf("Tam giac suy bien → khong co chieu cao.\n");
        return;
    }
    printf("Chieu cao ha = %.6lf\n", 2*S/a);
```

```
    printf("Chieu cao hb = %.6lf\n", 2*S/b);
    printf("Chieu cao hc = %.6lf\n", 2*S/c);
}

void tinh_trung_tuyen(double a, double b, double c) {
    printf("Trung tuyen ma = %.6lf\n", sqrt(2*b*b + 2*c*c - a*a)/2);
    printf("Trung tuyen mb = %.6lf\n", sqrt(2*a*a + 2*c*c - b*b)/2);
    printf("Trung tuyen mc = %.6lf\n", sqrt(2*a*a + 2*b*b - c*c)/2);
}

void tinh_phan_giac(double a, double b, double c) {
    printf("Phan giac la = %.6lf\n", sqrt(b*c * (1 - a*a/((b+c)*(b+c)))));
    printf("Phan giac lb = %.6lf\n", sqrt(a*c * (1 - b*b/((a+c)*(a+c)))));
    printf("Phan giac lc = %.6lf\n", sqrt(a*b * (1 - c*c/((a+b)*(a+b)))));
}

void tinh_ban_kinh(double a, double b, double c) {
    double S = dien_tich(a, b, c);
    double p = (a + b + c)/2.0;
    if (S == 0) {
        printf("Tam giac suy bien → khong ton tai ban kinh noi/ngoai tiep.\n");
        return;
    }
    printf("Ban kinh noi tiep r = %.6lf\n", S/p);
    printf("Ban kinh ngoai tiep R = %.6lf\n", a*b*c/(4*S));
}

void xu_ly() {
    printf("Nhap ba canh cua tam giac\n");
    double a = nhap_canh("a");
    double b = nhap_canh("b");
    double c = nhap_canh("c");
    if (!la_tam_giac(a, b, c)) {
        printf("Ba canh khong tao thanh tam giac.\n");
        return;
    }
    if (fabs(a + b - c) < EPSILON ||
```

```
        fabs(a + c - b) < EPSILON ||
        fabs(b + c - a) < EPSILON) {
        printf("Day la tam giac SUY BIEN.\n");
        return;
    }
    printf("\n=== Ket qua tinh toan ===\n");
    tinh_chieu_cao(a, b, c);
    tinh_trung_tuyen(a, b, c);
    tinh_phan_giac(a, b, c);
    tinh_ban_kinh(a, b, c);
}

int main() {
    xu_ly();
    return 0;
}
```

4.4. Kết quả

```
Nhap ba canh cua tam giac
Nhap canh a: 6
Nhap canh b: 6
Nhap canh c: 6

=== Ket qua tinh toan ===
Chieu cao ha = 5.196152
Chieu cao hb = 5.196152
Chieu cao hc = 5.196152
Trung tuyen ma = 5.196152
Trung tuyen mb = 5.196152
Trung tuyen mc = 5.196152
Phan giac la = 5.196152
Phan giac lb = 5.196152
Phan giac lc = 5.196152
Ban kinh noi tiep r = 1.732051
Ban kinh ngoai tiep R = 3.464102
```

Hình 4.1: Kết quả thực thi với tam giác đều

```
Nhap ba canh cua tam giac
Nhap canh a: 5
Nhap canh b: 5
Nhap canh c: 8

=== Ket qua tinh toan ===
Chieu cao ha = 4.800000
Chieu cao hb = 4.800000
Chieu cao hc = 3.000000
Trung tuyen ma = 6.184658
Trung tuyen mb = 6.184658
Trung tuyen mc = 3.000000
Phan giac la = 5.838051
Phan giac lb = 5.838051
Phan giac lc = 3.000000
Ban kinh noi tiep r = 1.333333
Ban kinh ngoai tiep R = 4.166667
```

Hình 4.2: Kết quả thực thi với tam giác cân

```
Nhap ba canh cua tam giac
Nhap canh a: 3
Nhap canh b: 4
Nhap canh c: 5

=== Ket qua tinh toan ===
Chieu cao ha = 4.000000
Chieu cao hb = 3.000000
Chieu cao hc = 2.400000
Trung tuyen ma = 4.272002
Trung tuyen mb = 3.605551
Trung tuyen mc = 2.500000
Phan giac la = 4.216370
Phan giac lb = 3.354102
Phan giac lc = 2.424366
Ban kinh noi tiep r = 1.000000
Ban kinh ngoai tiep R = 2.500000
```

Hình 4.3: Kết quả thực thi với tam giác vuông

```
Nhap ba canh cua tam giac
Nhap canh a: 7
Nhap canh b: 10
Nhap canh c: 12

=== Ket qua tinh toan ===
Chieu cao ha = 9.993620
Chieu cao hb = 6.995534
Chieu cao hc = 5.829612
Trung tuyen ma = 10.476163
Trung tuyen mb = 8.455767
Trung tuyen mc = 6.204837
Phan giac la = 10.385145
Phan giac lb = 7.793029
Phan giac lc = 5.926306
Ban kinh noi tiep r = 2.412253
Ban kinh ngoai tiep R = 6.003830
```

Hình 4.4: Kết quả thực thi thành công

```
Nhap ba canh cua tam giac
Nhap canh a: 1
Nhap canh b: 6
Nhap canh c: 7
Ba canh khong tao thanh tam giac.
```

Hình 4.5: Kết quả thực thi với số đo không phải là tam giác

4.5. Nhận xét

Bài toán vận dụng các công thức hình học cơ bản để tính toán đầy đủ các yếu tố của tam giác từ ba cạnh cho trước. Công thức được sử dụng làm nền tảng để tính diện tích, từ đó suy ra các đại lượng còn lại một cách có hệ thống. Thuật toán đảm bảo tính chính xác thông qua việc xử lý các trường hợp biên như tam giác suy biến.

5. Bài 20

5.1. Tên đề bài

Cho số tự nhiên n (n ≤ 99). Hãy viết chương trình sử dụng hàm kiểm tra n^2 có bằng tổng các lập phương các chữ số của n hay không ?

5.2. Thuật toán

5.2.1. Phát biểu thuật toán

- Nhập n trong [0, 99] (xử lý nhập sai)
- Tách từng chữ số của n bằng phép toán: lấy chữ số cuối digit = n % 10, bỏ chữ số cuối n = $\frac{n}{10}$. Lặp cho đến khi n = 0.
- Tính tổng lập phương: $\sum_{i=1}^{k} d_i^3$ với d_i là các chữ số của n.
- So sánh n^2 với tổng lập phương để đưa ra kết luận.

5.2.2. Mã giả (pseudocode C)

```
FUNCTION nhap_so_tu_nhien () RETURN INTEGER
   DECLARE DOUBLE x
   DECLARE INTEGER n
   WHILE (1)
      OUTPUT ("Nhap n (0 <= n <= 99): ")
      INPUT (x)
      IF (x == FLOOR(x)) THEN
         n = (INTEGER)x
         IF (n >= 0 && n <= 99) THEN
            RETURN n
         ENDIF
      ENDIF
      OUTPUT ("Gia tri khong hop le. Nhap lai!")
   ENDWHILE
ENDFUNCTION
```

```
FUNCTION tong_lap_phuong (INTEGER n) RETURN INTEGER
   DECLARE INTEGER sum = 0
   DECLARE INTEGER digit
   WHILE (n > 0)
      digit = n % 10
      sum = sum + digit * digit * digit
      n = n / 10
   ENDWHILE
   RETURN sum
ENDFUNCTION

FUNCTION kiem_tra (INTEGER n) RETURN INTEGER
   DECLARE INTEGER n2 = n * n
   DECLARE INTEGER tong = tong_lap_phuong(n)
   IF (n2 == tong) THEN RETURN 1 ENDIF
   RETURN 0
ENDFUNCTION

FUNCTION main ()
   DECLARE INTEGER n = nhap_so_tu_nhien()
   IF (kiem_tra(n) == 1) THEN
      OUTPUT ("Dung: n^2 = tong lap phuong cac chu so cua n")
   ELSE
      OUTPUT ("Sai: n^2 khong bang tong lap phuong cac chu so cua n")
   ENDIF
   RETURN 0
ENDFUNCTION
```

5.3. Chương trình nguồn

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

int nhap_so_tu_nhien() {
   double x;
   int n;
   while (1) {
      printf("Nhap n (0 <= n <= 99): ");
```

```
        if (scanf("%lf", &x) == 1) {
            if (x == (int)x) {
                n = (int)x;
                if (n >= 0 && n <= 99)
                    return n;
            }
        }
        printf("Gia tri khong hop le. Nhap lai!\n");
        while (getchar() != '\n');
    }
}

int tong_lap_phuong(int n) {
    int sum = 0;
    int temp = n;
    int digit;
    while (temp > 0) {
        digit = temp % 10;
        sum += digit * digit * digit;
        temp /= 10;
    }
    return sum;
}

int kiem_tra(int n) {
    int n2 = n * n;
    int tong = tong_lap_phuong(n);
    return n2 == tong;
}

int main() {
    int n = nhap_so_tu_nhien();
    if (kiem_tra(n)) {
        printf("Dung: n^2 = tong lap phuong cac chu so cua n\n");
    } else {
        printf("Sai: n^2 khong bang tong lap phuong cac chu so cua n\n");
    }
```

```
    return 0;
}
```

5.4. Kết quả

```
Nhap n (0 <= n <= 99): 0
Dung: n^2 = tong lap phuong cac chu so cua n
```

Hình 5.1: Kết quả thực thi với n bằng 0

```
Nhap n (0 <= n <= 99): 1
Dung: n^2 = tong lap phuong cac chu so cua n
```

Hình 5.2: Kết quả thực thi với n bằng 1

```
Nhap n (0 <= n <= 99): 2
Sai: n^2 khong bang tong lap phuong cac chu so cua n
```

Hình 5.3: Kết quả thực thi thành công với n là số ngẫu nhiên

5.5. Nhận xét

Bài toán kiểm tra tính chất số học, yêu cầu phân tích cấu trúc chữ số của một số tự nhiên. Thuật toán sử dụng kỹ thuật tách chữ số bằng phép chia lấy dư và chia nguyên để tính tổng lập phương. Xử lý nhập liệu kỹ (số thực, số âm, vượt phạm vi). Với n ∈ [0, 99], chỉ có n = 1 thỏa mãn ($1^2 = 1^3 = 1$).

6. Bài 23

6.1. Tên đề bài

Cho số thực a và số tự nhiên n. Hãy viết chương trình có sử dụng hàm để tính:

a. a^n

b. a(a+1)...(a+n-1)

c. $\frac{1}{a} + \frac{1}{a(a+1)} + \cdots + \frac{1}{a(a+1)\ldots(a+n)}$

6.2. Thuật toán

6.2.1. Phát biểu bài toán

- Nhập a (số thực) và n (số tự nhiên nghiêm ngặt)
- Thực hiện tính toán
  - a) Lũy thừa: $a^n = a_1 \times a_2 \times \ldots \times a_n$
  - b) Tích liên tiếp: $P = a \times (a+1) \times (a+2) \times \ldots \times (a+n-1) = \prod_{i=0}^{n-1}(a+i)$
  - c) Tổng phân số: $S = \frac{1}{a} + \frac{1}{a(a+1)} + \frac{1}{a(a+1)(a+2)} + \ldots = \sum_{i=0}^{n} \frac{1}{\prod_{j=0}^{i}(a+j)}$
- In kết quả ra màn hình

6.2.2. Mã giả (pseudocode C)

```
FUNCTION nhap_so_thuc (STRING ten) RETURN DOUBLE
  WHILE (1)
    OUTPUT ("Nhap ", ten, ": ")
```

```
        INPUT (x)
        IF (HOP_LE) RETURN x
      ENDWHILE
    ENDFUNCTION

    FUNCTION nhap_so_tu_nhien (STRING ten) RETURN INTEGER
      WHILE (1)
        OUTPUT ("Nhap ", ten, ": ")
        INPUT (s)
        IF (CHI_GOM_CHU_SO(s))
          n = CHUYEN_SANG_SO(s)
          RETURN n
        ENDIF
        OUTPUT ("Gia tri phai la so tu nhien. Nhap lai.")
      ENDWHILE
    ENDFUNCTION

    FUNCTION tinh_mu (DOUBLE a, INTEGER n) RETURN DOUBLE
      DECLARE DOUBLE kq = 1
      FOR (i = 0; i < n; i++)
        kq = kq * a
      ENDFOR
      RETURN kq
    ENDFUNCTION

    FUNCTION tinh_tich (DOUBLE a, INTEGER n) RETURN DOUBLE
      DECLARE DOUBLE kq = 1
      FOR (i = 0; i < n; i++)
        kq = kq * (a + i)
      ENDFOR
      RETURN kq
    ENDFUNCTION

    FUNCTION tinh_tong (DOUBLE a, INTEGER n, INTEGER *loi) RETURN
DOUBLE
      IF (a == 0)
        *loi = 1
```

```
        RETURN 0
    ENDIF
    *loi = 0
    DECLARE DOUBLE tich = 1, tong = 0
    FOR (i = 0; i <= n; i++)
        IF (i > 0) tich = tich * (a + i - 1)
        IF (tich == 0) *loi = 1; RETURN 0
        tong = tong + 1.0 / tich
    ENDFOR
    RETURN tong
ENDFUNCTION

FUNCTION main ()
    a = nhap_so_thuc("a")
    n = nhap_so_tu_nhien("n")
    OUTPUT ("a^n = ", tinh_mu(a, n))
    OUTPUT ("Tich = ", tinh_tich(a, n))
    tong = tinh_tong(a, n, loi)
    IF (loi) OUTPUT ("Khong the tinh")
    ELSE OUTPUT ("Tong = ", tong)
    RETURN 0
ENDFUNCTION
```

6.3. Chương trình nguồn

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>

double nhap_so_thuc(const char *ten) {
    double x;
    while (1) {
        printf("Nhap %s: ", ten);
        if (scanf("%lf", &x) == 1)
            return x;
        printf("Gia tri khong hop le. Nhap lai.\n");
        while (getchar() != '\n');
    }
}
```

```
int nhap_so_tu_nhien(const char *ten) {
    char s[100];
    int i;
    while (1) {
        int hop_le = 1;
        printf("Nhap %s: ", ten);
        scanf("%s", s);
        for (i = 0; s[i] != '\0'; i++) {
            if (!isdigit(s[i])) {
                hop_le = 0;
                break;
            }
        }
        if (hop_le) {
            int n = 0;
            for (i = 0; s[i] != '\0'; i++)
                n = n * 10 + (s[i] - '0');
            return n;
        }
        printf("Gia tri n phai la so tu nhien (khong dau cham, khong thap phan!). Nhap lai.\n");
    }
}

double tinh_mu(double a, int n) {
    int i;
    double kq = 1;
    for (i = 0; i < n; i++)
        kq *= a;
    return kq;
}

double tinh_tich(double a, int n) {
    int i;
    double kq = 1;
    for (i = 0; i < n; i++)
```

```
        kq *= (a + i);
    return kq;
}

double tinh_tong(double a, int n, int *loi) {
    int i;
    double tich = 1;
    double tong = 0;
    if (a == 0) {
        *loi = 1;
        return 0;
    }
    *loi = 0;
    for (i = 0; i <= n; i++) {
        if (i > 0)
            tich *= (a + i - 1);
        if (tich == 0) {
            *loi = 1;
            return 0;
        }
        tong += 1.0 / tich;
    }
    return tong;
}

int main(void) {
    double a = nhap_so_thuc("a");
    int n = nhap_so_tu_nhien("n");
    printf("\n=== Ket qua ===\n");
    printf("a) a^n = %lf\n", tinh_mu(a, n));
    printf("b) Tich = %lf\n", tinh_tich(a, n));
    int loi;
    double tong = tinh_tong(a, n, &loi);
    if (loi)
        printf("c) Khong the tinh (do chia 0)\n");
    else
        printf("c) Tong = %lf\n", tong);
```

```
    return 0;
}
```

6.4. Kết quả

```
Nhap a: 2
Nhap n: 3

=== Ket qua ===
a) a^n = 8.000000
b) Tich = 24.000000
c) Tong = 1.708333
```

Hình 6.1: Kết quả thực thi thành công

```
Nhap a: 0
Nhap n: 3

=== Ket qua ===
a) a^n = 0.000000
b) Tich = 0.000000
c) Khong the tinh (do chia 0)
```

Hình 6.2: Kết quả thực thi thành công với a bằng 0

```
Nhap a: 1
Nhap n: 0

=== Ket qua ===
a) a^n = 1.000000
b) Tich = 1.000000
c) Tong = 1.000000
```

Hình 6.3: Kết quả thực thi thành công với n bằn 0

6.5. Nhận xét

Bài toán tổng hợp các phép tính cơ bản trên dãy số, bao gồm lũy thừa, tích liên tiếp và tổng phân số. Thuật toán sử dụng kỹ thuật tích lũy để tối ưu việc tính toán, tránh phải tính lại tích ở mỗi bước. Nhập n nghiêm ngặt chỉ chấp nhận chuỗi chữ số (không cho 2.0, 3.0). Việc xử lý trường hợp chia cho 0 được thực hiện cẩn thận để đảm bảo tính ổn định của chương trình.

7. Bài 34

7.1. Tên đề bài

Viết chương trình nhập vào chuỗi số nguyên dương *a* và kiểm tra *a* có phải là chuỗi số tự đối xứng không?

Ví dụ: 5, 232, 5775 tự đối xứng, nhưng 2342 thì không tự đối xứng. 01 là không đối xứng

7.2. Thuật toán

7.2.1. Phát biểu thuật toán

- Đặt hai con trỏ: left ở đầu chuỗi, right ở cuối chuỗi.
- So sánh ký tự tại hai vị trí. Nếu khác nhau, kết luận không đối xứng.

- Di chuyển left sang phải, right sang trái. Lặp buocws treen cho đến khi hai con trỏ gặp nhau.
- Nếu hoàn thành vòng lặp, kết luận chuỗi đối xứng.

7.2.2. Mã giả (pseudocode C)

```
FUNCTION nhap_chuoi (STRING s, INTEGER max_len)
  WHILE (1)
    OUTPUT ("Nhap chuoi so nguyen duong a: ")
    INPUT (s)
    DECLARE INTEGER hop_le = 1
    FOR (i = 0; i < LENGTH(s); i++)
      IF (s[i] < '0' || s[i] > '9') THEN
        hop_le = 0
        BREAK
      ENDIF
    ENDFOR
    IF (hop_le == 1 && s[0] == '0' && LENGTH(s) > 1) THEN
      hop_le = 0
    ENDIF
    IF (hop_le == 1) THEN RETURN ENDIF
    OUTPUT ("Chuoi khong hop le. Nhap lai.")
  ENDWHILE
ENDFUNCTION

FUNCTION la_doi_xung (STRING s) RETURN INTEGER
  DECLARE INTEGER l = 0
  DECLARE INTEGER r = LENGTH(s) - 1
  WHILE (l < r)
    IF (s[l] != s[r]) THEN RETURN 0 ENDIF
    l = l + 1
    r = r - 1
  ENDWHILE
  RETURN 1
ENDFUNCTION

FUNCTION main ()
  DECLARE STRING a[105]
  nhap_chuoi(a, 105)
```

```
    IF (la_doi_xung(a) == 1) THEN
      OUTPUT ("Chuoi tu doi xung")
    ELSE
      OUTPUT ("Chuoi khong tu doi xung")
    ENDIF
    RETURN 0
ENDFUNCTION
```

7.3. Chương trình nguồn

```
#include <stdio.h>
#include <string.h>

void nhap_chuoi(char *s, int max_len) {
    while (1) {
        printf("Nhap chuoi so nguyen duong a: ");
        scanf("%s", s);
        int i;
        int hop_le = 1;
        for (i = 0; s[i] != '\0'; i++) {
            if (s[i] < '0' || s[i] > '9') {
                hop_le = 0;
                break;
            }
        }
        if (hop_le && s[0] == '0' && strlen(s) > 1)
            hop_le = 0;
        if (hop_le)
            return;
        printf("Chuoi khong hop le. Nhap lai.\n");
    }
}

int la_doi_xung(const char *s) {
    int l = 0;
    int r = strlen(s) - 1;
    while (l < r) {
        if (s[l] != s[r])
            return 0;
```

```
        l++;
        r--;
    }
    return 1;
}

int main(void) {
    char a[105];
    nhap_chuoi(a, 105);
    if (la_doi_xung(a))
        printf("Chuoi tu doi xung\n");
    else
        printf("Chuoi khong tu doi xung\n");
    return 0;
}
```

7.4. Kết quả

```
Nhap chuoi so nguyen duong a: 5
Chuoi tu doi xung
```

Hình 7.1: Kết quả thực thi thành công với a là số có 1 chữ số

```
Nhap chuoi so nguyen duong a: 232
Chuoi tu doi xung
```

Hình 7.2: Kết quả thực thi thành công

7.5. Nhận xét

Two Pointers là giải pháp tối ưu cho bài toán kiểm tra palindrome, chỉ cần duyệt $n/2$ cặp ký tự. Sử dụng chuỗi thay vì số giúp xử lý được số rất lớn (vượt quá giới hạn long long). Thuật toán dừng sớm khi phát hiện không đối xứng.

8. Bài 37

8.1. Tên đề bài

Cho số tự nhiên N (N<=7000000). Phân tích N thành tổng các số chính phương nhỏ dần sao cho số các số hạng là ít nhất.

Ví dụ: $30 = 5^2 + 2^2 + 1^2$

8.2. Thuật toán

8.2.1. Phát biểu thuật toán

- Nhập N ($1 \leq N \leq 7000000$)
- Lặp N > 0
    - Tìm số chính phương lớn nhất không vượt quá N bằng công thức: $sq = \sqrt{N}$. Lưu lại sq và căn bậc hai k.

- Trừ N đi sq và lặp lại cho đến khi N = 0. Mỗi bước đều chọn số chính phương lớn nhất có thể.

- Kết quả được biểu diễn dưới dạng: $N = k_1^2 + k_2^2 + \ldots + k_m^2$

8.2.2. Mã giả (pseudocode C)

```
FUNCTION nhap_n () RETURN INTEGER
  WHILE (1)
    OUTPUT ("Nhap N (N <= 7000000): ")
    INPUT (s)
    IF (CHI_GOM_CHU_SO(s)) THEN
      n = CHUYEN_SANG_SO(s)
      IF (n > 0 && n <= 7000000) THEN
        RETURN n
      ENDIF
    ENDIF
    OUTPUT ("Gia tri khong hop le. Nhap lai.")
  ENDWHILE
ENDFUNCTION

FUNCTION so_cp_lon_nhat (INTEGER x) RETURN INTEGER
  DECLARE INTEGER k = FLOOR(SQRT(x))
  RETURN k * k
ENDFUNCTION

FUNCTION main ()
  DECLARE INTEGER N = nhap_n()
  DECLARE INTEGER temp = N
  DECLARE ARRAY cp[100], mu[100]
  DECLARE INTEGER cnt = 0
  WHILE (temp > 0)
    sq = so_cp_lon_nhat(temp)
    k = SQRT(sq)
    cp[cnt] = sq
    mu[cnt] = k
    cnt = cnt + 1
    temp = temp - sq
  ENDWHILE
  OUTPUT (N, " = ")
```

```
    FOR (i = 0; i < cnt; i++)
      OUTPUT (mu[i], "^2")
      IF (i < cnt - 1) THEN OUTPUT (" + ") ENDIF
    ENDFOR
    RETURN 0
ENDFUNCTION
```

8.3. Chương trình nguồn

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

int nhap_n() {
    char s[100];
    int i;
    while (1) {
        int hop_le = 1;
        printf("Nhap N (N <= 7000000): ");
        scanf("%s", s);
        for (i = 0; s[i] != '\0'; i++) {
            if (!isdigit(s[i])) {
                hop_le = 0;
                break;
            }
        }
        if (hop_le) {
            long long n = 0;
            for (i = 0; s[i] != '\0'; i++)
                n = n * 10 + (s[i] - '0');
            if (n > 0 && n <= 7000000)
                return (int)n;
        }
        printf("Gia tri khong hop le. Nhap lai.\n");
    }
}

int so_cp_lon_nhat(int x) {
    int k = (int)sqrt(x);
```

```
    return k * k;
}

int main() {
    int N = nhap_n();
    int temp = N;
    int cp[100];
    int mu[100];
    int cnt = 0;
    while (temp > 0) {
        int sq = so_cp_lon_nhat(temp);
        int k = (int)sqrt(sq);
        cp[cnt] = sq;
        mu[cnt] = k;
        cnt++;
        temp -= sq;
    }
    printf("%d = ", N);
    int i;
    for (i = 0; i < cnt; i++) {
        printf("%d^2", mu[i]);
        if (i < cnt - 1) printf(" + ");
    }
    printf("\n");
    return 0;
}
```

8.4. Kết quả

```
Nhap N (N <= 7000000): 1
1 = 1^2
```

Hình 8.1: Kết quả thực thi thành công với n bằng 1

```
Nhap N (N <= 7000000): 18
18 = 4^2 + 1^2 + 1^2
```

Hình 8.2: Kết quả thực thi thành công

8.5. Nhận xét

Thuật toán Greedy đơn giản, dễ cài đặt nhưng không đảm bảo số hạng tối ưu. Theo định lý Lagrange, mọi số tự nhiên đều biểu diễn được thành tổng tối đa 4 số chính phương phù hợp với giới hạn $N \leq 7000000$.

9. Bài 42

9.1. Tên đề bài

Nhập vào các số nguyên bất kì, hãy tính tổng số nguyên đó đến khi nhập vào bàn phím số 0

9.2. Thuật toán

9.2.1. Phát biểu thuật toán

- Khởi tạo $tong = 0$. Nhập số cho đến khi gặp 0.
- Trước khi cộng, kiểm tra tràn: nếu $tong > 0$ và $so > LLONG\_MAX - tong$, hoặc $tong < 0$ và $so < LLONG\_MIN - tong$, thì báo tràn.Nếu không tràn, cộng $tong \leftarrow tong + so$ và tiếp tục.
- Nếu không tràn, cộng $tong \leftarrow tong + so$ và tiếp tục.

9.2.2. Mã giả (pseudocode C)

```
FUNCTION nhap_so_nguyen () RETURN LONG_LONG
  WHILE (1)
    OUTPUT ("Nhap so (0 de dung): ")
    INPUT (s)
    hop_le = 1
    i = 0
    IF (s[0] == '-') THEN i = 1 ENDIF
    FOR (j = i; j < LENGTH(s); j++)
      IF (s[j] IS NOT DIGIT) THEN hop_le = 0; BREAK ENDIF
    ENDFOR
    IF (hop_le == 1) THEN
      x = 0
      am = (s[0] == '-')
      FOR (j = (am ? 1 : 0); j < LENGTH(s); j++)
        digit = s[j] - '0'
        IF (x > (LLONG_MAX - digit) / 10) THEN
          hop_le = 0; BREAK
        ENDIF
        x = x * 10 + digit
      ENDFOR
      IF (hop_le == 1) THEN
        RETURN (am ? -x : x)
      ENDIF
    ENDIF
    OUTPUT ("Gia tri khong hop le. Nhap lai.")
```

```
    ENDWHILE
ENDFUNCTION

FUNCTION tinh_tong () RETURN LONG_LONG
    DECLARE LONG_LONG tong = 0
    WHILE (1)
        x = nhap_so_nguyen()
        IF (x == 0) THEN BREAK ENDIF
        IF ((tong > 0 && x > LLONG_MAX - tong) ||
            (tong < 0 && x < LLONG_MIN - tong)) THEN
            OUTPUT ("Tong bi tran!")
            CONTINUE
        ENDIF
        tong = tong + x
    ENDWHILE
    RETURN tong
ENDFUNCTION

FUNCTION main ()
    kq = tinh_tong()
    OUTPUT ("Tong cac so da nhap = ", kq)
    RETURN 0
ENDFUNCTION
```

9.3. Chương trình nguồn

```
#include <stdio.h>
#include <ctype.h>
#include <limits.h>

long long nhap_so_nguyen() {
    char s[200];
    int i;
    while (1) {
        int hop_le = 1;
        printf("Nhap so (0 de dung): ");
        scanf("%s", s);
        i = 0;
        if (s[0] == '-') i = 1;
```

```
        for (; s[i] != '\0'; i++) {
            if (!isdigit(s[i])) {
                hop_le = 0;
                break;
            }
        }
        if (hop_le) {
            long long x = 0;
            int am = (s[0] == '-');
            for (i = am ? 1 : 0; s[i] != '\0'; i++) {
                int digit = s[i] - '0';
                if (x > (LLONG_MAX - digit) / 10) {
                    hop_le = 0;
                    break;
                }
                x = x * 10 + digit;
            }
            if (hop_le) {
                return am ? -x : x;
            }
        }
        printf("Gia tri khong hop le. Nhap lai.\n");
    }
}

long long tinh_tong() {
    long long tong = 0;
    long long x;
    while (1) {
        x = nhap_so_nguyen();
        if (x == 0)
            break;
        if ((tong > 0 && x > LLONG_MAX - tong) ||
            (tong < 0 && x < LLONG_MIN - tong)) {
            printf("Tong bi tran! Khong the cong them so nay.\n");
            continue;
        }
```

```
        tong += x;
    }
    return tong;
}

int main() {
    long long kq = tinh_tong();
    printf("Tong cac so da nhap = %lld\n", kq);
    return 0;
}
```

9.4. Kết quả

```
Nhap so (0 de dung): 10
Nhap so (0 de dung): 20
Nhap so (0 de dung): -5
Nhap so (0 de dung): 0
Tong cac so da nhap = 25
```

Hình 9.1: Kết quả thực thi thành công

```
Nhap so (0 de dung): abc
Gia tri khong hop le. Nhap lai.
Nhap so (0 de dung): 12a
Gia tri khong hop le. Nhap lai.
Nhap so (0 de dung): 100
Nhap so (0 de dung): 0
Tong cac so da nhap = 100
```

Hình 9.2: Kết quả thực thi thành công với trường hợp người dùng nhập chuỗi ký tự

9.5. Nhận xét

Bài toán yêu cầu xử lý an toàn với số nguyên lớn, đặc biệt là phát hiện và ngăn chặn tràn số - một vấn đề quan trọng trong lập trình hệ thống. Thuật toán sử dụng kỹ thuật kiểm tra tràn trước khi thực hiện phép toán bằng cách so sánh với giới hạn LLONG_MAX và LLONG_MIN. Phương pháp nhập chuỗi và chuyển đổi thủ công cho phép phát hiện lỗi sớm và xử lý linh hoạt các trường hợp biên.

10. Bài 54

10.1. Tên đề bài

Viết chương trình sử dụng hàm nhập số n (hệ thập phân) từ bàn phím và đổi sang cơ số cơ số c cũng nhập từ bàn phím. In kết quả sang cơ số c ra màn hình

10.2. Thuật toán

10.2.1.Phát biểu thuật toán

- Đổi phần nguyên bằng phương pháp chia lấy dư: lặp chia cho cơ số, lấy dư, đảo ngược kết quả.

- Đổi phần thập phân bằng phương pháp nhân lấy phần nguyên: lặp nhân với cơ số, lấy phần nguyên.
- Ký tự biểu diễn: 0-9 cho giá trị 0-9, A-Z cho giá trị 10-35.

10.2.2.Mã giả (pseudocode C)

```
FUNCTION doi_phan_nguyen (LONG_LONG so, INTEGER coso, STRING ketqua)
    DECLARE STRING ky_tu = "0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ"
    IF (so == 0) THEN
      ketqua = "0"
      RETURN
    ENDIF
    am = (so < 0)
    IF (am) THEN so = -so ENDIF
    i = 0
    WHILE (so > 0)
      tam[i] = ky_tu[so % coso]
      i = i + 1
      so = so / coso
    ENDWHILE
    IF (am) THEN tam[i] = '-'; i = i + 1 ENDIF
    DAO_NGUOC(tam) -> ketqua
  ENDFUNCTION

  FUNCTION doi_phan_thap_phan (DOUBLE phanle, INTEGER coso, STRING ketqua)
    DECLARE STRING ky_tu = "0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ"
    gioihan = 20
    i = 0
    WHILE (phanle > 1e-15 && i < gioihan)
      phanle = phanle * coso
      so = FLOOR(phanle)
      ketqua[i] = ky_tu[so]
      i = i + 1
      phanle = phanle - so
    ENDWHILE
```

```
    ketqua[i] = NULL
ENDFUNCTION

FUNCTION main ()
    INPUT (chuoi_nhap)
    DO
        INPUT (coso)
    WHILE (coso < 2 || coso > 36)
    so_nhap = ATOF(chuoi_nhap)
    phan_nguyen = FLOOR(so_nhap)
    phan_thap_phan = ABS(so_nhap - phan_nguyen)
    doi_phan_nguyen(phan_nguyen, coso, kq_nguyen)
    doi_phan_thap_phan(phan_thap_phan, coso, kq_thap_phan)
    OUTPUT (kq_nguyen, ".", kq_thap_phan)
    RETURN 0
ENDFUNCTION
```

10.3. Chương trình nguồn

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

#define DAI_TOI_DA 256

void doi_phan_nguyen(long long so, int coso, char *ketqua) {
    char ky_tu[] = "0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ";
    char tam[DAI_TOI_DA];
    int i;
    int am;
    int dai;
    int j;
    i = 0;
    am = 0;
    if (so == 0) {
        strcpy(ketqua, "0");
        return;
    }
```

```
    if (so < 0) {
        am = 1;
        so = -so;
    }
    while (so > 0) {
        tam[i++] = ky_tu[so % coso];
        so /= coso;
    }
    if (am)
        tam[i++] = '-';
    tam[i] = '\0';
    dai = strlen(tam);
    for (j = 0; j < dai; j++) {
        ketqua[j] = tam[dai - j - 1];
    }
    ketqua[dai] = '\0';
}

void doi_phan_thap_phan(double phanle, int coso, char *ketqua) {
    char ky_tu[] = "0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ";
    int gioihan;
    int i;
    int so;
    gioihan = 20;
    i = 0;
    while (phanle > 1e-15 && i < gioihan) {
        phanle *= coso;
        so = (int)phanle;
        ketqua[i++] = ky_tu[so];
        phanle -= so;
    }
    ketqua[i] = '\0';
}

int main() {
    char chuoi_nhap[DAI_TOI_DA];
    double so_nhap;
```

```
    int coso;
    long long phan_nguyen;
    double phan_thap_phan;
    char kq_nguyen[DAI_TOI_DA];
    char kq_thap_phan[DAI_TOI_DA];
    printf("Nhap so he thap phan: ");
    scanf("%s", chuoi_nhap);
    do {
        printf("Nhap co so can doi (2 - 36): ");
        scanf("%d", &coso);
        if (coso < 2 || coso > 36)
            printf("Co so khong hop le! Vui long nhap lai.\n");
    } while (coso < 2 || coso > 36);
    so_nhap = atof(chuoi_nhap);
    phan_nguyen = (long long)so_nhap;
    phan_thap_phan = fabs(so_nhap - phan_nguyen);
    doi_phan_nguyen(phan_nguyen, coso, kq_nguyen);
    doi_phan_thap_phan(phan_thap_phan, coso, kq_thap_phan);
    printf("Ket qua: %s", kq_nguyen);
    if (phan_thap_phan > 0)
        printf(".%s\n", kq_thap_phan);
    else
        printf("\n");
    return 0;
}
```

10.4. Kết quả

```
Nhap canh so thuc n (he thap phan): 16
Nhap co so c (2 -> 36): 2

=== Ket qua ===
10000
```

Hình 10.1: Kết quả thực thi thành công với n là số nguyên

```
Nhap canh so thuc n (he thap phan): 10.625
Nhap co so c (2 -> 36): 2

=== Ket qua ===
1010.101
```

Hình 10.2: Kết quả thực thi thành công với n là số thực

10.5. Nhận xét

Thuật toán sử dụng phương pháp chia lấy dư cho phần nguyên và nhân lấy phần nguyên cho phần thập phân là phương pháp chuẩn cho chuyển đổi cơ số. Việc hỗ trợ cơ số từ 2 đến 36 cho phép biểu diễn số bằng các ký tự 0-9 và A-Z

11. Bài 55

11.1. Tên đề bài

Viết chương trình có sử dụng hàm

a. Nhập mảng số nguyên A gồm *n* phần tử và nhập số nguyên *k.*

b. Xóa phần tử có chỉ số k ra khỏi mảng A. Xuất mảng A sau khi xóa chỉ số chỉ số k ra màn hình.

11.2. Thuật toán

11.2.1.Phát biểu thuật toán

- Kiểm tra vị trí k hợp lệ ($0 \le k < n$).
- Dịch chuyển các phần tử từ vị trí $k+1$ đến $n-1$ về trước một đơn vị: $a[i] \leftarrow a[i+1]$.
- Giảm kích thước mảng: $n \leftarrow n-1$.

11.2.2.Mã giả (pseudocode C)

```
FUNCTION nhap_so_nguyen (STRING ten) RETURN LONG_LONG
   WHILE (1)
      OUTPUT ("Nhap ", ten, ": ")
      INPUT (s)
      IF (HOP_LE(s)) THEN
         RETURN CHUYEN_SANG_SO(s)
      ENDIF
      OUTPUT ("Gia tri khong hop le. Nhap lai.")
   ENDWHILE
ENDFUNCTION

FUNCTION xoa_vi_tri (ARRAY A[], INTEGER *n, INTEGER k)
   FOR (i = k; i < *n - 1; i++)
      A[i] = A[i + 1]
   ENDFOR
   *n = *n - 1
ENDFUNCTION

FUNCTION main ()
   n = nhap_so_nguyen("so luong phan tu n")
```

```
    WHILE (n <= 0)
      n = nhap_so_nguyen("so luong phan tu n")
    ENDWHILE
    FOR (i = 0; i < n; i++)
      A[i] = nhap_so_nguyen("A[i]")
    ENDFOR
    k = nhap_so_nguyen("chi so k can xoa")
    WHILE (k < 0 || k >= n)
      k = nhap_so_nguyen("chi so k can xoa")
    ENDWHILE
    xoa_vi_tri(A, n, k)
    OUTPUT ("Mang sau khi xoa: ")
    FOR (i = 0; i < n; i++)
      OUTPUT (A[i], " ")
    ENDFOR
    RETURN 0
  ENDFUNCTION
```

11.3. Chương trình nguồn

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>

long long nhap_so_nguyen(const char *ten) {
  char s[200];
  while (1) {
    int hop_le = 1;
    printf("Nhap %s: ", ten);
    scanf("%s", s);
    int i = 0;
    if (s[0] == '-' && s[1] != '\0')
      i = 1;
    for (; s[i] != '\0'; i++) {
      if (!isdigit(s[i])) {
        hop_le = 0;
        break;
      }
    }
```

```
        if (hop_le) {
            long long x = 0;
            int dau = (s[0] == '-') ? -1 : 1;
            i = (s[0] == '-') ? 1 : 0;
            for (; s[i] != '\0'; i++)
                x = x * 10 + (s[i] - '0');
            return dau * x;
        }
        printf("Gia tri khong hop le. Nhap lai.\n");
    }
}

void xoa_vi_tri(long long A[], int *n, int k) {
    for (int i = k; i < *n - 1; i++)
        A[i] = A[i + 1];
    (*n)--;
}

int main() {
    int n = (int)nhap_so_nguyen("so luong phan tu n");
    while (n <= 0) {
        printf("n phai > 0. Nhap lai.\n");
        n = (int)nhap_so_nguyen("so luong phan tu n");
    }
    long long A[200];
    printf("\n=== Nhap cac phan tu cua mang A ===\n");
    for (int i = 0; i < n; i++) {
        char ten[50];
        sprintf(ten, "A[%d]", i);
        A[i] = nhap_so_nguyen(ten);
    }
    int k = (int)nhap_so_nguyen("chi so k can xoa");
    while (k < 0 || k >= n) {
        printf("k phai nam trong [0 .. %d]. Nhap lai.\n", n - 1);
        k = (int)nhap_so_nguyen("chi so k can xoa");
    }
    xoa_vi_tri(A, &n, k);
```

```
    printf("\n=== Mang sau khi xoa A[%d] ===\n", k);
    for (int i = 0; i < n; i++)
        printf("%lld ", A[i]);
    printf("\n");
    return 0;
}
```

11.4. Kết quả

```
Nhap so luong phan tu n: 5

=== Nhap cac phan tu cua mang A ===
Nhap A[0]: 1
Nhap A[1]: 2
Nhap A[2]: 3
Nhap A[3]: 4
Nhap A[4]: 5
Nhap chi so k can xoa: 0

=== Mang sau khi xoa A[0] ===
2 3 4 5
```

Hình 11.1: Kết quả thực thi thành công

```
Nhap so luong phan tu n: 5

=== Nhap cac phan tu cua mang A ===
Nhap A[0]: a
Gia tri khong hop le. Nhap lai.
Nhap A[0]: -1
Nhap A[1]: b
Gia tri khong hop le. Nhap lai.
Nhap A[1]: 2
Nhap A[2]: 4
Nhap A[3]: -5
Nhap A[4]: 3
Nhap chi so k can xoa: -10
k phai nam trong [0 .. 4]. Nhap lai.
Nhap chi so k can xoa: 4

=== Mang sau khi xoa A[4] ===
-1 2 4 -5
```

Hình 11.2: Kết quả thực thi thành công với trường hợp nhập chuỗi hoặc số âm

11.5. Nhận xét

Bài toán xóa phần tử trong mảng là thao tác cơ bản trong xử lý dữ liệu tuần tự, yêu cầu dịch chuyển các phần tử để lấp đầy vị trí trống. Thuật toán phải dịch chuyển tối đa n-1 phần tử trong trường hợp xấu nhất. Việc kiểm tra chỉ số hợp lệ trước khi xóa đảm bảo tính an toàn và tránh lỗi truy cập ngoài phạm vi mảng.

12. Bài 56

12.1. Tên đề bài

Viết chương trình có sử dụng hàm

a. Viết chương trình nhập mảng một chiều A với n phần tử, n được nhập từ bàn

phím. Xuất mảng A ra màn hình ?

b. Số hoàn hảo là số nguyên dương và bằng tổng các ước thực sự của nó không kể chính nó. Ví dụ: 6=1+2+3.

Xuất các số hoàn hảo trong mảng A ra màn hình ?

12.2. Thuật toán

12.2.1.Phát biểu thuật toán

- Nhập n và mảng chuỗi A[n]
- Với mỗi phần tử hợp lệ (phải là số nguyên) trong mảng, tính tổng các ước thực sự (không kể chính nó).
- Duyệt từ 2 đến $\sqrt{n}$, nếu $i \mid n$ thì cộng cả i và n/i (nếu khác nhau).
- So sánh tổng với n. Nếu bằng nhau, n là số hoàn hảo.

12.2.2.Mã giả (pseudocode C)

```
FUNCTION nhap_chuoi (STRING ten, STRING out)
   OUTPUT ("Nhap ", ten, ": ")
   INPUT (out)
ENDFUNCTION

FUNCTION la_so_nguyen (STRING s) RETURN INTEGER
   i = 0
   IF (s[0] == '-' && s[1] != NULL) THEN i = 1 ENDIF
   FOR (j = i; j < LENGTH(s); j++)
      IF (s[j] IS NOT DIGIT) THEN RETURN 0 ENDIF
   ENDFOR
   RETURN 1
ENDFUNCTION

FUNCTION to_ll (STRING s) RETURN LONG_LONG
   x = 0
   dau = (s[0] == '-') ? -1 : 1
   i = (s[0] == '-') ? 1 : 0
   FOR (j = i; j < LENGTH(s); j++)
      x = x * 10 + (s[j] - '0')
   ENDFOR
   RETURN dau * x
ENDFUNCTION

FUNCTION so_hoan_hao (LONG_LONG x) RETURN INTEGER
```

```
    IF (x <= 0) THEN RETURN 0 ENDIF
    sum = 0
    FOR (i = 1; i <= x / 2; i++)
      IF (x % i == 0) THEN sum = sum + i ENDIF
    ENDFOR
    RETURN (sum == x)
  ENDFUNCTION

  FUNCTION main ()
    nhap n
    FOR (i = 0; i < n; i++)
      nhap_chuoi(ten, A[i])
    ENDFOR
    FOR (i = 0; i < n; i++)
      IF (la_so_nguyen(A[i])) THEN
        val = to_ll(A[i])
        IF (so_hoan_hao(val)) THEN
          OUTPUT (val)
        ENDIF
      ENDIF
    ENDFOR
    RETURN 0
  ENDFUNCTION
```

12.3. Chương trình nguồn

```
#include <stdio.h>
#include <ctype.h>
#include <string.h>

void nhap_chuoi(const char *ten, char *out) {
  printf("Nhap %s: ", ten);
  scanf("%s", out);
}

int la_so_nguyen(const char *s) {
  int i = 0;
  if (s[0] == '-' && s[1] != '\0')
    i = 1;
```

```
    for (; s[i]; i++)
        if (!isdigit(s[i]))
            return 0;
    return 1;
}

long long to_ll(const char *s) {
    long long x = 0;
    int dau = (s[0] == '-') ? -1 : 1;
    int i = (s[0] == '-') ? 1 : 0;
    for (; s[i]; i++)
        x = x * 10 + (s[i] - '0');
    return dau * x;
}

int so_hoan_hao(long long x) {
    if (x <= 0) return 0;
    long long sum = 0;
    for (long long i = 1; i * 1LL <= x / 2; i++)
        if (x % i == 0)
            sum += i;
    return (sum == x);
}

int main() {
    int n = 0;
    while (1) {
        n = 0;
        char buf[100];
        nhap_chuoi("so luong phan tu n", buf);
        if (la_so_nguyen(buf)) {
            n = (int)to_ll(buf);
            if (n > 0)
                break;
        }
        printf("n phai la so nguyen duong > 0. Nhap lai.\n");
    }
```

```
    char A[200][100];
    printf("\n=== Nhap cac phan tu cua mang A ===\n");
    for (int i = 0; i < n; i++) {
        char ten[50];
        sprintf(ten, "A[%d]", i);
        nhap_chuoi(ten, A[i]);
    }
    printf("\n=== Mang A vua nhap ===\n");
    for (int i = 0; i < n; i++)
        printf("%s ", A[i]);
    printf("\n\n=== Cac so hoan hao trong mang ===\n");
    int co = 0;
    for (int i = 0; i < n; i++) {
        if (la_so_nguyen(A[i])) {
            long long val = to_ll(A[i]);
            if (so_hoan_hao(val)) {
                printf("%lld ", val);
                co = 1;
            }
        }
    }
    if (!co)
        printf("Khong co so hoan hao nao.");
    printf("\n");
    return 0;
}
```

12.4. Kết quả

```
Nhap so luong phan tu n: 6
a[0] = 1
a[1] = 6
a[2] = 28
a[3] = 12
a[4] = 496
a[5] = 10
Mang vua nhap:
1 6 28 12 496 10
Cac so hoan hao trong mang: 6 28 496
```

Hình 12.1: Kết quả thực thi thành công

12.5. Nhận xét

Thuật toán kiểm tra số hoàn hảo, tối ưu hơn so với việc duyệt đến $\sqrt{n}$ giúp giảm đáng kể số phép tính so với duyệt đến $n/2$..

13. Bài 61

13.1. Tên đề bài

Viết chương trình có sử dụng hàm

a. Viết chương trình nhập mảng một chiều A với n phần tử (n>=15). Xuất mảng A ra màn hình.

b. Tính tổng các số nguyên tố trong mảng, xuất kết quả ra màn hình.

13.2. Thuật toán:

13.2.1.Phát biểu thuật toán

- Với mỗi số n trong mảng, kiểm tra tính nguyên tố: duyệt từ 2 đến $\sqrt{n}$, nếu có ước thì không nguyên tố.
- Nếu là nguyên tố, cộng vào tổng và xuất ra.
- Sau khi duyệt hết mảng, trả về tổng các số nguyên tố.

13.2.2.Mã giả (pseudocode C)

```
CONSTANT MAX = 200

FUNCTION la_so_nguyen_thuan (STRING s) RETURN INTEGER
   DECLARE INTEGER i = 0
   IF (s[0] == '-' && s[1] != NULL) THEN
      i = 1
   ENDIF
   FOR (k = i; k < LENGTH(s); k++)
      IF (!IsDigit(s[k])) THEN
         RETURN 0
      ENDIF
   ENDFOR
   RETURN 1
ENDFUNCTION

FUNCTION la_nguyen_to (LONG_LONG x) RETURN INTEGER
   IF (x < 2) THEN
      RETURN 0
   ENDIF
   FOR (i = 2; i <= FLOOR(SQRT(x)); i++)
      IF (x % i == 0) THEN
```

```
            RETURN 0
        ENDIF
    ENDFOR
    RETURN 1
ENDFUNCTION

FUNCTION main () RETURN INTEGER
    DECLARE ARRAY A[MAX][100]
    DECLARE INTEGER n
    DECLARE LONG_LONG val, tong
    DO
        OUTPUT ("Nhap so luong phan tu n (n >= 15): ")
        INPUT (n)
        IF (n < 15) THEN
            OUTPUT ("n phai lon hon hoac bang 15!")
        ENDIF
    WHILE (n < 15)
    FOR (i = 0; i < n; i++)
        OUTPUT ("Nhap A[", i, "]: ")
        INPUT (A[i])
    ENDFOR
    tong = 0
    FOR (i = 0; i < n; i++)
        IF (la_so_nguyen_thuan(A[i]) == 1) THEN
            val = ATOLL(A[i])
            IF (la_nguyen_to(val) == 1) THEN
                tong = tong + val
            ENDIF
        ENDIF
    ENDFOR
    OUTPUT ("Tong cac so nguyen to trong mang = ", tong)
    RETURN 0
ENDFUNCTION
```

13.3. Chương trình nguồn

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>
```

```
#include <stdlib.h>

#define MAX 200

int la_so_nguyen_thuan(const char *s)
{
   int i = 0;
   if (s[0] == '-' && s[1] != '\0')
      i = 1;
   for (; s[i] != '\0'; i++)
      if (!isdigit(s[i]))
         return 0;
   return 1;
}

int la_nguyen_to(long long x)
{
   if (x < 2) return 0;
   for (long long i = 2; i * i <= x; i++)
      if (x % i == 0)
         return 0;
   return 1;
}

int main()
{
   int n;
   char A[MAX][100];
   long long val;
   long long tong = 0;
   printf("Nhap so luong phan tu n (>=15): ");
   scanf("%d", &n);
   while (n < 15)
   {
      printf("n phai >= 15. Nhap lai: ");
      scanf("%d", &n);
   }
```

```
    printf("\n=== Nhap cac phan tu cua mang A ===\n");
    for (int i = 0; i < n; i++)
    {
        printf("Nhap A[%d]: ", i);
        scanf("%s", A[i]);
    }
    printf("\n=== Mang A vua nhap ===\n");
    for (int i = 0; i < n; i++)
        printf("%s ", A[i]);
    printf("\n\n=== Tong cac so nguyen to trong mang ===\n");
    for (int i = 0; i < n; i++)
    {
        if (la_so_nguyen_thuan(A[i]))
        {
            val = atoll(A[i]);
            if (la_nguyen_to(val))
                tong += val;
        }
    }
    printf("Tong = %lld\n", tong);
    return 0;
}
```

13.4. Kết quả

```
Nhap so luong phan tu n (>=15): 15

=== Nhap cac phan tu cua mang A ===
Nhap A[0]: 2
Nhap A[1]: 4
Nhap A[2]: 5
Nhap A[3]: 7
Nhap A[4]: 9
Nhap A[5]: 11
Nhap A[6]: 13
Nhap A[7]: 16
Nhap A[8]: 17
Nhap A[9]: 19
Nhap A[10]: 21
Nhap A[11]: 23
Nhap A[12]: 25
Nhap A[13]: 29
Nhap A[14]: 31

=== Mang A vua nhap ===
2 4 5 7 9 11 13 16 17 19 21 23 25 29 31

=== Tong cac so nguyen to trong mang ===
Tong = 157
```

Hình 13.1: Kết quả thực thi thành công

```
Nhap so luong phan tu n (>=15): 15

=== Nhap cac phan tu cua mang A ===
Nhap A[0]: 2.0
Nhap A[1]: 4
Nhap A[2]: 5
Nhap A[3]: 7
Nhap A[4]: 9
Nhap A[5]: 11
Nhap A[6]: 13
Nhap A[7]: 16
Nhap A[8]: 17
Nhap A[9]: 19
Nhap A[10]: 21
Nhap A[11]: 23
Nhap A[12]: 25
Nhap A[13]: 29
Nhap A[14]: 31

=== Mang A vua nhap ===
2.0 4 5 7 9 11 13 16 17 19 21 23 25 29 31
```

Hình 13.2: Kết quả thực thi thành công với số thực

13.5. Nhận xét

Kiểm tra nguyên tố bằng cách duyệt đến $\sqrt{n}$ là tối ưu về thời gian. Điều kiện n ≥ 15 phần tử đảm bảo mảng có đủ dữ liệu để thể hiện tính đa dạng của thuật toán.

14. Bài 62

14.1. Tên đề bài

Viết chương trình sử dụng hàm

a. Nhập các số tự nhiên từ bàn phím.

b. Đếm có bao nhiêu số 0 tận cùng của tích n số tự nhiên trên?

14.2. Thuật toán

14.2.1.Phát biểu bài toán

- Đếm số lần thừa số 2 và 5 trong tất cả các số của mảng.
- Mỗi số 0 tận cùng được tạo từ một cặp (2, 5). Kết quả = min (dem_2, dem_5).

14.2.2.Mã giả (pseudo code C)

```
CONSTANT MAX = 250

FUNCTION nhap_so_tu_nhien (STRING ten) RETURN LONG_LONG
   WHILE (1)
      OUTPUT ("Nhap ", ten, ": ")
      INPUT (s)
      hop_le = 1
      FOR (i = 0; i < LENGTH(s); i++)
         IF (!IsDigit(s[i])) THEN
            hop_le = 0
            BREAK
         ENDIF
      ENDFOR
      IF (hop_le == 1) THEN
         x = CHUYEN_SANG_SO(s)
         RETURN x
      ENDIF
      OUTPUT ("Gia tri khong hop le. Nhap lai.")
   ENDWHILE
ENDFUNCTION

FUNCTION dem_so_mu (LONG_LONG x, INTEGER p) RETURN
LONG_LONG
   DECLARE LONG_LONG dem = 0
   WHILE (x % p == 0 && x > 0)
      dem = dem + 1
      x = x / p
```

```
    ENDWHILE
    RETURN dem
ENDFUNCTION

FUNCTION main () RETURN INTEGER
    DECLARE ARRAY A[MAX]
    DECLARE LONG_LONG n, tong_mu2, tong_mu5, so_0_tan_cung
    n = nhap_so_tu_nhien("so luong phan tu n")
    WHILE (n <= 0)
        OUTPUT ("n phai > 0. Nhap lai.")
        n = nhap_so_tu_nhien("so luong phan tu n")
    ENDWHILE
    FOR (i = 0; i < n; i++)
        A[i] = nhap_so_tu_nhien("A[i]")
    ENDFOR
    tong_mu2 = 0
    tong_mu5 = 0
    FOR (i = 0; i < n; i++)
        tong_mu2 = tong_mu2 + dem_so_mu(A[i], 2)
        tong_mu5 = tong_mu5 + dem_so_mu(A[i], 5)
    ENDFOR
    IF (tong_mu2 < tong_mu5) THEN
        so_0_tan_cung = tong_mu2
    ELSE
        so_0_tan_cung = tong_mu5
    ENDIF
    OUTPUT ("So chu so 0 tan cung cua tich = ", so_0_tan_cung)
    RETURN 0
ENDFUNCTION
```

14.3. Chương trình nguồn

```
#include <stdio.h>
#include <ctype.h>

long long nhap_so_tu_nhien(const char *ten)
{
    char s[100];
    while (1)
```

```
    {
        int hop_le = 1;
        printf("Nhap %s: ", ten);
        scanf("%s", s);
        int i = 0;
        for (; s[i] != '\0'; i++)
        {
            if (!isdigit(s[i]))
            {
                hop_le = 0;
                break;
            }
        }
        if (hop_le)
        {
            long long x = 0;
            for (i = 0; s[i] != '\0'; i++)
                x = x * 10 + (s[i] - '0');
            return x;
        }
        printf("Gia tri khong hop le. Nhap lai.\n");
    }
}

long long dem_so_mu(long long x, int p)
{
    long long dem = 0;
    while (x % p == 0 && x > 0)
    {
        dem++;
        x /= p;
    }
    return dem;
}

int main()
{
```

```
    long long n = nhap_so_tu_nhien("so luong phan tu n");
    while (n <= 0)
    {
        printf("n phai > 0. Nhap lai.\n");
        n = nhap_so_tu_nhien("so luong phan tu n");
    }
    long long A[250];
    printf("\n=== Nhap cac so tu nhien ===\n");
    for (long long i = 0; i < n; i++)
    {
        char ten[50];
        sprintf(ten, "A[%lld]", i);
        A[i] = nhap_so_tu_nhien(ten);
    }
    printf("\n=== Day so vua nhap ===\n");
    for (long long i = 0; i < n; i++)
        printf("%lld ", A[i]);
    long long tong_mu2 = 0, tong_mu5 = 0;
    for (long long i = 0; i < n; i++)
    {
        tong_mu2 += dem_so_mu(A[i], 2);
        tong_mu5 += dem_so_mu(A[i], 5);
    }
    long long so_0_tan_cung = (tong_mu2 < tong_mu5) ? tong_mu2 : tong_mu5;
    printf("\n\n=== So chu so 0 tan cung cua tich ===\n");
    printf("Ket qua: %lld\n", so_0_tan_cung);
    return 0;
}
```

14.4. Kết quả

```
2> ./Bai_62.exe
Nhap so luong phan tu n: 3

=== Nhap cac so tu nhien ===
Nhap A[0]: 10
Nhap A[1]: 1
Nhap A[2]: 2

=== Day so vua nhap ===
10 1 2

=== So chu so 0 tan cung cua tich ===
Ket qua: 1
```

Hình 14.1: Kết quả thực thi thành công

14.5. Nhận xét

Bài toán vận dụng kiến thức về phân tích thừa số nguyên tố để đếm số 0 tận cùng mà không cần tính trực tiếp tích, tránh tràn số với các mảng lớn. Thuật toán dựa trên nguyên lý $10 = 2 \times 5$, mỗi cặp thừa số (2, 5) tạo ra một số 0 tận cùng.

15. Bài 63

15.1. Tên đề bài

Viết chương trình nhập mảng 1 chiều A có N phần tử có sử dụng hàm (với N nhập từ bàn phím).

a. Xuất các phần tử đã nhập ra màn hình

b. Xuất ra màn hình các chính phương của mảng và tính tổng các số đó?

15.2. Thuật toán

15.2.1.Phát biểu thuật toán:

- Với mỗi số $n \geq 0$, tính $k = \sqrt{n} + 0.5$ (để tránh sai số floating-point).
- Kiểm tra $k^2 = n$. Nếu đúng, n là số chính phương.
- Cộng tất cả số chính phương tìm được và xuất kết quả.

15.2.2.Mã giả (pseudocode C)

```
CONSTANT MAX = 300

FUNCTION nhap_chuoi (STRING ten, STRING out)
   OUTPUT ("Nhap ", ten, ": ")
   INPUT (out)
ENDFUNCTION

FUNCTION la_so_nguyen (STRING s) RETURN INTEGER
   DECLARE INTEGER i = 0
   IF (s[0] == '-' && s[1] != NULL) THEN
```

```
        i = 1
    ENDIF
    FOR (k = i; k < LENGTH(s); k++)
        IF (!IsDigit(s[k])) THEN
            RETURN 0
        ENDIF
    ENDFOR
    RETURN 1
ENDFUNCTION

FUNCTION to_ll (STRING s) RETURN LONG_LONG
    DECLARE LONG_LONG x = 0
    DECLARE INTEGER dau = (s[0] == '-') ? -1 : 1
    DECLARE INTEGER i = (s[0] == '-') ? 1 : 0
    FOR (k = i; k < LENGTH(s); k++)
        x = x * 10 + (s[k] - '0')
    ENDFOR
    RETURN dau * x
ENDFUNCTION

FUNCTION la_chinh_phuong (LONG_LONG x) RETURN INTEGER
    IF (x < 0) THEN
        RETURN 0
    ENDIF
    DECLARE LONG_LONG t = FLOOR(SQRT(x) + 0.5)
    IF (t * t == x) THEN
        RETURN 1
    ELSE
        RETURN 0
    ENDIF
ENDFUNCTION

FUNCTION main () RETURN INTEGER
    DECLARE ARRAY A[MAX][100]
    DECLARE INTEGER n
    DECLARE LONG_LONG val, tong
    DO
```

```
        nhap_chuoi("so luong phan tu n", buf)
        IF (la_so_nguyen(buf)) THEN
            n = to_ll(buf)
            IF (n > 0) THEN BREAK ENDIF
        ENDIF
        OUTPUT ("n phai la so nguyen duong > 0. Nhap lai.")
    WHILE (1)
    FOR (i = 0; i < n; i++)
        nhap_chuoi("A[i]", A[i])
    ENDFOR
    tong = 0
    FOR (i = 0; i < n; i++)
        IF (la_so_nguyen(A[i]) == 1) THEN
            val = to_ll(A[i])
            IF (la_chinh_phuong(val) == 1) THEN
                OUTPUT (val, " ")
                tong = tong + val
            ENDIF
        ENDIF
    ENDFOR
    OUTPUT ("Tong cac so chinh phuong = ", tong)
    RETURN 0
ENDFUNCTION
```

15.3. Chương trình nguồn

```
#include <stdio.h>
#include <ctype.h>
#include <string.h>
#include <math.h>

void nhap_chuoi(const char *ten, char *out)
{
    printf("Nhap %s: ", ten);
    scanf("%s", out);
}

int la_so_nguyen(const char *s)
{
```

```
    int i = 0;
    if (s[0] == '-' && s[1] != '\0')
        i = 1;
    for (; s[i]; i++)
        if (!isdigit(s[i]))
            return 0;
    return 1;
}

long long to_ll(const char *s)
{
    long long x = 0;
    int dau = (s[0] == '-') ? -1 : 1;
    int i = (s[0] == '-') ? 1 : 0;
    for (; s[i]; i++)
        x = x * 10 + (s[i] - '0');
    return dau * x;
}

int la_chinh_phuong(long long x)
{
    if (x < 0) return 0;
    long long t = (long long)(sqrt((long double)x) + 0.5);
    return (t * t == x);
}

int main()
{
    int n = 0;
    while (1)
    {
        char buf[100];
        nhap_chuoi("so luong phan tu n", buf);
        if (la_so_nguyen(buf))
        {
            n = (int)to_ll(buf);
            if (n > 0)
```

```
            break;
        }
        printf("n phai la so nguyen duong > 0. Nhap lai.\n");
    }
    char A[300][100];
    printf("\n=== Nhap cac phan tu cua mang A ===\n");
    for (int i = 0; i < n; i++)
    {
        char ten[50];
        sprintf(ten, "A[%d]", i);
        nhap_chuoi(ten, A[i]);
    }
    printf("\n=== Mang A vua nhap ===\n");
    for (int i = 0; i < n; i++)
        printf("%s ", A[i]);
    printf("\n\n=== Cac so chinh phuong trong mang ===\n");
    int co = 0;
    long long tong = 0;
    for (int i = 0; i < n; i++)
    {
        if (la_so_nguyen(A[i]))
        {
            long long val = to_ll(A[i]);
            if (la_chinh_phuong(val))
            {
                printf("%lld ", val);
                tong += val;
                co = 1;
            }
        }
    }
    if (!co)
        printf("Khong co so chinh phuong nao.");
    printf("\nTong cac so chinh phuong = %lld\n", tong);
    return 0;
}
```

15.4. Kết quả

```
Nhap so luong phan tu n: 5

=== Nhap cac phan tu cua mang A ===
Nhap A[0]: 0
Nhap A[1]: 1
Nhap A[2]: 2
Nhap A[3]: 4
Nhap A[4]: 10

=== Mang A vua nhap ===
0 1 2 4 10

=== Cac so chinh phuong trong mang ===
0 1 4
Tong cac so chinh phuong = 5
```

Hình 15.1: Kết quả thực thi thành công

```
Nhap so luong phan tu n: 6

=== Nhap cac phan tu cua mang A ===
Nhap A[0]: 0
Nhap A[1]: a
Nhap A[2]: 1.0
Nhap A[3]: -10
Nhap A[4]: 4
Nhap A[5]: 10

=== Mang A vua nhap ===
0 a 1.0 -10 4 10

=== Cac so chinh phuong trong mang ===
0 4
Tong cac so chinh phuong = 4
```

Hình 15.2: Kết quả thực thi với giá trị là chuỗi ký tự chữ hoặc số âm

```
Nhap so luong phan tu n: 5

=== Nhap cac phan tu cua mang A ===
Nhap A[0]: 2
Nhap A[1]: 3
Nhap A[2]: 5
Nhap A[3]: 6
Nhap A[4]: 7

=== Mang A vua nhap ===
2 3 5 6 7

=== Cac so chinh phuong trong mang ===
Khong co so chinh phuong nao.
Tong cac so chinh phuong = 0
```

Hình 15.3: Kết quả thực thi với giá trị đầu vào không có số chính phương

15.5. Nhận xét

Thuật toán kiểm tra số chính phương sử dụng hàm căn bậc hai so sánh bình phương của phần nguyên với số ban đầu. Số 0 được coi là số chính phương vì $0 = 0^2$, trong khi số âm không phải số chính phương trong tập số thực.

16. Bài 64

16.1. Tên đề bài

Viết chương trình nhập mảng 1 chiều A có N phần tử có sử dụng hàm (với N nhập từ bàn phím).:

a. Xuất các phần tử đã nhập ra màn hình.

b. Xuất ra màn hình số nguyên tố cuối cùng của mảng (nếu có),còn không xuất ra dòng "Không có số nguyên tố trong mảng" ?

16.2. Thuật toán

16.2.1.Phát biểu thuật toán

- Nhập mảng n phần tử
- Duyệt ngược từ $i = n - 1$ về 0
    - Với mỗi $a[i]$, kiểm tra có phải số nguyên tố không
    - Nếu $a[i]$ là số nguyên tố → trả về $a[i]$ và dừng
- Nếu duyệt hết mà không tìm thấy thì thông báo không có số nguyên tố
- Kiểm tra nguyên tố bằng cách duyệt từ 2 đến $\sqrt{n}$

16.2.2.Mã giả (pseudocode C)

```
CONSTANT MAX = 200

FUNCTION nhap_chuoi (STRING ten, STRING out)
  OUTPUT ("Nhap ", ten, ": ")
  INPUT (out)
ENDFUNCTION

FUNCTION la_so_nguyen (STRING s) RETURN INTEGER
  DECLARE INTEGER i = 0
  IF (s[0] == '-' && s[1] != NULL) THEN
    i = 1
  ENDIF
  FOR (k = i; k < LENGTH(s); k++)
    IF (!IsDigit(s[k])) THEN
      RETURN 0
    ENDIF
  ENDFOR
  RETURN 1
ENDFUNCTION

FUNCTION to_ll (STRING s) RETURN LONG_LONG
```

```
    DECLARE LONG_LONG x = 0
    DECLARE INTEGER dau = (s[0] == '-') ? -1 : 1
    DECLARE INTEGER i = (s[0] == '-') ? 1 : 0
    FOR (k = i; k < LENGTH(s); k++)
      x = x * 10 + (s[k] - '0')
    ENDFOR
    RETURN dau * x
ENDFUNCTION

FUNCTION la_nguyen_to (LONG_LONG x) RETURN INTEGER
    IF (x < 2) THEN
      RETURN 0
    ENDIF
    FOR (i = 2; i <= FLOOR(SQRT(x)); i++)
      IF (x % i == 0) THEN
        RETURN 0
      ENDIF
    ENDFOR
    RETURN 1
ENDFUNCTION

FUNCTION main () RETURN INTEGER
    DECLARE ARRAY A[MAX][100]
    DECLARE INTEGER N
    DECLARE LONG_LONG val, so_nt_cuoi
    DO
      nhap_chuoi("so luong phan tu N", buf)
      IF (la_so_nguyen(buf)) THEN
        N = to_ll(buf)
        IF (N > 0) THEN BREAK ENDIF
      ENDIF
      OUTPUT ("N phai la so nguyen duong > 0. Nhap lai.")
    WHILE (1)
    FOR (i = 0; i < N; i++)
      nhap_chuoi("A[i]", A[i])
    ENDFOR
    so_nt_cuoi = -1
```

```
    FOR (i = 0; i < N; i++)
      IF (la_so_nguyen(A[i]) == 1) THEN
        val = to_ll(A[i])
        IF (la_nguyen_to(val) == 1) THEN
          so_nt_cuoi = val
        ENDIF
      ENDIF
    ENDFOR
    IF (so_nt_cuoi == -1) THEN
      OUTPUT ("Khong co so nguyen to trong mang.")
    ELSE
      OUTPUT ("So nguyen to cuoi cung trong mang = ", so_nt_cuoi)
    ENDIF
    RETURN 0
  ENDFUNCTION
```

16.3. Chương trình nguồn

```
#include <stdio.h>
#include <ctype.h>
#include <string.h>
#include <math.h>

void nhap_chuoi(const char *ten, char *out)
{
  printf("Nhap %s: ", ten);
  scanf("%s", out);
}

int la_so_nguyen(const char *s)
{
  int i = 0;
  if (s[0] == '-' && s[1] != '\0')
    i = 1;
  for (; s[i]; i++)
    if (!isdigit(s[i]))
      return 0;
  return 1;
}
```

```
long long to_ll(const char *s)
{
    long long x = 0;
    int dau = (s[0] == '-') ? -1 : 1;
    int i = (s[0] == '-') ? 1 : 0;
    for (; s[i]; i++)
        x = x * 10 + (s[i] - '0');
    return dau * x;
}

int la_nguyen_to(long long x)
{
    if (x < 2) return 0;
    for (long long i = 2; i * i <= x; i++)
        if (x % i == 0)
            return 0;
    return 1;
}

int main()
{
    int N = 0;
    while (1)
    {
        char buf[100];
        nhap_chuoi("so luong phan tu N", buf);
        if (la_so_nguyen(buf))
        {
            N = (int)to_ll(buf);
            if (N > 0)
                break;
        }
        printf("N phai la so nguyen duong > 0. Nhap lai.\n");
    }
    char A[200][100];
    printf("\n=== Nhap cac phan tu cua mang A ===\n");
```

```
    for (int i = 0; i < N; i++)
    {
        char ten[50];
        sprintf(ten, "A[%d]", i);
        nhap_chuoi(ten, A[i]);
    }
    printf("\n=== Mang A vua nhap ===\n");
    for (int i = 0; i < N; i++)
        printf("%s ", A[i]);
    long long so_nt_cuoi = -1;
    for (int i = 0; i < N; i++)
    {
        if (la_so_nguyen(A[i]))
        {
            long long val = to_ll(A[i]);
            if (la_nguyen_to(val))
                so_nt_cuoi = val;
        }
    }
    printf("\n\n=== So nguyen to cuoi cung trong mang ===\n");
    if (so_nt_cuoi == -1)
        printf("Khong co so nguyen to trong mang.\n");
    else
        printf("%lld\n", so_nt_cuoi);
    return 0;
}
```

16.4. Kết quả

```
Nhap so luong phan tu N: 7

=== Nhap cac phan tu cua mang A ===
Nhap A[0]: 4
Nhap A[1]: 2
Nhap A[2]: 9
Nhap A[3]: 11
Nhap A[4]: 15
Nhap A[5]: 17
Nhap A[6]: 20

=== Mang A vua nhap ===
4 2 9 11 15 17 20

=== So nguyen to cuoi cung trong mang ===
17
```

Hình 16.1: Kết quả thực thi thành công

```
Nhap so luong phan tu N: 4

=== Nhap cac phan tu cua mang A ===
Nhap A[0]: 1
Nhap A[1]: 17
Nhap A[2]: abc
Nhap A[3]: 2.0

=== Mang A vua nhap ===
1 17 abc 2.0

=== So nguyen to cuoi cung trong mang ===
17
```

Hình 16.2: Kết quả thực thi thành công với giá trị đầu vào là chuỗi ký tự hoặc số thực

```
Nhap so luong phan tu N: 6

=== Nhap cac phan tu cua mang A ===
Nhap A[0]: 1
Nhap A[1]: 4
Nhap A[2]: 6
Nhap A[3]: 8
Nhap A[4]: 9
Nhap A[5]: 10

=== Mang A vua nhap ===
1 4 6 8 9 10

=== So nguyen to cuoi cung trong mang ===
Khong co so nguyen to trong mang.
```

Hình 16.3: Kết quả thực thi thành công với giá trị đầu vào không có số nguyên tố

16.5. Nhận xét

Bài toán yêu cầu tìm số nguyên tố cuối cùng trong mảng, sử duyệt ngược để tối ưu hiệu suất. Thay vì duyệt xuôi và lưu lại số nguyên tố cuối cùng gặp được, ta duyệt

từ cuối mảng về đầu và dừng ngay khi tìm thấy số nguyên tố đầu tiên. Phương pháp này giảm số lần kiểm tra trung bình, đặc biệt hiệu quả khi số nguyên tố xuất hiện ở phần cuối mảng.

17. Bài 66

17.1. Tên đề bài

Viết chương trình có sử dụng hàm

a. Viết chương trình nhập mảng một chiều A. Xuất mảng A ra màn hình

b. Xóa các phần âm trong mảng A. Xuất mảng A sau khi xóa các phần tử âm ra màn hình.

17.2. Thuật toán

17.2.1.Phát biểu thuật toán

- Dùng hai con trỏ: i (đọc) và j (ghi), khởi tạo $j = 0$.
- Duyệt i từ 0 đến $n - 1$
    - Nếu $a[i] \geq 0$ thì gán $a[j] = a[i]$ và tăng j
    - Nếu $a[i] < 0$ thì bỏ qua
- Trả về j là số phần tử mới sau khi xó

17.2.2.Mã giả (pseudocode C)

```
FUNCTION nhap_chuoi (STRING ten, STRING out)
   OUTPUT ("Nhap ", ten, ": ")
   INPUT (out)
ENDFUNCTION

FUNCTION la_so_am (STRING s) RETURN INTEGER
   RETURN (s[0] == '-')
ENDFUNCTION

FUNCTION main ()
   DECLARE ARRAY A[200][100], B[200][100]
   DECLARE INTEGER n, m

   WHILE (1)
      INPUT (buf)
      hop_le = 1
      FOR (i = 0; i < LENGTH(buf); i++)
         IF (!IsDigit(buf[i])) THEN hop_le = 0 ENDIF
      ENDFOR
      IF (hop_le == 1) THEN
```

```
            n = CHUYEN_SANG_SO(buf)
            IF (n > 0) THEN BREAK ENDIF
         ENDIF
         OUTPUT ("n phai la so nguyen duong > 0. Nhap lai.")
      ENDWHILE

      FOR (i = 0; i < n; i++)
         nhap_chuoi("A[i]", A[i])
      ENDFOR

      m = 0
      FOR (i = 0; i < n; i++)
         IF (!la_so_am(A[i])) THEN
            STRCPY(B[m], A[i])
            m = m + 1
         ENDIF
      ENDFOR

      OUTPUT ("Mang sau khi xoa cac so am:")
      FOR (i = 0; i < m; i++)
         OUTPUT (B[i], " ")
      ENDFOR

      RETURN 0
   ENDFUNCTION
```

17.3. Chương trình nguồn

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>

void nhap_chuoi(const char *ten, char *out)
{
   printf("Nhap %s: ", ten);
   scanf("%s", out);
}

int la_so_am(const char *s)
```

```
{
    return (s[0] == '-');
}

int main()
{
    int n;
    while (1)
    {
        char buf[50];
        printf("Nhap so luong phan tu n: ");
        scanf("%s", buf);
        int hop_le = 1;
        for (int i = 0; buf[i]; i++)
            if (!isdigit(buf[i]))
                hop_le = 0;
        if (hop_le)
        {
            n = 0;
            for (int i = 0; buf[i]; i++)
                n = n * 10 + (buf[i] - '0');
            if (n > 0) break;
        }
        printf("n phai la so nguyen duong > 0. Nhap lai.\n");
    }
    char A[200][100];
    char B[200][100];
    int m = 0;
    printf("\n=== Nhap cac phan tu cua mang A ===\n");
    for (int i = 0; i < n; i++)
    {
        char ten[20];
        sprintf(ten, "A[%d]", i);
        nhap_chuoi(ten, A[i]);
    }
    printf("\n=== Mang A vua nhap ===\n");
    for (int i = 0; i < n; i++)
```

```
        printf("%s ", A[i]);
    printf("\n\n=== Mang A sau khi xoa cac so am ===\n");
    for (int i = 0; i < n; i++)
    {
        if (!la_so_am(A[i]))
        {
            strcpy(B[m], A[i]);
            m++;
        }
    }
    for (int i = 0; i < m; i++)
        printf("%s ", B[i]);
    printf("\n");
    return 0;
}
```

17.4. Kết quả

```
Nhap so luong phan tu n: 4

=== Nhap cac phan tu cua mang A ===
Nhap A[0]: -5
Nhap A[1]: 10
Nhap A[2]: -1
Nhap A[3]: 2

=== Mang A vua nhap ===
-5 10 -1 2

=== Mang A sau khi xoa cac so am ===
10 2
```

Hình 17.1: Kết quả thực thi thành công

```
Nhap so luong phan tu n: 4

=== Nhap cac phan tu cua mang A ===
Nhap A[0]: -5
Nhap A[1]: a
Nhap A[2]: 10
Nhap A[3]: -1

=== Mang A vua nhap ===
-5 a 10 -1

=== Mang A sau khi xoa cac so am ===
a 10
```

Hình 17.2: Kết quả thực thi thành công với đầu vào là mảng chứa giá trị có chuỗi ký tự

17.5. Nhận xét

Kỹ thuật Two Pointers giúp xóa nhiều phần tử hiệu quả hơn nhiều so với việc xóa từng phần tử. Thuật toán in-place, không cần bộ nhớ phụ.

18. Bài 70

18.1. Tên đề bài

Viết chương trình có sử dụng hàm

a. Viết chương trình nhập hai mảng một chiều A và B. Xuất mảng A và B ra màn hình.

b. Nối mảng A vào B (không dùng mảng phụ và không đươc sắp xếp trước và sau) đảm bảo thứ tự tăng dần. Xuất mảng sau khi nối ra màn hình.

18.2. Thuật toán

18.2.1.Phát biểu thuật toán

- Nhập số phần tử n_A của mảng A (kiểm tra hợp lệ: là số nguyên dương)
- Nhập số phần tử n_B của mảng B (kiểm tra hợp lệ)
- Nhập các phần tử của mảng A và B
- Sắp xếp C bằng Insertion Sort: với mỗi phần tử, chèn vào vị trí đúng trong phần đã sắp xếp..
- Xuất mảng đã sắp xếp

18.2.2.Mã giả (pseudocode C)

```
CONSTANT MAX = 100

FUNCTION nhapMang (ARRAY a[], POINTER n, CHAR ten)
   DECLARE INTEGER i
   OUTPUT ("Nhap so phan tu mang ", ten, ": ")
   INPUT (*n)
   FOR (i = 0; i < *n; i++)
      OUTPUT (ten, "[", i, "] = ")
      INPUT (a[i])
   ENDFOR
ENDFUNCTION

FUNCTION xuatMang (ARRAY a[], INTEGER n, CHAR ten)
   DECLARE INTEGER i
   OUTPUT (ten, ": ")
   FOR (i = 0; i < n; i++)
      OUTPUT (a[i], " ")
   ENDFOR
```

```
    OUTPUT (newline)
ENDFUNCTION

FUNCTION gopMang (ARRAY a[], INTEGER nA, ARRAY b[], POINTER nB)
    DECLARE INTEGER i_a, i_b, j_b
    DECLARE DOUBLE x
    FOR (i_a = 0; i_a < nA; i_a++)
        b[*nB] = a[i_a]
        (*nB) = (*nB) + 1
        FOR (i_b = 1; i_b < *nB; i_b++)
            x = b[i_b]
            j_b = i_b - 1
            WHILE (j_b >= 0 AND b[j_b] > x)
                b[j_b + 1] = b[j_b]
                j_b = j_b - 1
            ENDWHILE
            b[j_b + 1] = x
        ENDFOR
    ENDFOR
ENDFUNCTION

FUNCTION main ()
    DECLARE ARRAY A[MAX], B[MAX]
    DECLARE INTEGER nA, nB
    CALL nhapMang(A, &nA, 'A')
    CALL nhapMang(B, &nB, 'B')
    OUTPUT ("--- MANG BAN DAU ---")
    CALL xuatMang(A, nA, 'A')
    CALL xuatMang(B, nB, 'B')
    CALL gopMang(A, nA, B, &nB)
    OUTPUT ("--- MANG B SAU KHI NOI A ---")
    CALL xuatMang(B, nB, 'B')
    RETURN 0
ENDFUNCTION
```

18.3. Chương trình nguồn

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

```
#define MAX 100

void nhapMang(double a[], int *n, const char ten) {
   int i;
   printf("Nhap so phan tu mang %c: ", ten);
   scanf("%d", n);
   for (i = 0; i < *n; i++) {
      printf("%c[%d] = ", ten, i);
      scanf("%lf", &a[i]);
   }
}

void xuatMang(double a[], int n, const char ten) {
   int i;
   printf("%c: ", ten);
   for (i = 0; i < n; i++)
      printf("%.6lf ", a[i]);
   printf("\n");
}

void gopMang(double a[], int nA, double b[], int *nB) {
   int i_a, i_b, j_b;
   double x;

   for (i_a = 0; i_a < nA; i_a++) {
      b[*nB] = a[i_a];
      (*nB)++;

      for (i_b = 1; i_b < *nB; i_b++) {
         x = b[i_b];
         j_b = i_b - 1;
         while (j_b >= 0 && b[j_b] > x) {
            b[j_b + 1] = b[j_b];
            j_b--;
         }
         b[j_b + 1] = x;
```

```
            }
        }
    }

    int main() {
        double A[MAX], B[MAX];
        int nA, nB;

        nhapMang(A, &nA, 'A');
        nhapMang(B, &nB, 'B');

        printf("\n--- MANG BAN DAU ---\n");
        xuatMang(A, nA, 'A');
        xuatMang(B, nB, 'B');

        gopMang(A, nA, B, &nB);

        printf("\n--- MANG B SAU KHI NOI A ---\n");
        xuatMang(B, nB, 'B');

        return 0;
    }
```

18.4. Kết quả

```
Nhap so phan tu mang A: 5
A[0] = -4
A[1] = 5
A[2] = 2
A[3] = 3
A[4] = 4
Nhap so phan tu mang B: 6
B[0] = 20
B[1] = 8
B[2] = -10
B[3] = 5
B[4] = 4
B[5] = 6

--- MANG BAN DAU ---
A: -4.000000 5.000000 2.000000 3.000000 4.000000
B: 20.000000 8.000000 -10.000000 5.000000 4.000000 6.000000

--- MANG B SAU KHI NOI A (TANG DAN) ---
B: -10.000000 -4.000000 2.000000 3.000000 4.000000 4.000000 5.000000 5.000000 6.000000 8.000000 20.000000
```

Hình 18.1: Kết quả thực thi thành công

18.5. Nhận xét

Chương trình sử dụng kỹ thuật xử lý chuỗi để kiểm tra và chuyển đổi dữ liệu nhập vào, đảm bảo tính hợp lệ của số nguyên. Insertion Sort phù hợp cho mảng nhỏ và dễ cài đặt. Và chương trình sắp xếp trực tiếp trên mảng hiện có, không cần sử dụng mảng phụ

19. Bài 73

19.1. Tên đề bài

Viết chương trình có sử dụng hàm để nhập ma trận A cấp nxn (với n nhập từ bàn phím):

a. Xuất ma trận A ra màn hình

b. Tìm ma trận nghịch đảo của ma trận A

19.2. Thuật toán

19.2.1.Phát biểu thuật toán

- Nhập ma trận vuông A cấp n
- Tính định thức $\det(A)$ bằng phương pháp khử Gauss
  - Nếu pivot = 0, hoán vị hàng và đổi dấu định thức
  - $\det(A) = (-1)^s \cdot \prod_{i=0}^{n-1} a[i][i]$
- Nếu $\det(A) = 0 \rightarrow$ ma trận suy biến, không có nghịch đảo
- Tính ma trận phụ hợp $\text{adj}(A)$
  - Tính ma trận phụ hợp: $\text{adj}(A)_{ij} = (-1)^{i+j} \cdot M_{ji}$
- Tính nghịch đảo: $A^{-1} = \frac{1}{\det(A)} \cdot \text{adj}(A)$

19.2.2.Mã giả (pseudocode C)

```
CONSTANT MAX = 50

FUNCTION la_so_nguyen_duong (STRING s) RETURN INTEGER
   DECLARE INTEGER i = 0
   IF (s[0] == '+') THEN i = 1 ENDIF
   IF (s[0] == '-') THEN RETURN 0 ENDIF
   FOR (k = i; k < LENGTH(s); k++)
      IF (!IsDigit(s[k])) THEN RETURN 0 ENDIF
   ENDFOR
   RETURN 1
ENDFUNCTION

FUNCTION nhap_n () RETURN INTEGER
   WHILE (1)
      OUTPUT ("Nhap cap ma tran vuong n : ")
      INPUT (s)
      IF (la_so_nguyen_duong(s)) THEN
         x = ATOI(s)
         IF (x > 0 && x <= MAX) THEN RETURN x ENDIF
```

```
        ENDIF
        OUTPUT ("Kich thuoc khong hop le! Hay nhap lai.")
    ENDWHILE
ENDFUNCTION

FUNCTION nhapMat (ARRAY a[][MAX], INTEGER n)
    FOR (i = 0; i < n; i++)
        FOR (j = 0; j < n; j++)
            OUTPUT ("A[", i + 1, "][", j + 1, "] = ")
            INPUT (a[i][j])
        ENDFOR
    ENDFOR
ENDFUNCTION

FUNCTION xuatMat (ARRAY a[][MAX], INTEGER n)
    FOR (i = 0; i < n; i++)
        FOR (j = 0; j < n; j++)
            OUTPUT (a[i][j], " ")
        ENDFOR
        OUTPUT (newline)
    ENDFOR
ENDFUNCTION

FUNCTION detGauss (ARRAY A[][MAX], INTEGER n) RETURN REAL
    DECLARE ARRAY a[MAX][MAX]
    FOR (i = 0; i < n; i++)
        FOR (j = 0; j < n; j++)
            a[i][j] = A[i][j]
        ENDFOR
    ENDFOR
    det = 1.0
    FOR (i = 0; i < n; i++)
        IF (ABS(a[i][i]) < 1e-12) THEN
            sw = 0
            FOR (k = i + 1; k < n; k++)
                IF (ABS(a[k][i]) > 1e-12) THEN
                    SWAP_ROWS(a, i, k)
```

```
                det = -det
                sw = 1
                BREAK
            ENDIF
        ENDFOR
        IF (sw == 0) THEN RETURN 0 ENDIF
    ENDIF
    det = det * a[i][i]
    FOR (k = i + 1; k < n; k++)
        ratio = a[k][i] / a[i][i]
        FOR (j = i; j < n; j++)
            a[k][j] = a[k][j] - ratio * a[i][j]
        ENDFOR
    ENDFOR
ENDFOR
RETURN det
ENDFUNCTION

FUNCTION adjoint (ARRAY a[][MAX], ARRAY adj[][MAX], INTEGER n)
    IF (n == 1) THEN
        adj[0][0] = 1
        RETURN
    ENDIF
    DECLARE ARRAY temp[MAX][MAX]
    FOR (i = 0; i < n; i++)
        FOR (j = 0; j < n; j++)
            p = 0; q = 0
            FOR (row = 0; row < n; row++)
                IF (row == i) THEN CONTINUE ENDIF
                FOR (col = 0; col < n; col++)
                    IF (col == j) THEN CONTINUE ENDIF
                    temp[p][q] = a[row][col]
                    q = q + 1
                    IF (q == n - 1) THEN q = 0; p = p + 1 ENDIF
                ENDFOR
            ENDFOR
            sign = ((i + j) % 2 == 0) ? 1.0 : -1.0
```

```
            adj[j][i] = sign * detGauss(temp, n - 1)
        ENDFOR
    ENDFOR
ENDFUNCTION

FUNCTION inverse (ARRAY a[][MAX], ARRAY inv[][MAX], INTEGER n)
RETURN INTEGER
    DECLARE REAL det = detGauss(a, n)
    OUTPUT ("Determinant (det) = ", det)
    IF (ABS(det) < 1e-12) THEN
        OUTPUT ("Ma tran suy bien, khong co nghich dao!")
        RETURN 0
    ENDIF
    DECLARE ARRAY adj[MAX][MAX]
    adjoint(a, adj, n)
    FOR (i = 0; i < n; i++)
        FOR (j = 0; j < n; j++)
            inv[i][j] = adj[i][j] / det
        ENDFOR
    ENDFOR
    RETURN 1
ENDFUNCTION

FUNCTION main ()
    DECLARE ARRAY A[MAX][MAX], INV[MAX][MAX]
    DECLARE INTEGER n
    n = nhap_n()
    OUTPUT ("Nhap ma tran A:")
    nhapMat(A, n)
    OUTPUT ("Ma tran A:")
    xuatMat(A, n)
    OUTPUT ("Ma tran nghich dao A^-1:")
    IF (inverse(A, INV, n) == 1) THEN
        xuatMat(INV, n)
    ENDIF
    RETURN 0
ENDFUNCTION
```

19.3. Chương trình nguồn

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

#define MAX 50

int la_so_nguyen_duong(const char *s) {
   int i = 0;
   if(s[0] == '+') i = 1;
   if(s[0] == '-') return 0;
   for(; s[i] != '\0'; i++)
      if(!isdigit(s[i])) return 0;
   return 1;
}

int nhap_n() {
   char s[100];
   int x;
   while(1) {
      printf("Nhap cap ma tran vuong n : ");
      scanf("%s", s);
      if(la_so_nguyen_duong(s)) {
         x = atoi(s);
         if(x > 0 && x <= MAX) return x;
      }
      printf("Kich thuoc khong hop le! Hay nhap lai.\n");
   }
}

void nhapMat(double a[][MAX], int n){
   for(int i=0; i<n; i++)
      for(int j=0; j<n; j++){
         printf("A[%d][%d] = ", i+1, j+1);
         scanf("%lf", &a[i][j]);
```

```
        }
    }

    void xuatMat(double a[][MAX], int n){
        for(int i=0; i<n; i++){
            for(int j=0; j<n; j++)
                printf("%15.6lf", a[i][j]);
            printf("\n");
        }
    }

    double detGauss(double A[][MAX], int n){
        double a[MAX][MAX];
        for(int i=0; i<n; i++)
            for(int j=0; j<n; j++)
                a[i][j] = A[i][j];
        double det = 1.0;
        for(int i=0; i<n; i++){
            if(fabs(a[i][i]) < 1e-12){
                int sw = 0;
                for(int k=i+1; k<n; k++){
                    if(fabs(a[k][i]) > 1e-12){
                        for(int j=0; j<n; j++){
                            double tmp = a[i][j];
                            a[i][j] = a[k][j];
                            a[k][j] = tmp;
                        }
                        det = -det;
                        sw = 1;
                        break;
                    }
                }
                if(!sw) return 0;
            }
            det *= a[i][i];
            for(int k=i+1; k<n; k++){
                double ratio = a[k][i] / a[i][i];
```

```
            for(int j=i; j<n; j++)
                a[k][j] -= ratio * a[i][j];
        }
    }
    return det;
}

void adjoint(double a[][MAX], double adj[][MAX], int n){
    if(n == 1){
        adj[0][0] = 1;
        return;
    }
    double temp[MAX][MAX];
    for(int i=0; i<n; i++){
        for(int j=0; j<n; j++){
            int p = 0, q = 0;
            for(int row=0; row<n; row++){
                if(row == i) continue;
                for(int col=0; col<n; col++){
                    if(col == j) continue;
                    temp[p][q++] = a[row][col];
                    if(q == n-1){
                        q = 0;
                        p++;
                    }
                }
            }
            double sign = ((i + j) % 2 == 0) ? 1.0 : -1.0;
            adj[j][i] = sign * detGauss(temp, n - 1);
        }
    }
}

int inverse(double a[][MAX], double inv[][MAX], int n){
    double det = detGauss(a, n);
    printf("Determinant (det) = %f\n", det);
    if(fabs(det) < 1e-12){
```

```
        printf("Ma tran suy bien, khong co nghich dao!\n");
        return 0;
    }
    double adj[MAX][MAX];
    adjoint(a, adj, n);
    for(int i=0; i<n; i++)
        for(int j=0; j<n; j++)
            inv[i][j] = adj[i][j] / det;
    return 1;
}

int main(){
    int n;
    double A[MAX][MAX], INV[MAX][MAX];
    n = nhap_n();
    printf("Nhap ma tran A:\n");
    nhapMat(A, n);
    printf("\nMa tran A:\n");
    xuatMat(A, n);
    printf("\nMa tran nghich dao A^-1:\n");
    if(inverse(A, INV, n)){
        xuatMat(INV, n);
    }
    return 0;
}
```

19.4. Kết quả

```
Nhap cap ma tran vuong n : 2
Nhap ma tran A:
A[1][1] = 4
A[1][2] = 7
A[2][1] = 2
A[2][2] = 6

Ma tran A:
       4.000000       7.000000
       2.000000       6.000000

Ma tran nghich dao A^-1:
Determinant (det) = 10.000000
       0.600000      -0.700000
      -0.200000       0.400000
```

Hình 19.1: Kết quả thực thi thành công

19.5. Nhận xét

Bài toán sử dụng phương pháp ma trận phụ hợp (adjoint matrix) để tính nghịch đảo theo công thức $A^{-1} = \frac{1}{\det(A)} \cdot \text{adj}(A)$. Định thức được tính bằng phương pháp khử Gauss có xử lý hoán vị hàng khi gặp pivot bằng 0. Ma trận phụ hợp được tính từ các phần bù đại số (cofactor) của từng phần tử, phải tính n^2 định thức con cấp $n-1$.

20. Bài 74

20.1. Tên đề bài

Viết chương trình có sử dụng hàm để nhập ma trận A cấp mxn (với m, n nhập từ bàn phím):

a. Xuất ma trận A ra màn hình

b. Đưa các phần tử của từng hàng của ma trận lên đường chéo chính và tính tổng các phần tử lớn nhất đó

20.2. Thuật toán

20.2.1.Phát biểu thuật toán

- Nhập số hàng m và số cột n (kiểm tra hợp lệ)
- Nhập ma trận A có $m \times n$ phần tử
- Kiểm tra ma trận vuông ($m = n$), nếu không thì dừng
- Với mỗi hàng i từ 0 đến $n - 1$:
    - Tìm max của hàng i
    - $a[i][i] = \max$
    - $\text{sum} = \text{sum} + \max$
- Xuất ma trận đã thay đổi và tổng

20.2.2.Mã giả (pseudocode C)

```
CONSTANT MAX = 50

FUNCTION la_so_nguyen (STRING s) RETURN INTEGER
   DECLARE INTEGER i = 0
   IF (s[0] == '+' || s[0] == '-') THEN i = 1 ENDIF
   FOR (k = i; k < LENGTH(s); k++)
      IF (!IsDigit(s[k])) THEN RETURN 0 ENDIF
   ENDFOR
   RETURN 1
ENDFUNCTION

FUNCTION nhap_so_nguyen (STRING nhac) RETURN INTEGER
   WHILE (1)
      OUTPUT (nhac)
      INPUT (s)
      IF (la_so_nguyen(s)) THEN
         x = ATOI(s)
         IF (x > 0 && x <= MAX) THEN RETURN x ENDIF
      ENDIF
```

```
            OUTPUT ("Gia tri khong hop le! Nhap lai.")
        ENDWHILE
    ENDFUNCTION

    FUNCTION nhapMat (ARRAY a[][MAX], INTEGER m, INTEGER n)
        FOR (i = 0; i < m; i++)
            FOR (j = 0; j < n; j++)
                OUTPUT ("A[", i + 1, "][", j + 1, "] = ")
                INPUT (a[i][j])
            ENDFOR
        ENDFOR
    ENDFUNCTION

    FUNCTION xuatMat (ARRAY a[][MAX], INTEGER m, INTEGER n)
        FOR (i = 0; i < m; i++)
            FOR (j = 0; j < n; j++)
                OUTPUT (a[i][j], " ")
            ENDFOR
            OUTPUT (newline)
        ENDFOR
    ENDFUNCTION

    FUNCTION timMaxDong (ARRAY a[][MAX], INTEGER n, INTEGER dong)
RETURN REAL
        DECLARE REAL max = a[dong][0]
        FOR (j = 1; j < n; j++)
            IF (a[dong][j] > max) THEN
                max = a[dong][j]
            ENDIF
        ENDFOR
        RETURN max
    ENDFUNCTION

    FUNCTION main ()
        DECLARE ARRAY A[MAX][MAX]
        DECLARE INTEGER m, n
        DECLARE REAL sum, maxx
```

```
    m = nhap_so_nguyen("Nhap m = ")
    n = nhap_so_nguyen("Nhap n = ")
    OUTPUT ("Nhap ma tran A:")
    nhapMat(A, m, n)
    OUTPUT ("Ma tran A vua nhap:")
    xuatMat(A, m, n)
    IF (m != n) THEN
      OUTPUT ("Khong phai ma tran vuong!")
      RETURN 0
    ENDIF
    sum = 0
    FOR (i = 0; i < n; i++)
      maxx = timMaxDong(A, n, i)
      A[i][i] = maxx
      sum = sum + maxx
    ENDFOR
    OUTPUT ("Ma tran sau khi dua max len duong cheo:")
    xuatMat(A, m, n)
    OUTPUT ("Tong cac phan tu duong cheo = ", sum)
    RETURN 0
  ENDFUNCTION
```

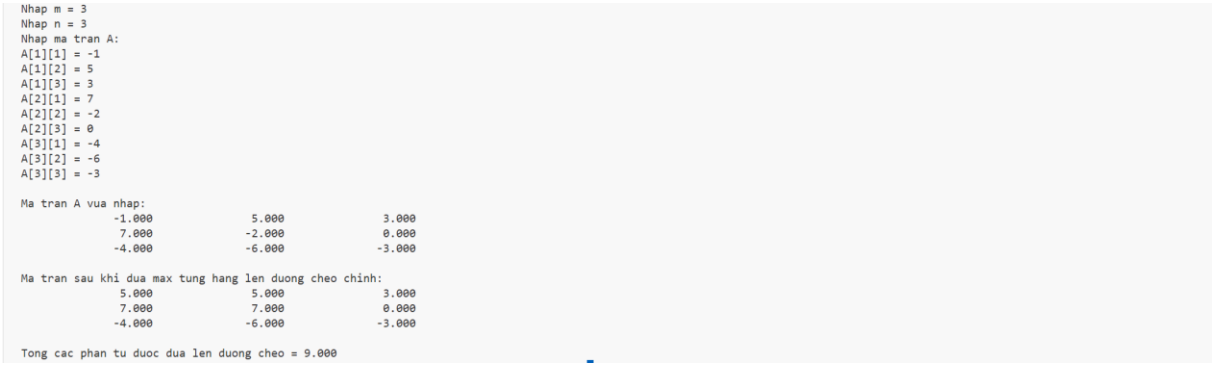
20.3. Chương trình nguồn

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

#define MAX 50

int la_so_nguyen(const char *s) {
  int i = 0;
  if(s[0] == '+' || s[0] == '-') i = 1;
  for(; s[i] != '\0'; i++)
    if(!isdigit(s[i])) return 0;
  return 1;
}
```

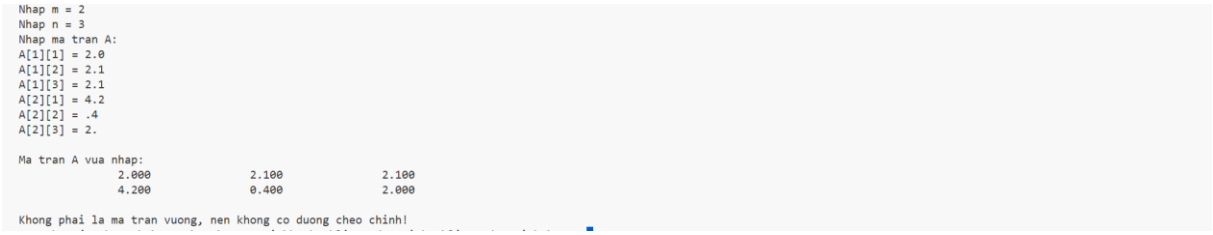
```
int nhap_so_nguyen(const char *nhac) {
   char s[100];
   int x;
   while(1) {
      printf("%s", nhac);
      scanf("%s", s);
      if(la_so_nguyen(s)) {
         x = atoi(s);
         if(x > 0 && x <= MAX) return x;
      }
      printf("Gia tri khong hop le! Nhap lai.\n");
   }
}

void nhapMat(double a[][MAX], int m, int n) {
   int i, j;
   for(i = 0; i < m; i++)
      for(j = 0; j < n; j++) {
         printf("A[%d][%d] = ", i+1, j+1);
         scanf("%lf", &a[i][j]);
      }
}

void xuatMat(double a[][MAX], int m, int n) {
   int i, j;
   for(i = 0; i < m; i++) {
      for(j = 0; j < n; j++)
         printf("%20.3lf", a[i][j]);
      printf("\n");
   }
}

double timMaxDong(double a[][MAX], int n, int dong) {
   int j;
   double max = a[dong][0];
   for(j = 1; j < n; j++)
```

```
            if(a[dong][j] > max)
                max = a[dong][j];
        return max;
    }

    int main() {
        int m, n;
        double A[MAX][MAX];
        double sum;
        int i;
        m = nhap_so_nguyen("Nhap m = ");
        n = nhap_so_nguyen("Nhap n = ");
        printf("Nhap ma tran A:\n");
        nhapMat(A, m, n);
        printf("\nMa tran A vua nhap:\n");
        xuatMat(A, m, n);
        if(m != n) {
            printf("\nKhong phai la ma tran vuong, nen khong co duong cheo chinh!\n");
            return 0;
        }
        sum = 0;
        for(i = 0; i < n; i++) {
            double maxx = timMaxDong(A, n, i);
            A[i][i] = maxx;
            sum += maxx;
        }
        printf("\nMa tran sau khi dua max tung hang len duong cheo chinh:\n");
        xuatMat(A, m, n);
        printf("\nTong cac phan tu duoc dua len duong cheo = %.3lf\n", sum);
        return 0;
    }
```

20.4. Kết quả



Hình 20.1: Kết quả thực thi thành công



Hình 20.2: Kết quả thực thi thành công với ma trận không có đường chéo chính

20.5. Nhận xét

Bài toán kết hợp: tìm max trên mỗi hàng và thao tác với đường chéo chính ma trận vuông. Thuật toán duyệt từng hàng, tìm phần tử lớn nhất cho mỗi hàng, sau đó ghi đè giá trị max lên vị trí $a[i][i]$.

21. Bài 76

21.1. Tên đề bài

Viết chương trình có sử dụng hàm để nhập ma trận A cấp nxn (với n nhập từ bàn phím):

a. Xuất ma trận A ra màn hình

b. Đưa phần tử bé nhất của từng hàng của ma trận lên đường chéo phụ, và tính tổng các phần tử bé nhất đó

21.2. Thuật toán

21.2.1.Phát biểu thuật toán

- Nhập cấp ma trận n (kiểm tra $n > 0$)
- Nhập ma trận vuông A cấp n
- Với mỗi hàng i từ 0 đến $n - 1$:
  - Tìm min của hàng i
  - $a[i][n-1-i] = \min$ (vị trí đường chéo phụ)
  - $\text{sum} = \text{sum} + \min$
- Xuất ma trận đã thay đổi và tổng

21.2.2.Mã giả (pseudocode C)

```
CONSTANT MAX = 100

FUNCTION la_so_nguyen (STRING s) RETURN INTEGER
   DECLARE INTEGER i = 0
   IF (s[0] == '-' || s[0] == '+') THEN i = 1 ENDIF
   FOR (k = i; k < LENGTH(s); k++)
      IF (!IsDigit(s[k])) THEN RETURN 0 ENDIF
   ENDFOR
   RETURN 1
ENDFUNCTION

FUNCTION nhap_so_nguyen (STRING nhac) RETURN INTEGER
   WHILE (1)
      OUTPUT (nhac)
      INPUT (s)
      IF (la_so_nguyen(s)) THEN
         n = ATOI(s)
         IF (n > 0 && n <= MAX) THEN RETURN n ENDIF
      ENDIF
      OUTPUT ("Gia tri khong hop le. Vui long nhap lai!")
   ENDWHILE
ENDFUNCTION

FUNCTION nhapMaTran (ARRAY a[][50], INTEGER n)
   FOR (i = 0; i < n; i++)
      FOR (j = 0; j < n; j++)
         OUTPUT ("A[", i + 1, "][", j + 1, "] = ")
         INPUT (a[i][j])
      ENDFOR
   ENDFOR
ENDFUNCTION

FUNCTION xuatMaTran (ARRAY a[][50], INTEGER n)
   FOR (i = 0; i < n; i++)
      FOR (j = 0; j < n; j++)
         OUTPUT (a[i][j], " ")
```

```
            ENDFOR
            OUTPUT (newline)
        ENDFOR
    ENDFUNCTION

    FUNCTION minHang (ARRAY a[][50], INTEGER n, INTEGER dong)
RETURN REAL
        DECLARE REAL min = a[dong][0]
        FOR (j = 1; j < n; j++)
            IF (a[dong][j] < min) THEN
                min = a[dong][j]
            ENDIF
        ENDFOR
        RETURN min
    ENDFUNCTION

    FUNCTION main ()
        DECLARE ARRAY A[50][50]
        DECLARE INTEGER n
        DECLARE REAL sum, minn
        n = nhap_so_nguyen("Nhap n (n > 0): ")
        OUTPUT ("Nhap ma tran A:")
        nhapMaTran(A, n)
        OUTPUT ("Ma tran A vua nhap:")
        xuatMaTran(A, n)
        sum = 0
        FOR (i = 0; i < n; i++)
            minn = minHang(A, n, i)
            A[i][n - 1 - i] = minn
            sum = sum + minn
        ENDFOR
        OUTPUT ("Ma tran sau khi dua min len duong cheo phu:")
        xuatMaTran(A, n)
        OUTPUT ("Tong cac phan tu duong cheo phu = ", sum)
        RETURN 0
    ENDFUNCTION
```

21.3. Chương trình nguồn

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>
#include <stdlib.h>

#define MAX 100

int la_so_nguyen(const char *s) {
   int i = 0;
   if(s[0] == '-' || s[0] == '+') i = 1;
   for(; s[i] != '\0'; i++) {
      if(!isdigit(s[i])) return 0;
   }
   return 1;
}

int nhap_so_nguyen(const char *nhac) {
   char s[100];
   int n;
   while(1) {
      printf("%s", nhac);
      scanf("%s", s);
      if(la_so_nguyen(s)) {
         n = atoi(s);
         if(n > 0 && n <= MAX) return n;
      }
      printf("Gia tri khong hop le. Vui long nhap lai!\n");
   }
}

void nhapMaTran(double a[][50], int n) {
   for(int i=0;i<n;i++){
      for(int j=0;j<n;j++){
         printf("A[%d][%d] = ", i+1, j+1);
         scanf("%lf", &a[i][j]);
      }
```

```
    }
}

void xuatMaTran(double a[][50], int n){
    for(int i=0;i<n;i++){
        for(int j=0;j<n;j++){
            printf("%15.3f ", a[i][j]);
        }
        printf("\n");
    }
}

double minHang(double a[][50], int n, int dong){
    double min = a[dong][0];
    for(int j=1;j<n;j++){
        if(a[dong][j] < min) min = a[dong][j];
    }
    return min;
}

int main(){
    int n;
    double A[50][50];
    n = nhap_so_nguyen("Nhap n (n > 0): ");
    printf("Nhap ma tran A:\n");
    nhapMaTran(A, n);
    printf("Ma tran A vua nhap:\n");
    xuatMaTran(A, n);
    double sum = 0;
    for(int i=0;i<n;i++){
        double minn = minHang(A, n, i);
        A[i][n-1-i] = minn;
        sum += minn;
    }
    printf("Ma tran sau khi dua min tung hang len duong cheo phu:\n");
    xuatMaTran(A, n);
    printf("Tong cac gia tri nho nhat = %.3f\n", sum);
```

```
    return 0;
}
```

21.4. Kết quả

```
Nhap n (n > 0): 2
Nhap ma tran A:
A[1][1] = 4
A[1][2] = 1
A[2][1] = 6
A[2][2] = 3
Ma tran A vua nhap:
          4.000           1.000
          6.000           3.000
Ma tran sau khi dua phan tu nho nhat tung hang len duong cheo phu:
          4.000           1.000
          3.000           3.000
Tong cac phan tu nho nhat duoc dua len duong cheo phu = 4.000
```

Hình 21.1: Kết quả thực thi thành công

```
Nhap n (n > 0): 2
Nhap ma tran A:
A[1][1] = -1
A[1][2] = 2.0
A[2][1] = 4
A[2][2] = 3
Ma tran A vua nhap:
         -1.000           2.000
          4.000           3.000
Ma tran sau khi dua phan tu nho nhat tung hang len duong cheo phu:
         -1.000          -1.000
          3.000           3.000
Tong cac phan tu nho nhat duoc dua len duong cheo phu = 2.000
```

Hình 21.2: Kết quả thực thi thành công với ma trận đầu vào chứa số thực

21.5. Nhận xét

Bài toán kết hợp: tìm min trên mỗi hàng và thao tác với đường chéo phụ ma trận vuông. Đường chéo phụ gồm các phần tử $a[i][n-1-i]$ với $i = 0,1,\ldots,n-1$, chạy từ góc trên phải xuống góc dưới trái. Thuật toán duyệt từng hàng, tìm phần tử nhỏ nhất cho mỗi hàng, sau đó ghi đè giá trị min lên vị trí đường chéo phụ tương ứng.

22. Bài 79

22.1. Tên đề bài

Viết chương trình có sử dụng hàm để nhập ma trận A cấp nxn (với n nhập từ bàn phím):

a. Xuất ma trận A ra màn hình

b. Đưa phần tử lớn nhất của từng hàng của ma trận lên đường chéo chính và tính tổng các phần tử lớn nhất đó

22.2. Thuật toán

22.2.1.Phát biểu bài toán

- Nhập n (kiểm tra hợp lệ bằng chuỗi)
- Nhập ma trận vuông n×n
- Với mỗi hàng i từ 0 đến n-1:
    - Tìm phần tử lớn nhất trong hàng i
    - Gán giá trị max vào vị trí a[i][i] (đường chéo chính)
    - Cộng dồn max vào tổng
- Xuất ma trận sau khi xử lý và tổng

22.2.2.Mã giả (pseudocode C)

FUNCTION nhap_chuoi (STRING ten, STRING out)

```
    OUTPUT ("Nhap ", ten, ": ")
    INPUT (out)
ENDFUNCTION

FUNCTION la_so_nguyen (STRING s) RETURN INTEGER
    DECLARE INTEGER i
    i = 0
    IF (s[0] == '-' && s[1] != NULL) THEN
        i = 1
    ENDIF
    FOR (k = i; k < LENGTH(s); k++)
        IF (s[k] IS NOT DIGIT) THEN
            RETURN 0
        ENDIF
    ENDFOR
    RETURN 1
ENDFUNCTION

FUNCTION to_ll (STRING s) RETURN LONG_LONG
    DECLARE LONG_LONG x
    DECLARE INTEGER dau, i
    x = 0
    dau = (s[0] == '-') ? -1 : 1
    i = (s[0] == '-') ? 1 : 0
    FOR (k = i; k < LENGTH(s); k++)
        x = x * 10 + (s[k] - '0')
    ENDFOR
    RETURN dau * x
ENDFUNCTION

FUNCTION nhapMaTran (ARRAY a[][], INTEGER n)
    FOR (i = 0; i < n; i++)
        FOR (j = 0; j < n; j++)
            OUTPUT ("A[", i+1, "][", j+1, "] = ")
            INPUT (a[i][j])
        ENDFOR
    ENDFOR
```

```
ENDFUNCTION

FUNCTION xuatMaTran (ARRAY a[][], INTEGER n)
  FOR (i = 0; i < n; i++)
    FOR (j = 0; j < n; j++)
      OUTPUT (a[i][j], " ")
    ENDFOR
    OUTPUT (newline)
  ENDFOR
ENDFUNCTION

FUNCTION timMaxDong (ARRAY a[][], INTEGER n, INTEGER dong)
RETURN DOUBLE
  DECLARE DOUBLE max
  max = a[dong][0]
  FOR (j = 1; j < n; j++)
    IF (a[dong][j] > max) THEN
      max = a[dong][j]
    ENDIF
  ENDFOR
  RETURN max
ENDFUNCTION

FUNCTION main ()
  DECLARE INTEGER n
  DECLARE STRING buf
  n = 0
  WHILE (1)
    nhap_chuoi("n (n > 0)", buf)
    IF (la_so_nguyen(buf)) THEN
      n = to_ll(buf)
      IF (n > 0 && n <= 50) THEN
        BREAK
      ENDIF
    ENDIF
    OUTPUT ("n phai la so nguyen duong. Nhap lai!")
  ENDWHILE
```

```
    DECLARE ARRAY A[50][50]
    OUTPUT ("Nhap ma tran A:")
    nhapMaTran(A, n)
    OUTPUT ("Ma tran A vua nhap:")
    xuatMaTran(A, n)
    DECLARE DOUBLE sum
    sum = 0
    FOR (i = 0; i < n; i++)
      DECLARE DOUBLE maxVal
      maxVal = timMaxDong(A, n, i)
      A[i][i] = maxVal
      sum = sum + maxVal
    ENDFOR
    OUTPUT ("Ma tran sau khi dua max tung hang len duong cheo chinh:")
    xuatMaTran(A, n)
    OUTPUT ("Tong cac phan tu lon nhat duoc dua len duong cheo = ", sum)
    RETURN 0
ENDFUNCTION
```

22.3. Chương trình nguồn

```
#include <stdio.h>
#include <ctype.h>
#include <string.h>

void nhap_chuoi(const char *ten, char *out) {
    printf("Nhap %s: ", ten);
    scanf("%s", out);
}

int la_so_nguyen(const char *s) {
    int i = 0;
    if (s[0] == '-' && s[1] != '\0') i = 1;
    for (; s[i]; i++)
        if (!isdigit(s[i])) return 0;
    return 1;
}

long long to_ll(const char *s) {
```

```
    long long x = 0;
    int dau = (s[0] == '-') ? -1 : 1;
    int i = (s[0] == '-') ? 1 : 0;
    for (; s[i]; i++)
        x = x * 10 + (s[i] - '0');
    return dau * x;
}

void nhapMaTran(double a[][50], int n) {
    for (int i = 0; i < n; i++)
        for (int j = 0; j < n; j++) {
            printf("A[%d][%d] = ", i + 1, j + 1);
            scanf("%lf", &a[i][j]);
        }
}

void xuatMaTran(double a[][50], int n) {
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++)
            printf("%15.3lf ", a[i][j]);
        printf("\n");
    }
}

double timMaxDong(double a[][50], int n, int dong) {
    double max = a[dong][0];
    for (int j = 1; j < n; j++)
        if (a[dong][j] > max)
            max = a[dong][j];
    return max;
}

int main() {
    int n = 0;
    char buf[100];
    while (1) {
        nhap_chuoi("n (n > 0)", buf);
```

```
            if (la_so_nguyen(buf)) {
                n = (int)to_ll(buf);
                if (n > 0 && n <= 50) break;
            }
            printf("n phai la so nguyen duong. Nhap lai!\n");
        }
        double A[50][50];
        printf("Nhap ma tran A:\n");
        nhapMaTran(A, n);
        printf("\nMa tran A vua nhap:\n");
        xuatMaTran(A, n);
        double sum = 0;
        for (int i = 0; i < n; i++) {
            double maxVal = timMaxDong(A, n, i);
            A[i][i] = maxVal;
            sum += maxVal;
        }
        printf("\nMa tran sau khi dua max tung hang len duong cheo chinh:\n");
        xuatMaTran(A, n);
        printf("\nTong cac phan tu lon nhat duoc dua len duong cheo = %.3lf\n", sum);
        return 0;
    }
```

22.4. Kết quả

```
Nhap n (n > 0): 2
Nhap ma tran A:
A[1][1] = 2
A[1][2] = 9
A[2][1] = 3
A[2][2] = 1

Ma tran A vua nhap:
          2.000           9.000
          3.000           1.000

Ma tran sau khi dua max tung hang len duong cheo chinh:
          9.000           9.000
          3.000           3.000

Tong cac phan tu lon nhat duoc dua len duong cheo = 12.000
```

Hình 22.1: Kết quả thực thi thành công

```
Nhap n (n > 0): 2
Nhap ma tran A:
A[1][1] = 2.0
A[1][2] = -9
A[2][1] = 3
A[2][2] = 1

Ma tran A vua nhap:
          2.000          -9.000
          3.000           1.000

Ma tran sau khi dua max tung hang len duong cheo chinh:
          2.000          -9.000
          3.000           3.000

Tong cac phan tu lon nhat duoc dua len duong cheo = 5.000
```

Hình 22.2: Kết quả thực thi thành công với ma trận đầu vào chứa số thực hoặc số âm

22.5. Nhận xét

Bài toán yêu cầu với mỗi hàng i, tìm phần tử lớn nhất và gán vào vị trí $a[i][i]$ (đường chéo chính). Thuật toán sử dụng kỹ thuật tìm max đơn giản cho từng hàng. Chương trình sử dụng kiểu double để hỗ trợ số thực và có kiểm tra đầu vào hợp lệ bằng chuỗi ký tự.

23. Bài 82

23.1. Tên đề bài

Tạo một ma trận xoắn có dạng như sau:

a. Xoắn từ ngoài vào trong

b. Xoắn từ trong ra ngoài.

23.2. Thuật toán

23.2.1.Phát biểu thuật toán

- Khởi tạo 4 biên: top=0, bottom=n-1, left=0, right=m-1
- Xoắn ốc từ ngoài vào:
    - Lặp khi còn ô trống:
        - Đi phải trên hàng top, tăng top
        - Đi xuống trên cột right, giảm right
        - Đi trái trên hàng bottom, giảm bottom
        - Đi lên trên cột left, tăng left
- Xoắn ốc từ ngoài vào:
    - Bắt đầu từ tâm ma trận
    - Mở rộng dần ra 4 hướng: phải, xuống, trái, lên

23.2.2.Mã giả (pseudocode C)

```
FUNCTION nhap_chuoi (STRING ten, STRING out)
   OUTPUT ("Nhap ", ten, ": ")
   INPUT (out)
ENDFUNCTION

FUNCTION la_so_nguyen (STRING s) RETURN INTEGER
   DECLARE INTEGER i
   i = 0
   IF (s[0] == '-' && s[1] != NULL) THEN
      i = 1
   ENDIF
   FOR (k = i; k < LENGTH(s); k++)
      IF (s[k] IS NOT DIGIT) THEN
         RETURN 0
```

```
        ENDIF
    ENDFOR
    RETURN 1
ENDFUNCTION

FUNCTION to_ll (STRING s) RETURN LONG_LONG
    DECLARE LONG_LONG x
    x = 0
    FOR (k = 0; k < LENGTH(s); k++)
        x = x * 10 + (s[k] - '0')
    ENDFOR
    RETURN x
ENDFUNCTION

FUNCTION xuatMat (ARRAY a[][], INTEGER n, INTEGER m)
    FOR (i = 0; i < n; i++)
        FOR (j = 0; j < m; j++)
            OUTPUT (a[i][j], " ")
        ENDFOR
        OUTPUT (newline)
    ENDFOR
ENDFUNCTION

FUNCTION xoanNgoaiVao (ARRAY a[][], INTEGER n, INTEGER m)
    DECLARE INTEGER top, bottom, left, right, num
    top = 0
    bottom = n - 1
    left = 0
    right = m - 1
    num = 1
    WHILE (top <= bottom && left <= right)
        FOR (i = left; i <= right; i++)
            a[top][i] = num
            num = num + 1
        ENDFOR
        top = top + 1
        FOR (i = top; i <= bottom; i++)
```

```
            a[i][right] = num
            num = num + 1
         ENDFOR
         right = right - 1
         IF (top <= bottom) THEN
            FOR (i = right; i >= left; i--)
               a[bottom][i] = num
               num = num + 1
            ENDFOR
            bottom = bottom - 1
         ENDIF
         IF (left <= right) THEN
            FOR (i = bottom; i >= top; i--)
               a[i][left] = num
               num = num + 1
            ENDFOR
            left = left + 1
         ENDIF
      ENDWHILE
   ENDFUNCTION

   FUNCTION xoanTrongRaNgoai (ARRAY a[][], INTEGER n, INTEGER m)
      DECLARE INTEGER top, bottom, left, right, num
      top = 0
      bottom = n - 1
      left = 0
      right = m - 1
      num = n * m
      WHILE (top <= bottom && left <= right)
         FOR (i = left; i <= right; i++)
            a[top][i] = num
            num = num - 1
         ENDFOR
         top = top + 1
         FOR (i = top; i <= bottom; i++)
            a[i][right] = num
            num = num - 1
```

```
        ENDFOR
        right = right - 1
        IF (top <= bottom) THEN
            FOR (i = right; i >= left; i--)
                a[bottom][i] = num
                num = num - 1
            ENDFOR
            bottom = bottom - 1
        ENDIF
        IF (left <= right) THEN
            FOR (i = bottom; i >= top; i--)
                a[i][left] = num
                num = num - 1
            ENDFOR
            left = left + 1
        ENDIF
    ENDWHILE
ENDFUNCTION

FUNCTION main ()
    DECLARE INTEGER n, m
    DECLARE STRING buf
    n = 0
    m = 0
    WHILE (1)
        nhap_chuoi("so dong n (n > 0)", buf)
        IF (la_so_nguyen(buf)) THEN
            n = to_ll(buf)
            IF (n > 0 && n <= 100) THEN
                BREAK
            ENDIF
        ENDIF
        OUTPUT ("n phai la so nguyen duong. Nhap lai!")
    ENDWHILE
    WHILE (1)
        nhap_chuoi("so cot m (m > 0)", buf)
        IF (la_so_nguyen(buf)) THEN
```

```
            m = to_ll(buf)
            IF (m > 0 && m <= 100) THEN
               BREAK
            ENDIF
         ENDIF
         OUTPUT ("m phai la so nguyen duong. Nhap lai!")
      ENDWHILE
      DECLARE ARRAY a[100][100]
      OUTPUT ("Ma tran xoan oc tu NGOAI VAO TRONG:")
      xoanNgoaiVao(a, n, m)
      xuatMat(a, n, m)
      OUTPUT ("Ma tran xoan oc tu TRONG RA NGOAI:")
      xoanTrongRaNgoai(a, n, m)
      xuatMat(a, n, m)
      RETURN 0
   ENDFUNCTION
```

23.3. Chương trình nguồn

```
#include <stdio.h>
#include <ctype.h>
#include <string.h>

void nhap_chuoi(const char *ten, char *out) {
   printf("Nhap %s: ", ten);
   scanf("%s", out);
}

int la_so_nguyen(const char *s) {
   int i = 0;
   if (s[0] == '-' && s[1] != '\0') i = 1;
   for (; s[i]; i++)
      if (!isdigit(s[i])) return 0;
   return 1;
}

long long to_ll(const char *s) {
   long long x = 0;
   int dau = (s[0] == '-') ? -1 : 1;
```

```
    int i = (s[0] == '-') ? 1 : 0;
    for (; s[i]; i++)
        x = x * 10 + (s[i] - '0');
    return dau * x;
}

void xuatMat(int a[][100], int n, int m) {
    int i, j;
    for (i = 0; i < n; i++) {
        for (j = 0; j < m; j++)
            printf("%5d ", a[i][j]);
        printf("\n");
    }
}

void xoanNgoaiVao(int a[][100], int n, int m) {
    int top = 0, bottom = n - 1;
    int left = 0, right = m - 1;
    int num = 1;
    while (top <= bottom && left <= right) {
        for (int i = left; i <= right; i++)
            a[top][i] = num++;
        top++;
        for (int i = top; i <= bottom; i++)
            a[i][right] = num++;
        right--;
        if (top <= bottom) {
            for (int i = right; i >= left; i--)
                a[bottom][i] = num++;
            bottom--;
        }
        if (left <= right) {
            for (int i = bottom; i >= top; i--)
                a[i][left] = num++;
            left++;
        }
    }
```

```
}

void xoanTrongRaNgoai(int a[][100], int n, int m) {
    int centerRow = (n - 1) / 2;
    int centerCol = (m - 1) / 2;
    int top = centerRow, bottom = centerRow;
    int left = centerCol, right = centerCol;
    int num = 1;
    a[centerRow][centerCol] = num++;
    while (num <= n * m) {
        right++;
        if (right < m) {
            for (int i = top; i <= bottom && right < m; i++)
                if (num <= n * m) a[i][right] = num++;
        }
        bottom++;
        if (bottom < n) {
            for (int i = right; i >= left && bottom < n; i--)
                if (num <= n * m) a[bottom][i] = num++;
        }
        left--;
        if (left >= 0) {
            for (int i = bottom; i >= top && left >= 0; i--)
                if (num <= n * m) a[i][left] = num++;
        }
        top--;
        if (top >= 0) {
            for (int i = left; i <= right && top >= 0; i++)
                if (num <= n * m) a[top][i] = num++;
        }
    }
}

int main() {
    int n = 0, m = 0;
    char buf[100];
    while (1) {
```

```
        nhap_chuoi("so dong n (n > 0)", buf);
        if (la_so_nguyen(buf)) {
            n = (int)to_ll(buf);
            if (n > 0 && n <= 100) break;
        }
        printf("n phai la so nguyen duong. Nhap lai!\n");
    }
    while (1) {
        nhap_chuoi("so cot m (m > 0)", buf);
        if (la_so_nguyen(buf)) {
            m = (int)to_ll(buf);
            if (m > 0 && m <= 100) break;
        }
        printf("m phai la so nguyen duong. Nhap lai!\n");
    }
    int a[100][100];
    printf("\nMa tran xoan oc tu NGOAI VAO TRONG:\n");
    xoanNgoaiVao(a, n, m);
    xuatMat(a, n, m);
    printf("\nMa tran xoan oc tu TRONG RA NGOAI:\n");
    xoanTrongRaNgoai(a, n, m);
    xuatMat(a, n, m);
    return 0;
}
```

23.4. Kết quả

```
Nhap so dong n: 6
Nhap so cot m: 7

--- Ma tran xoan tu ngoai vao trong ---
   1    2    3    4    5    6    7
  22   23   24   25   26   27    8
  21   36   37   38   39   28    9
  20   35   42   41   40   29   10
  19   34   33   32   31   30   11
  18   17   16   15   14   13   12

--- Ma tran xoan tu trong ra ngoai ---
  42   41   40   39   38   37   36
  21   20   19   18   17   16   35
  22    7    6    5    4   15   34
  23    8    1    2    3   14   33
  24    9   10   11   12   13   32
  25   26   27   28   29   30   31
```

Hình 23.1: Kết quả thực thi thành công

23.5. Nhận xét

Bài toán tạo ma trận xoắn ốc sử dụng Four Boundaries (4 biên) để điền số theo hình xoắn ốc. Sử dụng 4 biến biên (top, bottom, left, right) để thu hẹp dần vùng cần điền. Mỗi vòng xoắn gồm 4 bước: đi phải, đi xuống, đi trái, đi lên. Chương trình có thêm kiểm tra đầu vào hợp lệ bằng chuỗi ký tự.

24. Bài 89

24.1. Tên đề bài

Tính toán số lớn.

Viết chương trình xử lý tính toán các số lớn. Yêu cầu chỉ thực hiện hai phép toán cộng và trừ, sử dụng chuỗi ký tự để biểu diễn số lớn.

24.2. Thuật toán

24.2.1.Phát biểu thuật toán

- Tách A và B thành dấu, phần nguyên, phần thập phân
- Cân bằng độ dài phần thập phân
- Nếu cùng dấu: cộng trị tuyệt đối, giữ nguyên dấu
- Nếu khác dấu: trừ trị tuyệt đối, lấy dấu số lớn hơn
- A - B = A + (-B)
- Đổi dấu B rồi áp dụng quy tắc cộng

24.2.2.Mã giả (pseudocode C)

```
FUNCTION xoaSo0Dau (STRING s)
   DECLARE INTEGER i
   i = 0
   WHILE (s[i] == '0' && s[i+1] != NULL)
      i = i + 1
   ENDWHILE
   IF (i > 0) THEN
      MOVE s[i..end] TO s[0..end-i]
   ENDIF
ENDFUNCTION

FUNCTION xoaSo0Cuoi (STRING s)
   DECLARE INTEGER len
   len = LENGTH(s)
   WHILE (len > 1 && s[len-1] == '0')
      len = len - 1
   ENDWHILE
   s[len] = NULL
ENDFUNCTION

FUNCTION soSanhAbs (STRING aInt, STRING aFrac, STRING bInt, STRING
bFrac) RETURN INTEGER
   IF (LENGTH(aInt) != LENGTH(bInt)) THEN
```

```
        RETURN (LENGTH(aInt) > LENGTH(bInt)) ? 1 : -1
    ENDIF
    cmp = STRCMP(aInt, bInt)
    IF (cmp != 0) THEN
        RETURN (cmp > 0) ? 1 : -1
    ENDIF
    la = LENGTH(aFrac)
    lb = LENGTH(bFrac)
    L = MAX(la, lb)
    FOR (i = 0; i < L; i++)
        ca = (i < la) ? aFrac[i] : '0'
        cb = (i < lb) ? bFrac[i] : '0'
        IF (ca != cb) THEN
            RETURN (ca > cb) ? 1 : -1
        ENDIF
    ENDFOR
    RETURN 0
ENDFUNCTION
```

FUNCTION congAbs (STRING aInt, STRING aFrac, STRING bInt, STRING bFrac, STRING resInt, STRING resFrac)

```
    la = LENGTH(aFrac)
    lb = LENGTH(bFrac)
    L = MAX(la, lb)
    A = aFrac; B = bFrac
    WHILE (LENGTH(A) < L) A = A + '0' ENDWHILE
    WHILE (LENGTH(B) < L) B = B + '0' ENDWHILE
    carry = 0
    resFrac[L] = NULL
    FOR (i = L - 1; i >= 0; i--)
        sum = (A[i] - '0') + (B[i] - '0') + carry
        resFrac[i] = (sum % 10) + '0'
        carry = sum / 10
    ENDFOR
    ia = LENGTH(aInt); ib = LENGTH(bInt)
    imax = MAX(ia, ib)
    resInt[imax] = NULL
```

```
    k = imax - 1; ia = ia - 1; ib = ib - 1
    WHILE (imax > 0)
      da = (ia >= 0) ? aInt[ia] - '0' : 0; ia = ia - 1
      db = (ib >= 0) ? bInt[ib] - '0' : 0; ib = ib - 1
      sum = da + db + carry
      resInt[k] = (sum % 10) + '0'; k = k - 1
      carry = sum / 10
      imax = imax - 1
    ENDWHILE
    IF (carry) THEN
      MOVE resInt TO resInt + 1
      resInt[0] = '1'
    ENDIF
    xoaSo0Cuoi(resFrac)
    xoaSo0Dau(resInt)
  ENDFUNCTION

  FUNCTION truAbs (STRING aInt, STRING aFrac, STRING bInt, STRING
bFrac, STRING resInt, STRING resFrac)
    la = LENGTH(aFrac)
    lb = LENGTH(bFrac)
    L = MAX(la, lb)
    A = aFrac; B = bFrac
    WHILE (LENGTH(A) < L) A = A + '0' ENDWHILE
    WHILE (LENGTH(B) < L) B = B + '0' ENDWHILE
    borrow = 0
    resFrac[L] = NULL
    FOR (i = L - 1; i >= 0; i--)
      diff = (A[i] - '0') - (B[i] - '0') - borrow
      IF (diff < 0) THEN
        diff = diff + 10
        borrow = 1
      ELSE
        borrow = 0
      ENDIF
      resFrac[i] = diff + '0'
    ENDFOR
```

```
    ia = LENGTH(aInt); ib = LENGTH(bInt)
    imax = MAX(ia, ib)
    resInt[imax] = NULL
    k = imax - 1; ia = ia - 1; ib = ib - 1
    WHILE (imax > 0)
      da = (ia >= 0) ? aInt[ia] - '0' : 0; ia = ia - 1
      db = (ib >= 0) ? bInt[ib] - '0' : 0; ib = ib - 1
      diff = da - db - borrow
      IF (diff < 0) THEN
        diff = diff + 10
        borrow = 1
      ELSE
        borrow = 0
      ENDIF
      resInt[k] = diff + '0'; k = k - 1
      imax = imax - 1
    ENDWHILE
    xoaSo0Cuoi(resFrac)
    xoaSo0Dau(resInt)
  ENDFUNCTION

  FUNCTION tachSo (STRING num, INTEGER am, STRING nguyen, STRING
thapPhan)
    am = (num[0] == '-') ? 1 : 0
    p = num + (am ? 1 : 0)
    dot = FIND('.', p)
    IF (dot != NULL) THEN
      COPY p[0..dot-1] TO nguyen
      COPY p[dot+1..end] TO thapPhan
    ELSE
      COPY p TO nguyen
      thapPhan = "0"
    ENDIF
    xoaSo0Dau(nguyen)
    xoaSo0Cuoi(thapPhan)
  ENDFUNCTION
```

```
FUNCTION main ()
   DECLARE STRING A, B
   DECLARE INTEGER signA, signB
   DECLARE STRING intA, fracA, intB, fracB
   INPUT (A)
   INPUT (B)
   tachSo(A, signA, intA, fracA)
   tachSo(B, signB, intB, fracB)
   IF (signA == signB) THEN
      rSign = signA
      congAbs(intA, fracA, intB, fracB, rInt, rFrac)
   ELSE
      cmp = soSanhAbs(intA, fracA, intB, fracB)
      IF (cmp == 0) THEN
         rSign = 1; rInt = "0"; rFrac = "0"
      ELSE IF (cmp > 0) THEN
         rSign = signA
         truAbs(intA, fracA, intB, fracB, rInt, rFrac)
      ELSE
         rSign = signB
         truAbs(intB, fracB, intA, fracA, rInt, rFrac)
      ENDIF
   ENDIF
   OUTPUT ("A + B = ", rSign, rInt, ".", rFrac)
   OUTPUT ("A - B = ", ...)
   RETURN 0
ENDFUNCTION
```

24.3. Chương trình nguồn

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

#define MAX 2000

void xoaSo0Dau(char *s) {
   int i = 0;
   while (s[i] == '0' && s[i + 1] != '\0') i++;
```

```
    if (i > 0) memmove(s, s + i, strlen(s) - i + 1);
}

void xoaSo0Cuoi(char *s) {
    int len = strlen(s);
    while (len > 1 && s[len - 1] == '0') len--;
    s[len] = '\0';
}

void themSo0Truoc(char *s, int cnt) {
    int len = strlen(s);
    memmove(s + cnt, s, len + 1);
    for (int i = 0; i < cnt; i++) s[i] = '0';
}

void themSo0Sau(char *s, int cnt) {
    int len = strlen(s);
    for (int i = 0; i < cnt; i++) s[len + i] = '0';
    s[len + cnt] = '\0';
}

int sosanh(const char *a, const char *b) {
    int lenA = strlen(a), lenB = strlen(b);
    if (lenA != lenB) return lenA > lenB ? 1 : -1;
    return strcmp(a, b);
}

void congChuoi(const char *a, const char *b, char *c) {
    int lenA = strlen(a), lenB = strlen(b);
    int maxLen = lenA > lenB ? lenA : lenB;
    int carry = 0, k = 0;
    for (int i = lenA - 1, j = lenB - 1; i >= 0 || j >= 0 || carry; i--, j--) {
        int dA = (i >= 0) ? a[i] - '0' : 0;
        int dB = (j >= 0) ? b[j] - '0' : 0;
        int sum = dA + dB + carry;
        carry = sum / 10;
        c[k++] = (sum % 10) + '0';
```

```
    }
    c[k] = '\0';
    for (int i = 0; i < k / 2; i++) {
        char t = c[i]; c[i] = c[k - 1 - i]; c[k - 1 - i] = t;
    }
    xoaSo0Dau(c);
}

void truChuoi(const char *a, const char *b, char *c) {
    int lenA = strlen(a), lenB = strlen(b);
    int borrow = 0, k = 0;
    for (int i = lenA - 1, j = lenB - 1; i >= 0; i--, j--) {
        int dA = a[i] - '0';
        int dB = (j >= 0) ? b[j] - '0' : 0;
        int diff = dA - dB - borrow;
        if (diff < 0) { diff += 10; borrow = 1; }
        else borrow = 0;
        c[k++] = diff + '0';
    }
    c[k] = '\0';
    for (int i = 0; i < k / 2; i++) {
        char t = c[i]; c[i] = c[k - 1 - i]; c[k - 1 - i] = t;
    }
    xoaSo0Dau(c);
}

void tachSo(const char *num, int *sign, char *intPart, char *fracPart) {
    int i = 0;
    if (num[0] == '-') { *sign = -1; i = 1; }
    else if (num[0] == '+') { *sign = 1; i = 1; }
    else *sign = 1;
    const char *dot = strchr(num + i, '.');
    if (dot) {
        int lenInt = dot - (num + i);
        strncpy(intPart, num + i, lenInt);
        intPart[lenInt] = '\0';
        strcpy(fracPart, dot + 1);
```

```
    } else {
        strcpy(intPart, num + i);
        fracPart[0] = '0';
        fracPart[1] = '\0';
    }
    xoaSo0Dau(intPart);
    xoaSo0Cuoi(fracPart);
    if (fracPart[0] == '\0') { fracPart[0] = '0'; fracPart[1] = '\0'; }
}

void canBangThapPhan(char *f1, char *f2) {
    int len1 = strlen(f1), len2 = strlen(f2);
    if (len1 < len2) themSo0Sau(f1, len2 - len1);
    else if (len2 < len1) themSo0Sau(f2, len1 - len2);
}

void congThapPhan(const char *f1, const char *f2, char *resF, int *carryOut) {
    int len = strlen(f1);
    int carry = 0;
    for (int i = len - 1; i >= 0; i--) {
        int sum = (f1[i] - '0') + (f2[i] - '0') + carry;
        carry = sum / 10;
        resF[i] = (sum % 10) + '0';
    }
    resF[len] = '\0';
    *carryOut = carry;
}

void truThapPhan(const char *f1, const char *f2, char *resF, int *borrowOut) {
    int len = strlen(f1);
    int borrow = 0;
    for (int i = len - 1; i >= 0; i--) {
        int diff = (f1[i] - '0') - (f2[i] - '0') - borrow;
        if (diff < 0) { diff += 10; borrow = 1; }
        else borrow = 0;
        resF[i] = diff + '0';
    }
```

```
    resF[len] = '\0';
    *borrowOut = borrow;
}

int soSanhTuyetDoi(const char *i1, const char *f1, const char *i2, const char *f2) {
    int cmpInt = sosanh(i1, i2);
    if (cmpInt != 0) return cmpInt;
    return strcmp(f1, f2);
}

int main() {
    char A[MAX], B[MAX];
    int signA, signB;
    char intA[MAX], fracA[MAX];
    char intB[MAX], fracB[MAX];
    printf("Nhap so A: "); scanf("%s", A);
    printf("Nhap so B: "); scanf("%s", B);
    tachSo(A, &signA, intA, fracA);
    tachSo(B, &signB, intB, fracB);
    canBangThapPhan(fracA, fracB);
    char rInt[MAX], rFrac[MAX];
    int rSign;
    printf("\n===== PHEP CONG A + B =====\n");
    if (signA == signB) {
        rSign = signA;
        int carry;
        congThapPhan(fracA, fracB, rFrac, &carry);
        congChuoi(intA, intB, rInt);
        if (carry) {
            char one[2] = "1";
            char temp[MAX];
            congChuoi(rInt, one, temp);
            strcpy(rInt, temp);
        }
    } else {
        int cmp = soSanhTuyetDoi(intA, fracA, intB, fracB);
```

```
            if (cmp == 0) {
                rSign = 1; strcpy(rInt, "0"); strcpy(rFrac, "0");
            } else {
                const char *bigInt, *bigFrac, *smallInt, *smallFrac;
                if (cmp > 0) {
                    bigInt = intA; bigFrac = fracA;
                    smallInt = intB; smallFrac = fracB;
                    rSign = signA;
                } else {
                    bigInt = intB; bigFrac = fracB;
                    smallInt = intA; smallFrac = fracA;
                    rSign = signB;
                }
                int borrow;
                truThapPhan(bigFrac, smallFrac, rFrac, &borrow);
                char bigIntCopy[MAX];
                strcpy(bigIntCopy, bigInt);
                if (borrow) {
                    char one[2] = "1";
                    char temp[MAX];
                    truChuoi(bigIntCopy, one, temp);
                    strcpy(bigIntCopy, temp);
                }
                truChuoi(bigIntCopy, smallInt, rInt);
            }
        }
        xoaSo0Dau(rInt);
        xoaSo0Cuoi(rFrac);
        if (rFrac[0] == '\0') { rFrac[0] = '0'; rFrac[1] = '\0'; }
        printf("A + B = ");
        if (rSign == -1 && !(strcmp(rInt, "0") == 0 && strcmp(rFrac, "0") == 0))
printf("-");
        printf("%s", rInt);
        if (strcmp(rFrac, "0") != 0) printf(".%s", rFrac);
        printf("\n");
        char rSubInt[MAX], rSubFrac[MAX];
        int rSubSign;
```

```
        printf("\n===== PHEP TRU A - B =====\n");
        int signBNeg = -signB;
        if (signA == signBNeg) {
            rSubSign = signA;
            int carry;
            congThapPhan(fracA, fracB, rSubFrac, &carry);
            congChuoi(intA, intB, rSubInt);
            if (carry) {
                char one[2] = "1";
                char temp[MAX];
                congChuoi(rSubInt, one, temp);
                strcpy(rSubInt, temp);
            }
        } else {
            int cmp = soSanhTuyetDoi(intA, fracA, intB, fracB);
            if (cmp == 0) {
                rSubSign = 1; strcpy(rSubInt, "0"); strcpy(rSubFrac, "0");
            } else {
                const char *bigInt, *bigFrac, *smallInt, *smallFrac;
                if (cmp > 0) {
                    bigInt = intA; bigFrac = fracA;
                    smallInt = intB; smallFrac = fracB;
                    rSubSign = signA;
                } else {
                    bigInt = intB; bigFrac = fracB;
                    smallInt = intA; smallFrac = fracA;
                    rSubSign = signBNeg;
                }
                int borrow;
                truThapPhan(bigFrac, smallFrac, rSubFrac, &borrow);
                char bigIntCopy[MAX];
                strcpy(bigIntCopy, bigInt);
                if (borrow) {
                    char one[2] = "1";
                    char temp[MAX];
                    truChuoi(bigIntCopy, one, temp);
                    strcpy(bigIntCopy, temp);
```

```
            }
            truChuoi(bigIntCopy, smallInt, rSubInt);
        }
    }
    xoaSo0Dau(rSubInt);
    xoaSo0Cuoi(rSubFrac);
    if (rSubFrac[0] == '\0') { rSubFrac[0] = '0'; rSubFrac[1] = '\0'; }
    printf("A - B = ");
    if (rSubSign == -1 && !(strcmp(rSubInt, "0") == 0 && strcmp(rSubFrac, "0")
== 0)) printf("-");
    printf("%s", rSubInt);
    if (strcmp(rSubFrac, "0") != 0) printf(".%s", rSubFrac);
    printf("\n");
    return 0;
}
```

24.4. Kết quả

```
Nhap so thu nhat: 12456321.23123
Nhap so thu hai: 213123.123

--- PHEP CONG A + B ---
12669444.35423

--- PHEP TRU A - B ---
12243198.10823
```

Hình 24.1: Kết quả thực thi thành công

24.5. Nhận xét

Bài toán cộng/trừ hai số thực lớn sử dụng kỹ thuật Big Number Arithmetic với chuỗi ký tự. Thuật toán chia số thành phần nguyên và phần thập phân, xử lý riêng rồi ghép kết quả. Xử lý dấu: cùng dấu thì cộng trị tuyệt đối, khác dấu thì trừ trị tuyệt đối và lấy dấu của số có trị tuyệt đối lớn hơn.

25. Bài 90

25.1. Tên đề bài

Xử lý tập hợp.

Viết chương trình xử lý các phép toán trên tập hợp, gồm: phép hợp, phép giao, phép hiệu của hai tập hợp và phép kiểm tra tập con. Yêu cầu sử dụng kiểu dữ liệu mảng để cài đặt kiểu tập hợp.

25.2. Thuật toán

25.2.1.Phát biểu bài toán

- Nhập hai tập hợp chuỗi A và B
- Loại bỏ trùng lặp
- Thực hiện các phép toán
    - $A \cup B$: Thêm tất cả A, sau đó thêm phần tử B nếu chưa có trong kết quả

- $A \cap B$: Thêm phần tử A nếu nó cũng có trong B
- $A \setminus B$: Thêm phần tử thuộc A nhưng không thuộc B
- $A \subseteq B$: Kiểm tra mọi phần tử A đều có trong B

- Xuất kết quả đã tính toán

25.2.2.Mã giả (pseudocode C)

```
FUNCTION loaiBoTrungLap (ARRAY a[][], INTEGER n) RETURN INTEGER
    DECLARE ARRAY b[MAX][MAXLEN]
    DECLARE INTEGER m
    m = 0
    FOR (i = 0; i < n; i++)
      DECLARE INTEGER tonTai
      tonTai = 0
      FOR (j = 0; j < m; j++)
        IF (STRCMP(a[i], b[j]) == 0) THEN
          tonTai = 1
          BREAK
        ENDIF
      ENDFOR
      IF (tonTai == 0) THEN
        STRCPY(b[m], a[i])
        m = m + 1
      ENDIF
    ENDFOR
    FOR (i = 0; i < m; i++)
      STRCPY(a[i], b[i])
    ENDFOR
    RETURN m
  ENDFUNCTION

  FUNCTION hop (ARRAY a[][], INTEGER nA, ARRAY b[][], INTEGER nB, ARRAY kq[][]) RETURN INTEGER
    m = 0
    FOR (i = 0; i < nA; i++)
      STRCPY(kq[m], a[i])
      m = m + 1
    ENDFOR
```

```
        FOR (i = 0; i < nB; i++)
            tonTai = 0
            FOR (j = 0; j < m; j++)
                IF (STRCMP(b[i], kq[j]) == 0) THEN
                    tonTai = 1
                    BREAK
                ENDIF
            ENDFOR
            IF (tonTai == 0) THEN
                STRCPY(kq[m], b[i])
                m = m + 1
            ENDIF
        ENDFOR
        RETURN m
    ENDFUNCTION

    FUNCTION giao (ARRAY a[][], INTEGER nA, ARRAY b[][], INTEGER nB,
ARRAY kq[][]) RETURN INTEGER
        m = 0
        FOR (i = 0; i < nA; i++)
            FOR (j = 0; j < nB; j++)
                IF (STRCMP(a[i], b[j]) == 0) THEN
                    STRCPY(kq[m], a[i])
                    m = m + 1
                    BREAK
                ENDIF
            ENDFOR
        ENDFOR
        RETURN m
    ENDFUNCTION

    FUNCTION hieu (ARRAY a[][], INTEGER nA, ARRAY b[][], INTEGER nB,
ARRAY kq[][]) RETURN INTEGER
        m = 0
        FOR (i = 0; i < nA; i++)
            tonTai = 0
            FOR (j = 0; j < nB; j++)
```

```
            IF (STRCMP(a[i], b[j]) == 0) THEN
               tonTai = 1
               BREAK
            ENDIF
         ENDFOR
         IF (tonTai == 0) THEN
            STRCPY(kq[m], a[i])
            m = m + 1
         ENDIF
      ENDFOR
      RETURN m
   ENDFUNCTION

   FUNCTION tapCon (ARRAY a[][], INTEGER nA, ARRAY b[][], INTEGER
nB) RETURN INTEGER
      FOR (i = 0; i < nA; i++)
         timThay = 0
         FOR (j = 0; j < nB; j++)
            IF (STRCMP(a[i], b[j]) == 0) THEN
               timThay = 1
               BREAK
            ENDIF
         ENDFOR
         IF (timThay == 0) THEN
            RETURN 0
         ENDIF
      ENDFOR
      RETURN 1
   ENDFUNCTION

   FUNCTION inTap (STRING ten, ARRAY a[][], INTEGER n)
      OUTPUT (ten, " = { ")
      FOR (i = 0; i < n; i++)
         OUTPUT ("\"", a[i], "\"")
         IF (i < n - 1) THEN
            OUTPUT (", ")
         ENDIF
```

```
        ENDFOR
        OUTPUT (" }")
    ENDFUNCTION

    FUNCTION main ()
        DECLARE ARRAY a[MAX][MAXLEN], b[MAX][MAXLEN],
kq[MAX][MAXLEN]
        DECLARE INTEGER nA, nB, nKQ
        OUTPUT ("Nhap so phan tu tap A: ")
        INPUT (nA)
        FOR (i = 0; i < nA; i++)
            INPUT (a[i])
        ENDFOR
        OUTPUT ("Nhap so phan tu tap B: ")
        INPUT (nB)
        FOR (i = 0; i < nB; i++)
            INPUT (b[i])
        ENDFOR
        nA = loaiBoTrungLap(a, nA)
        nB = loaiBoTrungLap(b, nB)
        inTap("Tap A", a, nA)
        inTap("Tap B", b, nB)
        nKQ = hop(a, nA, b, nB, kq)
        inTap("Hop A ∪ B", kq, nKQ)
        nKQ = giao(a, nA, b, nB, kq)
        inTap("Giao A ∩ B", kq, nKQ)
        nKQ = hieu(a, nA, b, nB, kq)
        inTap("Hieu A \\ B", kq, nKQ)
        IF (tapCon(a, nA, b, nB) == 1) THEN
            OUTPUT ("A la tap con cua B")
        ELSE
            OUTPUT ("A khong la tap con cua B")
        ENDIF
        RETURN 0
    ENDFUNCTION
```

25.3. Chương trình nguồn

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

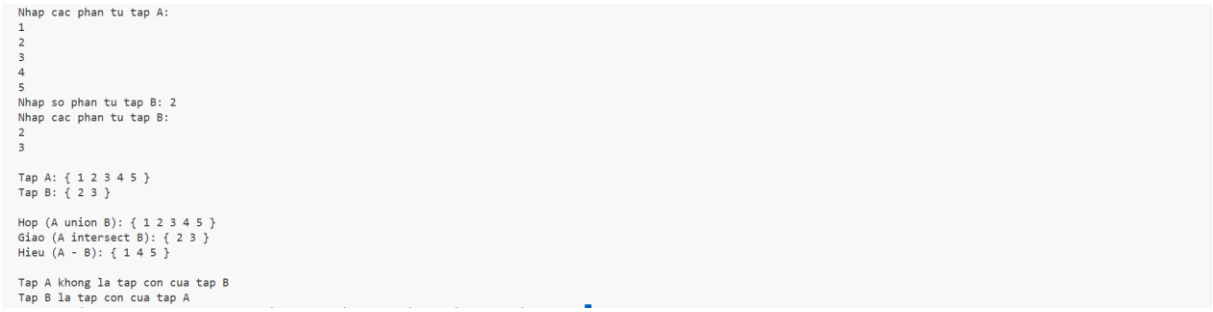
```
#include <string.h>

#define MAX 1000
#define MAXLEN 100

int loaiBoTrungLap(char a[][MAXLEN], int n) {
    char b[MAX][MAXLEN];
    int m = 0;
    for (int i = 0; i < n; i++) {
        int tonTai = 0;
        for (int j = 0; j < m; j++) {
            if (strcmp(a[i], b[j]) == 0) {
                tonTai = 1;
                break;
            }
        }
        if (!tonTai) {
            strcpy(b[m], a[i]);
            m++;
        }
    }
    for (int i = 0; i < m; i++)
        strcpy(a[i], b[i]);
    return m;
}

int timKiem(char a[][MAXLEN], int n, const char *x) {
    for (int i = 0; i < n; i++)
        if (strcmp(a[i], x) == 0)
            return 1;
    return 0;
}

void hopTapHop(char a[][MAXLEN], int nA, char b[][MAXLEN], int nB, char
c[][MAXLEN], int *nC) {
    *nC = 0;
    for (int i = 0; i < nA; i++) {
```

```
            strcpy(c[*nC], a[i]);
            (*nC)++;
        }
        for (int i = 0; i < nB; i++) {
            if (!timKiem(a, nA, b[i])) {
                strcpy(c[*nC], b[i]);
                (*nC)++;
            }
        }
    }

    void giaoTapHop(char a[][MAXLEN], int nA, char b[][MAXLEN], int nB, char
c[][MAXLEN], int *nC) {
        *nC = 0;
        for (int i = 0; i < nA; i++) {
            if (timKiem(b, nB, a[i])) {
                strcpy(c[*nC], a[i]);
                (*nC)++;
            }
        }
    }

    void hieuTapHop(char a[][MAXLEN], int nA, char b[][MAXLEN], int nB, char
c[][MAXLEN], int *nC) {
        *nC = 0;
        for (int i = 0; i < nA; i++) {
            if (!timKiem(b, nB, a[i])) {
                strcpy(c[*nC], a[i]);
                (*nC)++;
            }
        }
    }
    int tapCon(char a[][MAXLEN], int nA, char b[][MAXLEN], int nB) {
        for (int i = 0; i < nA; i++)
            if (!timKiem(b, nB, a[i]))
                return 0;
        return 1;
```

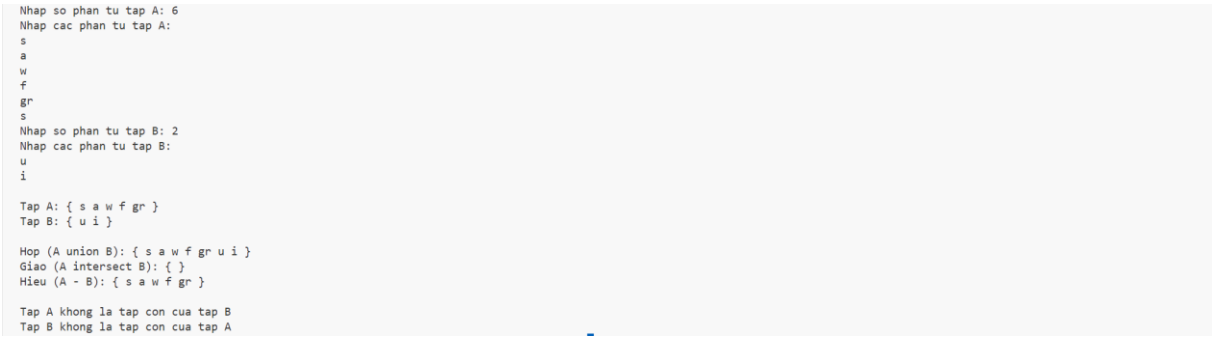
```
}
void xuatTapHop(const char *ten, char a[][MAXLEN], int n) {
    printf("%s = { ", ten);
    for (int i = 0; i < n; i++) {
        printf("\"%s\"", a[i]);
        if (i < n - 1) printf(", ");
    }
    printf(" }\n");
}
int main() {
    char a[MAX][MAXLEN], b[MAX][MAXLEN], c[MAX][MAXLEN];
    int nA, nB, nC;
    printf("Nhap so phan tu tap A: ");
    scanf("%d", &nA);
    printf("Nhap cac phan tu tap A:\n");
    for (int i = 0; i < nA; i++) {
        printf("A[%d] = ", i);
        scanf("%s", a[i]);
    }
    printf("Nhap so phan tu tap B: ");
    scanf("%d", &nB);
    printf("Nhap cac phan tu tap B:\n");
    for (int i = 0; i < nB; i++) {
        printf("B[%d] = ", i);
        scanf("%s", b[i]);
    }
    nA = loaiBoTrungLap(a, nA);
    nB = loaiBoTrungLap(b, nB);
    printf("\n");
    xuatTapHop("Tap A", a, nA);
    xuatTapHop("Tap B", b, nB);
    printf("\n--- Cac phep toan tap hop ---\n");
    hopTapHop(a, nA, b, nB, c, &nC);
    xuatTapHop("A hop B", c, nC);
    giaoTapHop(a, nA, b, nB, c, &nC);
    xuatTapHop("A giao B", c, nC);
    hieuTapHop(a, nA, b, nB, c, &nC);
```

```
    xuatTapHop("A \\ B", c, nC);
    hieuTapHop(b, nB, a, nA, c, &nC);
    xuatTapHop("B \\ A", c, nC);
    printf("\n--- Kiem tra tap con ---\n");
    if (tapCon(a, nA, b, nB))
        printf("A la tap con cua B\n");
    else
        printf("A KHONG la tap con cua B\n");
    if (tapCon(b, nB, a, nA))
        printf("B la tap con cua A\n");
    else
        printf("B KHONG la tap con cua A\n");
    return 0;
}
```

25.4. Kết quả



Hình 25.1: Kết quả thực thi thành công



Hình 25.2: Kết quả thực thi thành công với mảng đầu vào là chuỗi ký tự

25.5. Nhận xét

Bài toán các phép toán trên tập hợp áp dụng kỹ thuật Linear Search để kiểm tra phần tử có tồn tại trong tập hợp. Tập hợp được biểu diễn bằng mảng không có phần tử trùng lặp. Các phép toán cơ bản bao gồm: hợp $A \cup B$ (tất cả phần tử thuộc A hoặc B), giao $A \cap B$ (phần tử thuộc cả A và B), hiệu $A \setminus B$ (phần tử thuộc A nhưng không thuộc B), và kiểm tra tập con $A \subseteq B$ (mọi phần tử của A đều thuộc B).

TÀI LIỆU THAM KHẢO

[1] https://www.researchgate.net/figure/Pseudocode-of-fault-tolerant-routing-logic-used-by-nodes-C-style-comments-and-block_fig4_3300397

[2] https://www.researchgate.net/figure/The-C-Like-Pseudocode-of-schedule_fig1_311336291

[3] https://www.researchgate.net/figure/The-C-style-pseudo-code-for-showing-how-N-n-n-K-P_fig1_220763968