

TRƯỜNG ĐẠI HỌC ĐÀ LẠT
KHOA TOÁN - TIN HỌC
∞  ∞

TRẦN NGỌC ANH

THỰC HÀNH NHẬP MÔN LẬP TRÌNH
(Bài Giảng Tóm Tắt)



-- Lưu hành nội bộ --

∞ Đà Lạt 2008 ∞

LỜI MỞ ĐẦU

Học phần này nhằm cung cấp cho sinh viên các kỹ năng gỡ rối, sửa lỗi trên Visual C⁺⁺, bổ sung thêm một số kiến thức về chuỗi, con trỏ và cung cấp một số lượng tương đối lớn các bài tập nhằm giúp sinh viên học tốt học phần “Nhập môn lập trình”.

Nội dung gồm 6 mục:

Mục 1: Hướng dẫn viết và chạy chương trình (CT) bằng VC⁺⁺ 6.0.

Mục 2: Hướng dẫn sửa một số lỗi / cảnh báo thường gặp.

Mục 3: Kỹ thuật chạy Debug để gỡ rối CT.

Mục 4: Một kỹ thuật kiểm chứng tự động CT trên các bộ dữ liệu được sinh ngẫu nhiên.

Mục 5: Con trỏ.

Mục 6: Tìm hiểu một số hàm xử lý chuỗi trong thư viện string.h.

Bài tập.

Đà lạt, 5/2008

Tác giả

MỤC LỤC

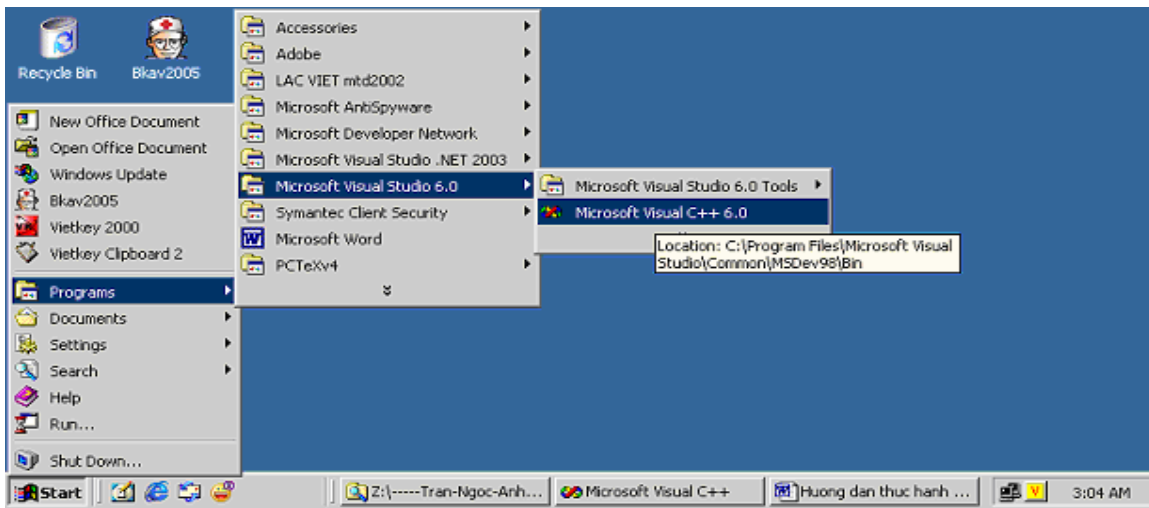
Mục 1: Hướng dẫn viết và chạy chương trình (CT) bằng VC ⁺⁺ 6.0.	1
1.1 Chạy Visual Studio C++ 6.0.	1
1.2 Mở dự án Win32 Console Application.	1
1.3 Viết chương trình (CT).	4
1.4 Chạy và kiểm tra tính đúng của CT	5
Mục 2: Hướng dẫn sửa một số lỗi / cảnh báo thường gặp.....	8
Mục 3: Kỹ thuật chạy Debug để gỡ rối CT.	12
3.1 Xét CT xuất ra bảng mã ASCII của 256 ký tự.	12
3.2 Xét CT đổi số sang hệ 16.....	15
3.3 Xét CT đổi số từ hệ 10 sang hệ b dùng hàm.....	18
Mục 4: Một kỹ thuật kiểm chứng tự động CT trên các bộ dữ liệu được sinh ngẫu nhiên.....	22
Mục 5: Con trỏ.....	24
5.1 Định nghĩa, khai báo, khởi tạo và sử dụng con trỏ.....	24
5.2 Cấp phát và thu hồi vùng nhớ bằng con trỏ.....	25
5.3 Toán tử tăng / giảm trên biến con trỏ	25
5.4 Một ứng dụng con trỏ để hoán vị giá trị hai biến	26
Mục 6: Tìm hiểu một số hàm xử lý chuỗi trong thư viện string.h	27
Bài tập.....	28

Mục 1: Hướng dẫn viết và chạy chương trình (CT) bằng VC⁺⁺ 6.0

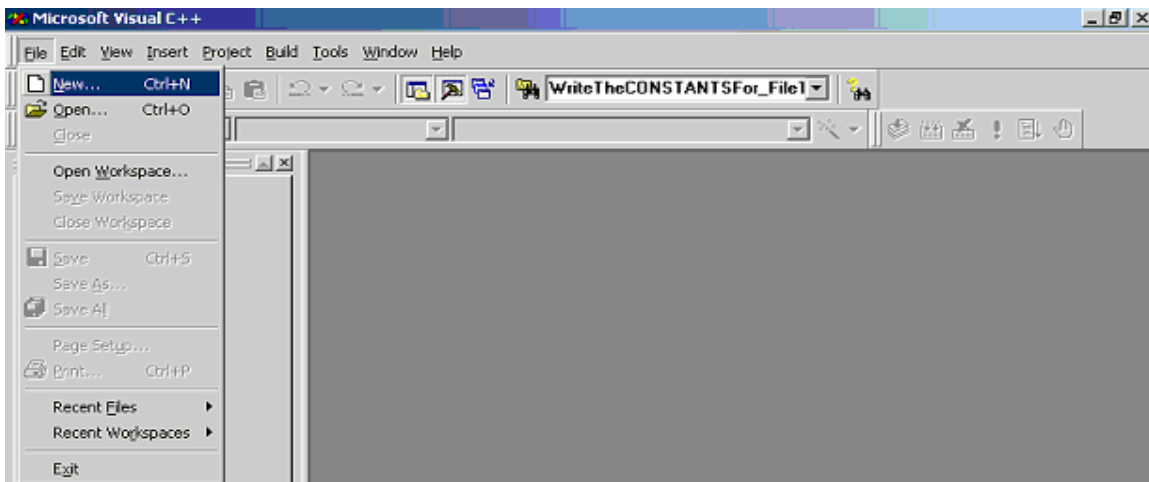
Để thực hành “Nhập môn lập trình” trên Visual C⁺⁺ 6.0, sinh viên cần thực hiện các bước:

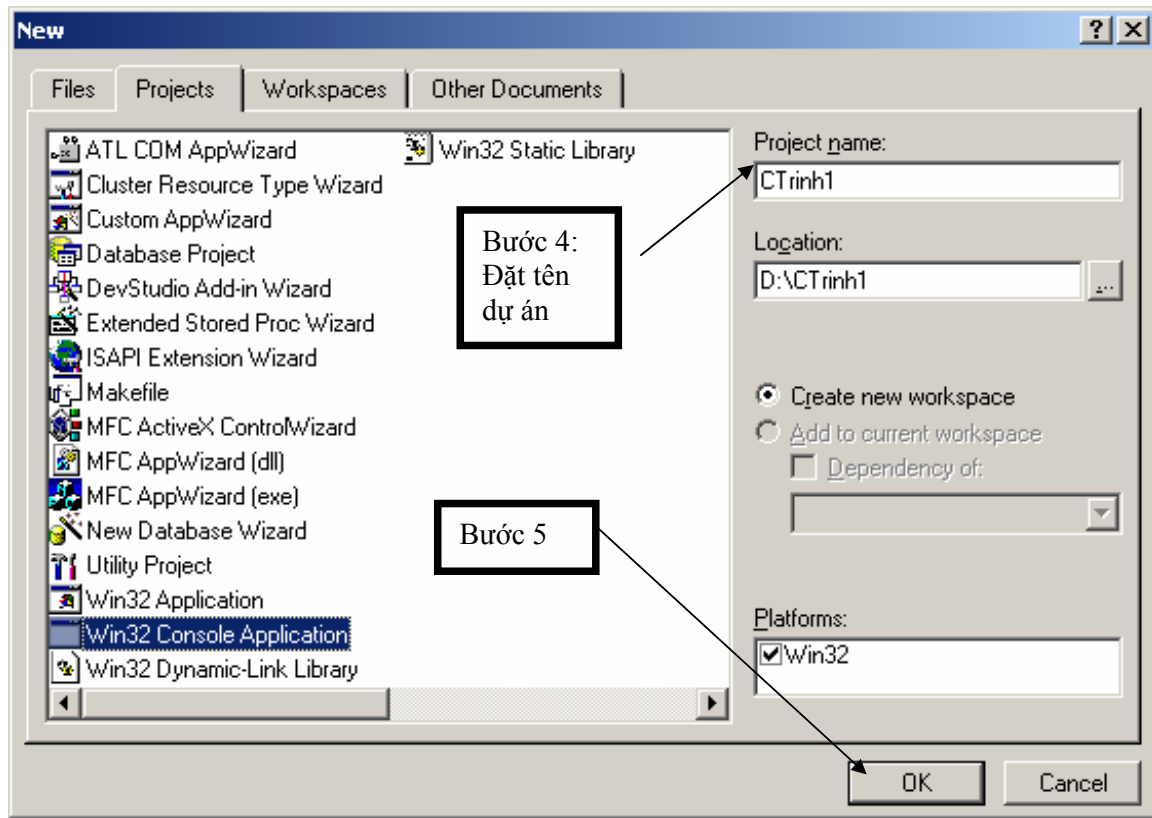
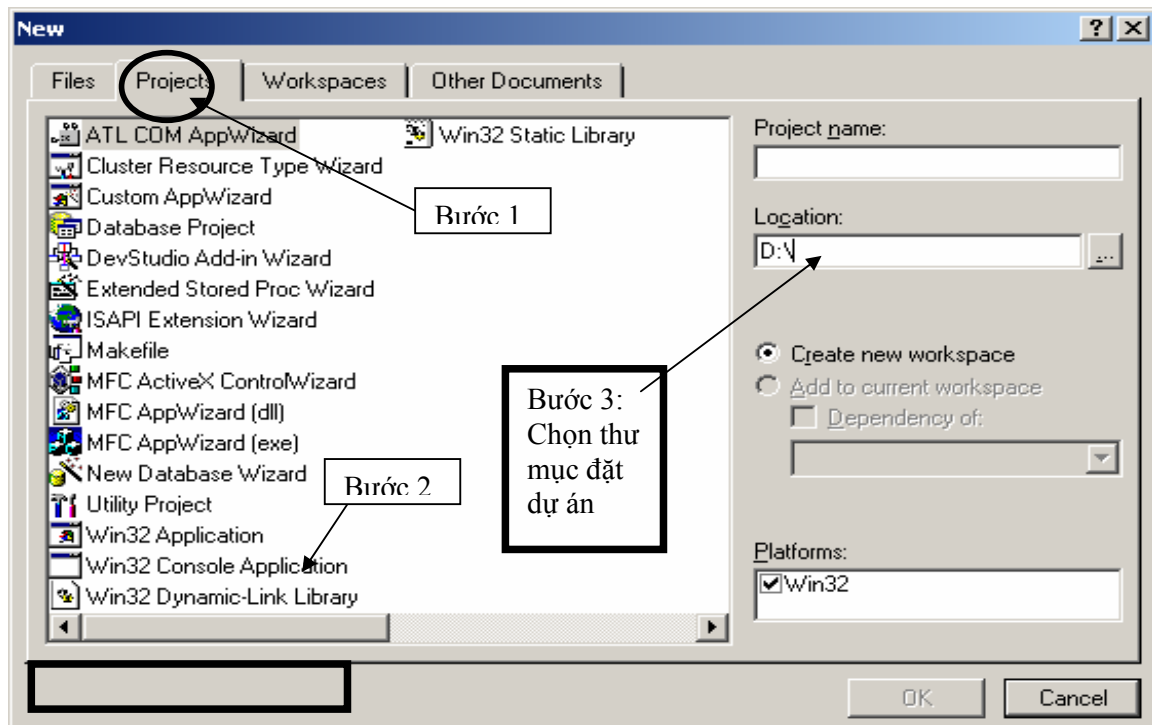
- 1) Chạy Visual Studio C⁺⁺ 6.0.
- 2) Mở dự án Win32 Console Application.
- 3) Viết chương trình (CT).
- 4) Chạy và kiểm tra tính đúng của CT:
 - a. Nếu CT có lỗi cú pháp, quay lại (3) để sửa lỗi cú pháp.
 - b. Nếu CT cho ra kết quả không đúng (biểu diễn dữ liệu và thuật toán) mong muốn, quay lại (3).
 - c. Nếu CT cho ra kết quả đúng, tiếp tục chạy thử CT trên các bộ dữ liệu khác (ứng với nhiều trường hợp khác nhau của bài toán).

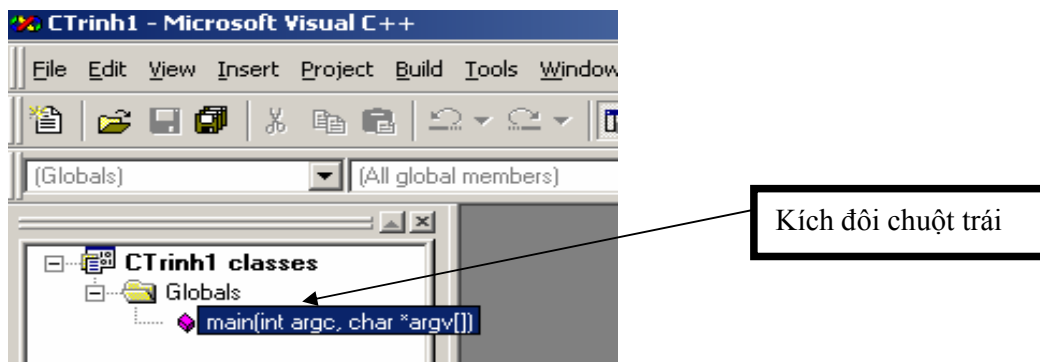
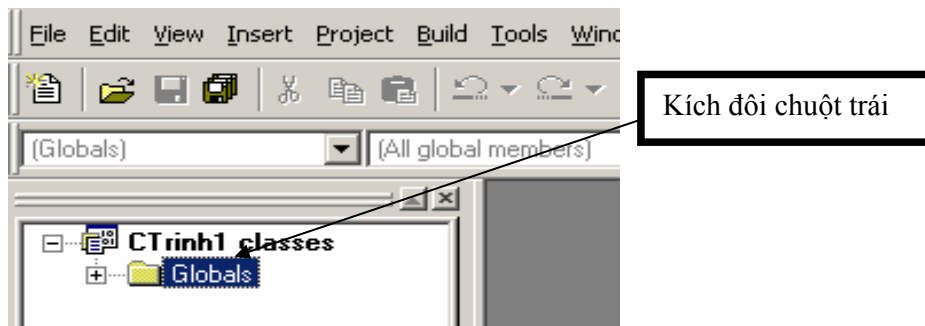
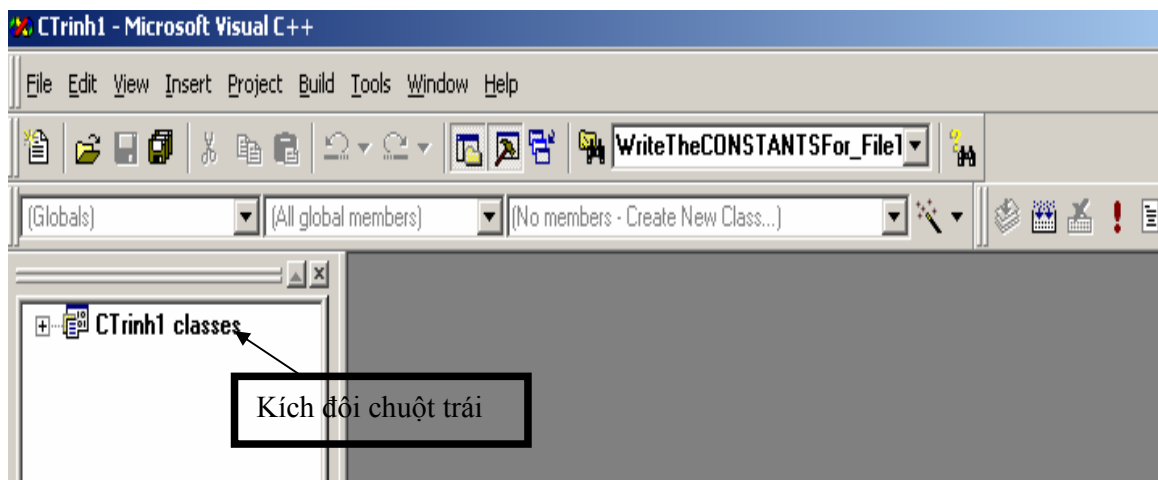
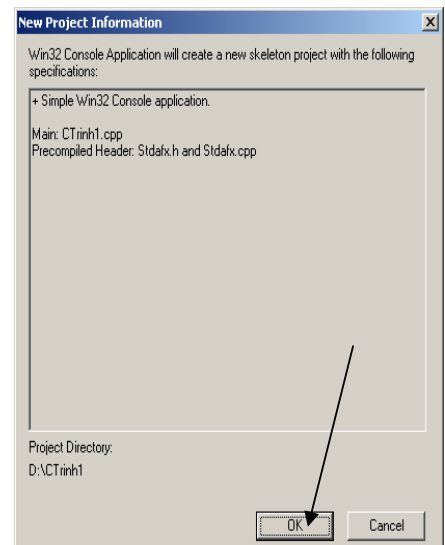
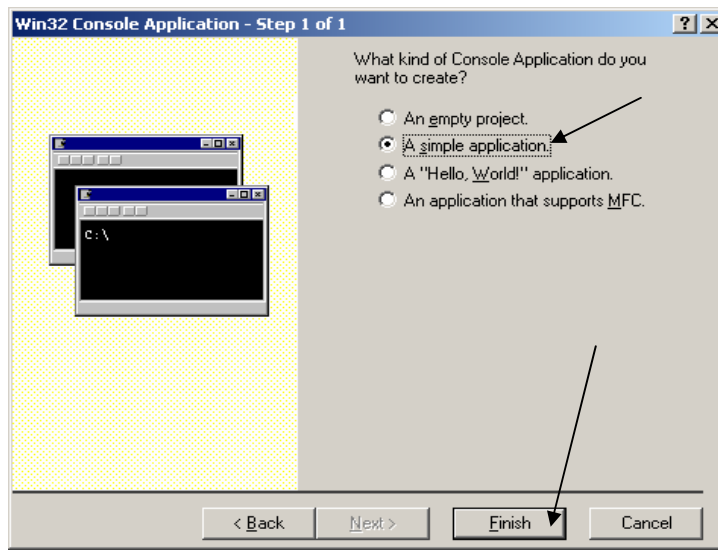
1.1: Chạy Visual C⁺⁺ 6.0

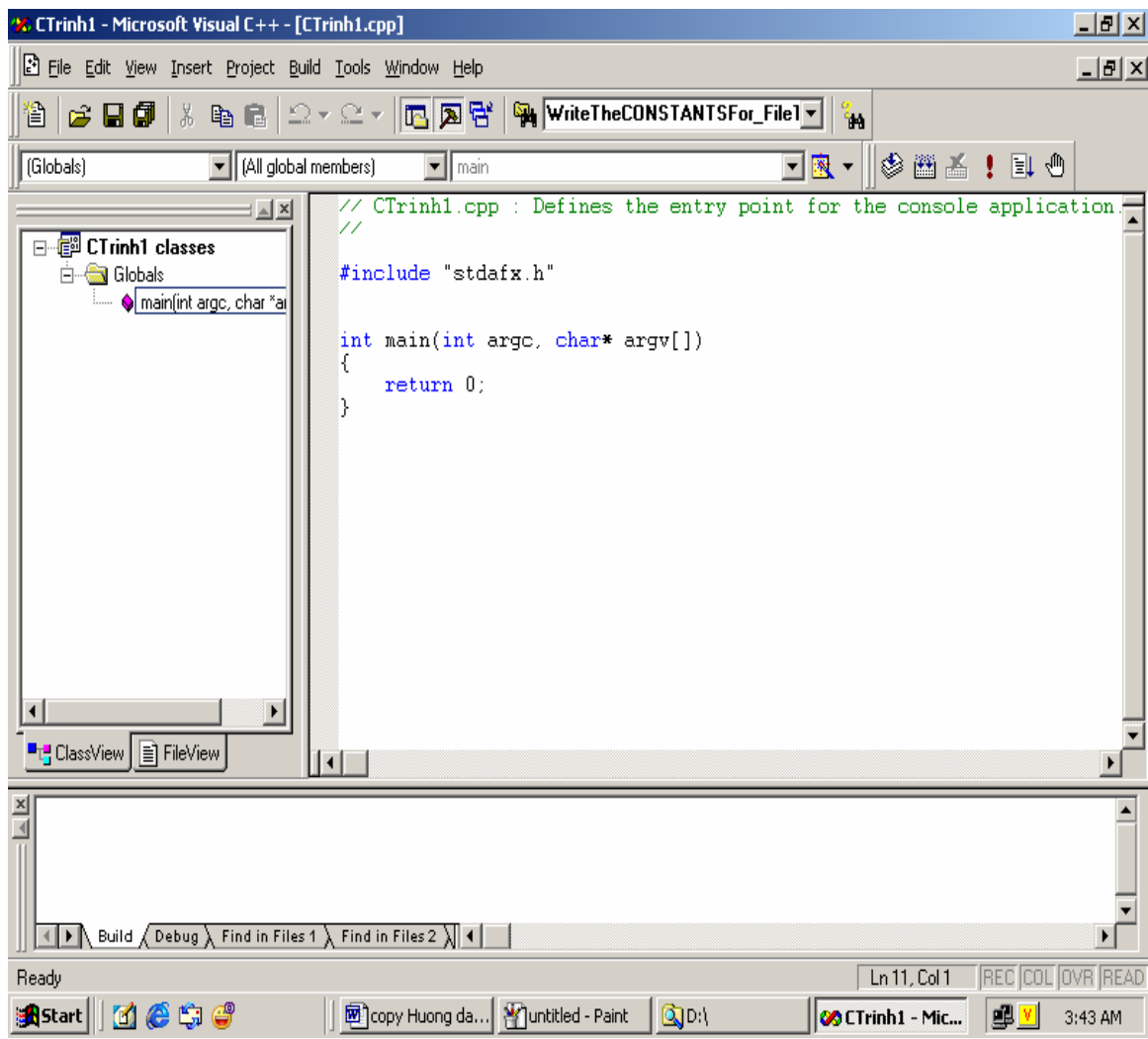


1.2: Mở dự án Win32 Console Application: File \ New \ Projects \ Win32 Console Application \ A simple application:



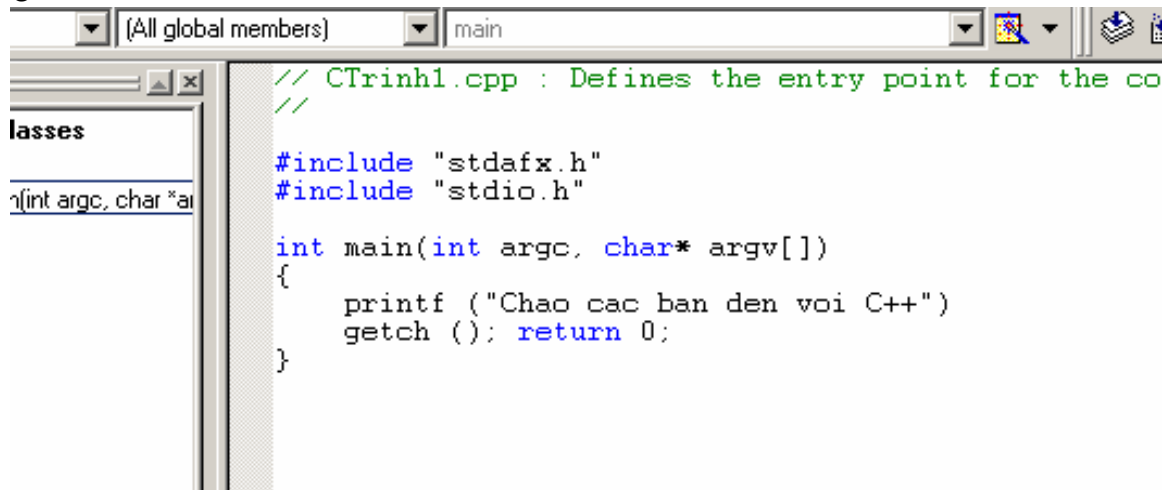




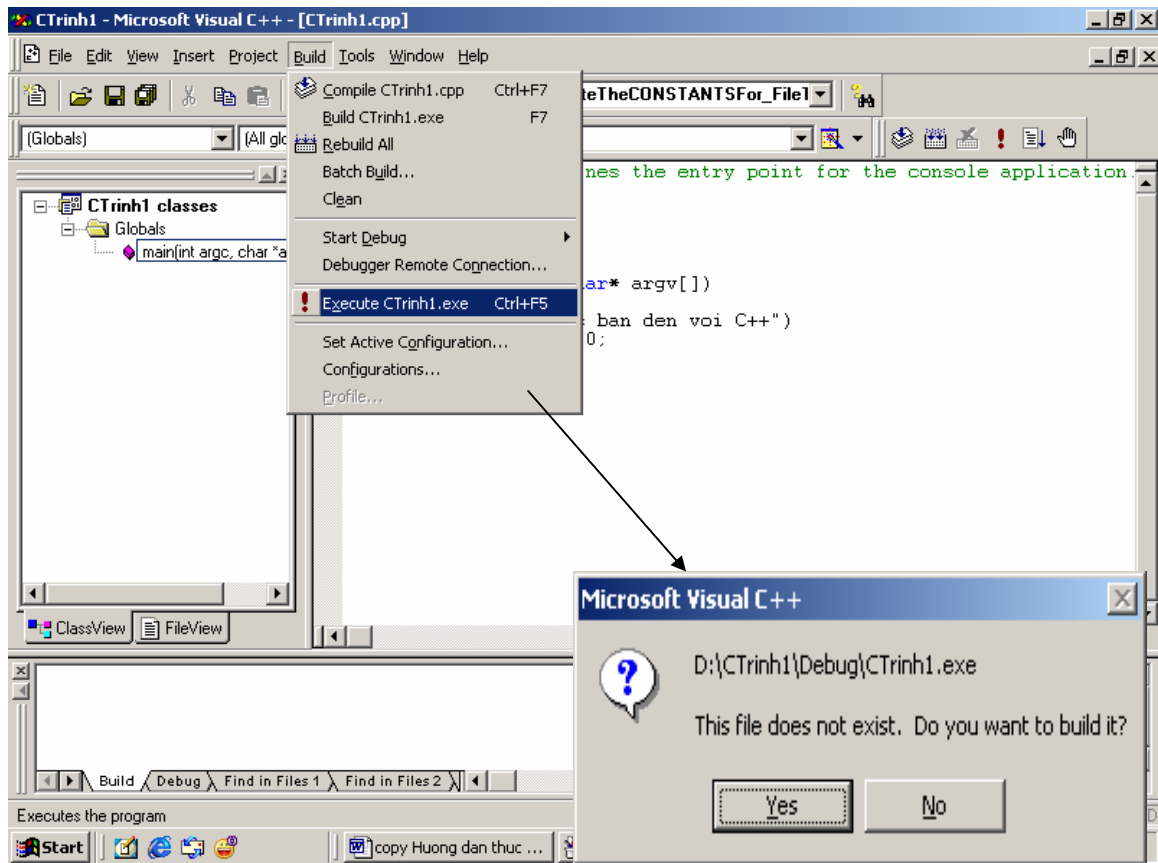


1.3: Viết CT:

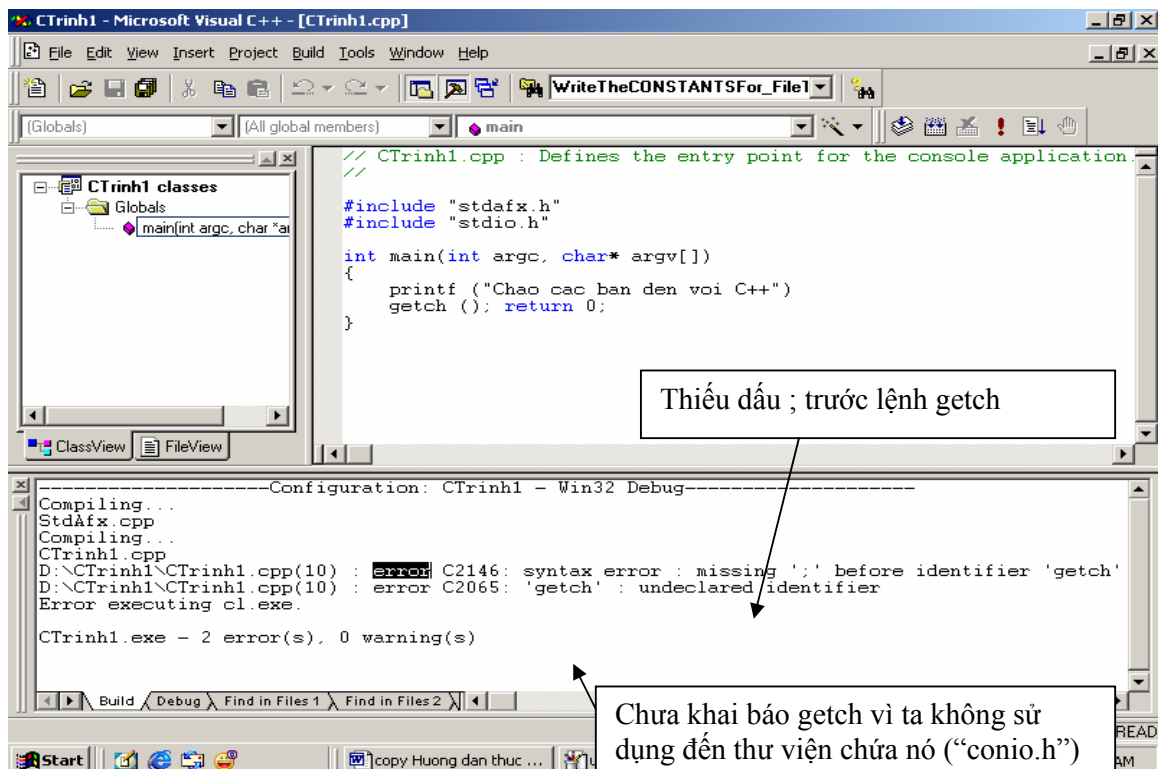
Xét *CT* in ra câu chào “Chào các bạn đến với C++.



1.4: Chạy và kiểm tra tính đúng của CT



CT này có lỗi cú pháp, do đó trong cửa sổ thông báo có xuất hiện lỗi:



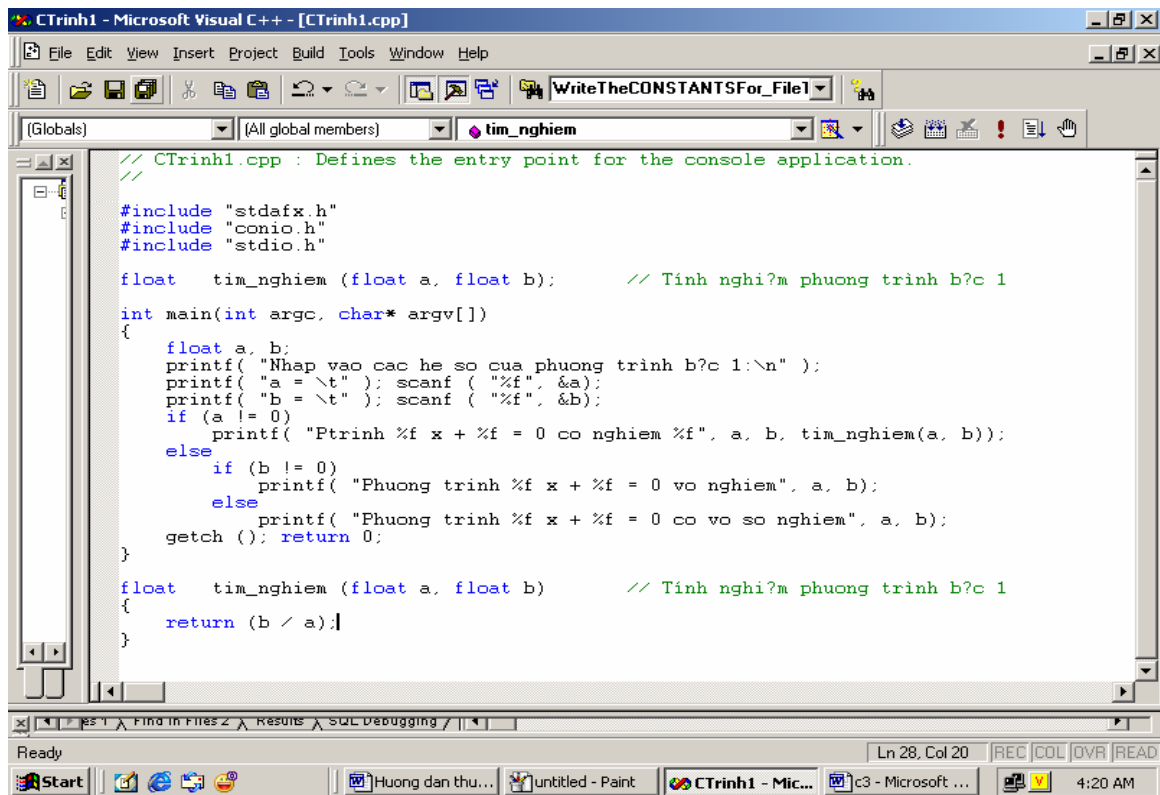
Quay về cửa sổ soạn thảo, ta sửa lại CT như sau:

```
#include "stdafx.h"
#include "stdio.h"
#include "conio.h"

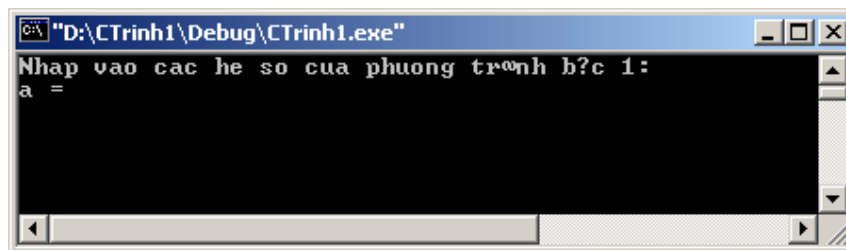
int main(int argc, char* argv[])
{
    printf ("Chao cac ban den voi C++")\n;
    getch (); return 0;
}
```

Thực hiện lại Bước 4, CT sẽ in ra kết quả “Chao cac ban den voi C++” (đúng với kết quả mong muốn). Ấn Enter để trở lại Visual C++.

1.4.1 XÉT CT GIẢI PHƯƠNG TRÌNH BẬC NHẤT:



Biên dịch và chạy: nhập giá trị cho a là 10, ấn Enter



nhập giá trị cho b là 0, ấn Enter.

kết quả CT hiện ra là đúng, nhưng ta không vội. Thử chạy lại CT với $a = 10$ và $b = 5$:

nghiệm của phương trình phải là -0.5 ($= -5/10$), nhưng kết quả lại là 0.5 . Quay lại CT, sửa lại hàm tìm nghiệm:

```
printf( "b = \t" ); scanf( "%f", &b);
if (a != 0)
    printf( "Ptrinh %f x + %f = 0 co nghiem %f", a, b, tim_nghiem(a, b));
else
    if (b != 0)
        printf( "Phuong tṛnh %f x + %f = 0 vo nghiem", a, b);
    else
        printf( "Phuong tṛnh %f x + %f = 0 co vo so nghiem", a, b);
getch(); return 0;

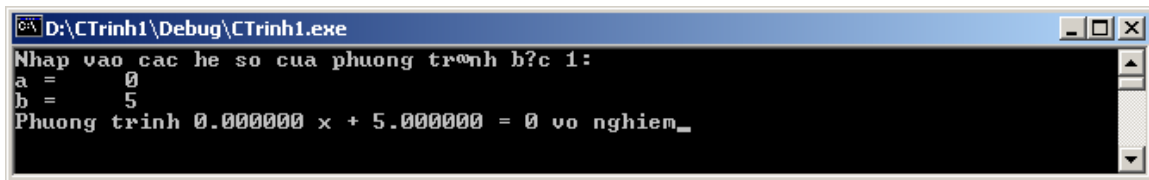
float tim_nghiem(float a, float b) // Tinh nghi?m phuong tṛnh b?c 1
{
    return (b / a);
}
```

Sai: phải là $-b/a$

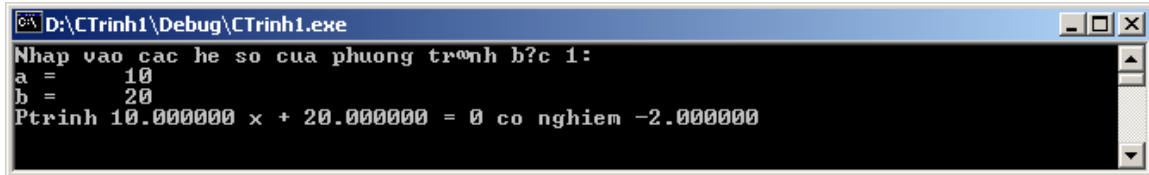
Sau khi sửa lại b/a thành $-b/a$, ta tiếp tục biên dịch và chạy thử CT. Nhập lại $a = 10$ và $b = 5$:

ta có nghiệm đúng là -0.5 . Nhưng rút kinh nghiệm lần trước, ta lại biên dịch và chạy CT. Lần này, ta thử với $a = 0$ và $b = 0$:

Ta nhận được kết luận “Phương trình có vô số nghiệm”, kết luận này đúng. Lại tiếp tục thử với bộ dữ liệu $a = 0$ và $b = 5$:



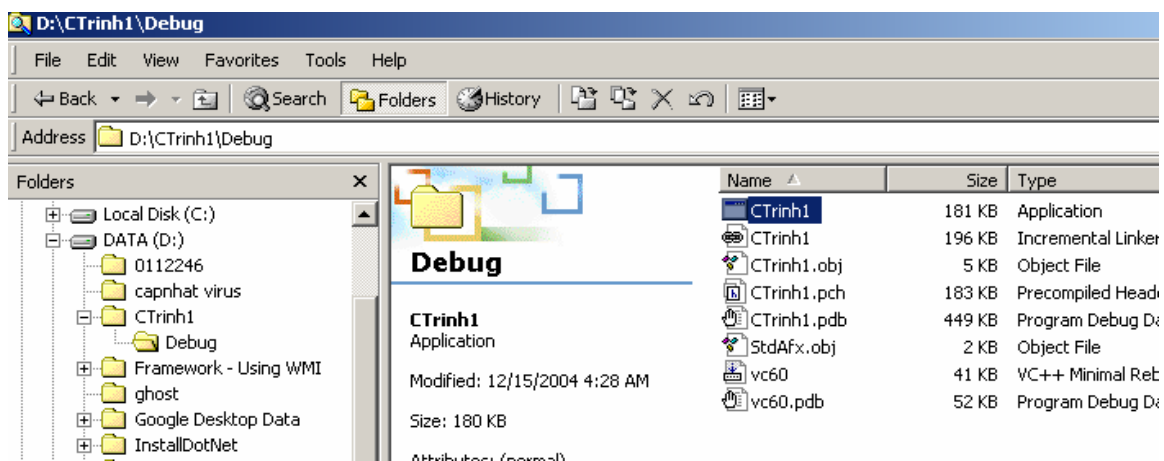
Ta nhận được kết luận “Phương trình vô nghiệm”, kết luận này đúng. Lại tiếp tục thử với bộ dữ liệu $a = 10$ và $b = 20$:



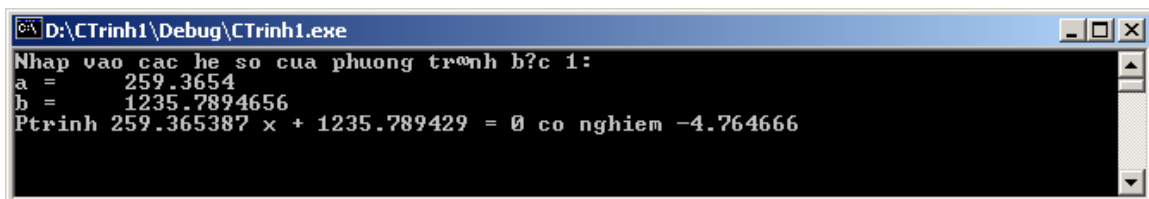
Ta nhận được nghiệm đúng -2 . Lúc này ta tạm yên tâm về tính đúng của CT.

1.4.2 TẬP TIN THI HÀNH CỦA CT:

Sau khi, có được CT đúng, ta vào thư mục Debug trong thư mục lưu dự án *D:\CTrinh1* để lấy tập tin thi hành có tên *CTrinh1*



để sử dụng khi muốn giải phương trình bậc 1. Chẳng hạn:



Mục 2: HƯỚNG DẪN SỬA MỘT SỐ LỖI / CẢNH BÁO THƯỜNG GẶP:

Nguyên tắc sửa lỗi (lỗi / cảnh báo – error / warning (có thể bỏ qua mà không cần sửa, tuy nhiên một số warning nếu không được sửa sẽ làm cho chương trình (CT) chạy không đúng)):

- Kích đôi chuột vào thông báo lỗi để nhảy đến vị trí có lỗi trong chương trình (CT).
- Đọc dòng chứa con trỏ hoặc dòng trên (dưới) để sửa lỗi.
- Nếu không tìm thấy lỗi thì phải dò lỗi từ đầu CT đến dòng chứa con trỏ (có thể là do lỗi ở phần trên của dòng chứa con trỏ chứ không phải ở dòng chứa con trỏ (hoặc dòng trên nó)).

Các lỗi ngữ nghĩa (CT vẫn thực thi nhưng kết quả sai) trình biên dịch C++ không phát hiện được.

1. **(expected:** thiếu '('
2. **) expected:** thiếu ')'
3. **, expected:** thiếu ','
4. **: expected after private:** thiếu ':' sau private
5. **: expected after protected:** thiếu ':' sau protected
6. **: expected after public:** thiếu ':' sau public
7. **< expected:** thiếu dấu <
8. **{ expected:** thiếu dấu {
9. **} expected:** thiếu dấu }
10. **Array bounds missing]:** thiếu ']' bao dãy
11. **Array must have at least one element:** dãy phải có ít nhất một phần tử
12. **Array size too large:** kích cỡ dãy quá lớn
13. **Body already defined for this function:** nội dung hàm này đã được viết.
14. **Call of nonfunction:** tên được gọi không được khai báo như một hàm, do khai báo hàm không chính xác hoặc viết sai tên hàm.
15. **Cannot cast from 'type1' to 'type2':** không thể ép từ kiểu 'type1' đến kiểu 'type2'
16. **Cannot convert 'type1' to 'type2':** không thể chuyển đổi 'type1' thành 'type2'
17. **Cannot modify a const object:** không thể thay đổi một đối tượng hằng (const)
18. **Cannot overload 'main':** không thể định nghĩa chồng hàm main
19. **Cannot use tiny or huge memory model with Windows:** không thể sử dụng mô hình bộ nhớ tiny hoặc huge với Windows
20. **Cannot open such file or directory "xxx":** không thể mở file hoặc thư mục xxx
21. **Cannot open "Debug\.." for writting:** không thể mở file Debug\.. để ghi (hãy đóng CT đã chạy trước đây để có thể chạy lại CT)
22. **Case outside of switch:** 'case' bên ngoài switch
23. **Case statement missing ':':** 'case' thiếu dấu ':'
24. **Character constant must be one or two characters long:** hằng ký tự chỉ có thể là một ký tự ('a') hoặc hai ký tự ('\n')
25. **Compound statement missing }:** thiếu dấu } cho khối lệnh (câu lệnh phức).
26. **Constant expression required:** dãy phải được khai báo với kích thước là hằng số (thường là do khai báo hằng (#define) không đúng).
27. **Constant variable 'variable' must be initialized:** biến có kiểu const phải được khởi tạo (vì ta không thể gán giá trị cho biến có kiểu const trong quá trình thi hành CT).

28. **Could not find a match for argument(s)**: các đối số không phù hợp (kiểm tra lại khai báo hàm và các đối số truyền vào)
29. **Could not find file 'filename'**: không thể tìm file '**filename**'
30. **Declaration does not specify a tag or an identifier**: khai báo (kiểu struct hoặc kiểu union) không chứa thành phần
31. **Declaration is not allowed here**: không cho phép khai báo ở đây
32. **Declaration missing ';'** : khai báo thiếu dấu ';'
33. **Declaration syntax error**: khai báo sai lỗi cú pháp
34. **Declaration terminated incorrectly**: kết thúc khai báo không chính xác
35. **Declaration was expected**: khai báo được mong muốn ở đây nhưng không tìm thấy
36. **Default argument value redeclared**: giá trị của tham số mặc định bên trong hàm bị thay đổi
37. **Default argument value redeclared for parameter 'parameter'**: giá trị của tham số (đối số) mặc định '**parameter**' bên trong hàm bị thay đổi
38. **Default expression may not use local variables**: một biểu thức tham số (đối số) mặc định bên trong hàm không được phép sử dụng tham số khác
39. **Default outside of switch**: default bên ngoài switch
40. **Default value missing**: tham số theo sau một tham số mặc định phải có giá trị mặc định
41. **Default value missing following parameter 'parameter'**: thiếu giá trị mặc định cho tham số '**parameter**' (vì nó theo sau một tham số mặc định nên phải có giá trị mặc định)
42. **Define directive needs an identifier**: khai báo define cần có một tên
43. **Delete array size missing]**: thiếu ']' khi hủy một dãy
44. **Division by zero**: chia cho 0
45. **do statement must have while**: do phải có while
46. **do-while statement missing (**: do-while thiếu dấu '('
47. **do-while statement missing)**: do-while thiếu dấu ')'
48. **do-while statement missing ;**: do-while thiếu dấu ';'
49. **Duplicate case**: mỗi case trong switch phải có giá trị đi kèm
50. **Enum syntax error**: khai báo kiểu enum sai cú pháp
51. **Expression expected**: một biểu thức được mong muốn ở đây nhưng ký hiệu hiện thời không thể bắt đầu cho một biểu thức
52. **Expression of scalar type expected**: mong muốn biểu thức có kiểu vô hướng. Các toán tử !, ++, và – yêu cầu một biểu thức có kiểu vô hướng (char, short, int, long, enum, float, double, long double, pointer)
53. **Expression syntax**: cú pháp biểu thức
54. **File name too long**: tên file quá dài
55. **For statement missing '('**: câu lệnh for thiếu '('
56. **For statement missing)**: câu lệnh for thiếu ')'
57. **For statement missing ;**: câu lệnh for thiếu ';'
58. **'function' cannot return a value**: hàm không thể trả về giá trị (nó là hàm void)
59. **'function' must be declared with no parameters**: hàm phải được khai báo với không tham số
60. **'function' must be declared with one parameter**: hàm phải được khai báo với một tham số

61. **'function' must be declared with two paraameters:** hàm phải được khai báo với một tham số
62. **'function1' cannot be distinguished from 'function2':** không thể phân biệt 'function1' với 'function2'
63. **Function 'function' should have a prototype:** hàm 'function' nên có tiêu đề.
64. **Function call missing):** thiếu dấu ')' khi gọi hàm.
65. **Function should return a value:** chưa trả về giá trị cho hàm
66. **'identifier' cannot start a parameter declaration:** 'identifier' không thể bắt đầu cho khai báo một tham số
67. **'identifier' is not a member of struct:** 'identifier' không phải là thành phần của struct
68. **'identifier' is not a non-static member and can't be initialized here:** 'identifier' không phải là một biến tĩnh và không thể được khởi tạo ở đây
69. **'identifier' is not a parameter:** 'identifier' không phải là một tham số
70. **Identifier expected:** mong muốn một định danh
71. **If statement missing (:** câu lệnh if thiếu '('
72. **If statement missing):** câu lệnh if thiếu ')'
73. **Illegal character 'character' (0x'value'):** hằng ký tự sai
74. **Illegal structure operation:** toán tử trên struct không đúng (chỉ có thể là: '.', '&, '=)
75. **Illegal use of floating point:** toán tử trên số thực chấm động không đúng (chỉ có thể là: SHL, SHR, AND, OR, XOR, NOT, '? :', '*', ...)
76. **Improper use of typedef 'identifier':** kiểm tra khai báo 'identifier' ở dòng typedef
77. **Incorrect number format:** định dạng số không đúng
78. **Incorrect use of default:** sau default không có dấu ':'
79. **Invalid use of dot: sử dụng dấu '.' không đúng, ví dụ:**

```
struct foo {
    int x;
    int y;
    }p = {0,0};

int main (...)
{
    p.x++;    /* Đúng */
    p. y++;   /* Sai: Invalid use of dot */
    return 0;
}
```
80. **Lvalue required:** thành phần bên trái của lệnh gán phải là biến
81. **main must have a return type of int:** hàm main phải return về kiểu int
82. **Misplaced break:** break không nằm trong switch hoặc một vòng lặp
83. **Misplaced continue:** continue không nằm trong một vòng lặp
84. **Misplaced else:** else không có if
85. **Missing xxx before yyy:** thiếu xxx trước yyy
86. **Missing function header (old-style):** sai tiêu đề ở phần định nghĩa hàm (có thể thừa dấu ; sau tiêu đề)
87. **Multiple declaration for 'identifier':** trùng khai báo cho 'identifier'
88. **Need an identifer to declare:** cần một định danh cho khai báo
89. **No : following the ?:** không có : sau ? trong cấu trúc tam phân (... ? ... : ...)

90. **Not an allowed type**: không cho phép kiểu này (chẳng hạn, không thể trả về dữ liệu kiểu mảng tĩnh cho hàm)
91. **Numeric constant too large**: hằng số quá lớn
92. **new line in constant**: thiếu dấu ”
93. **operator [] missing**: toán tử [] thiếu]
94. **sizeof may not be applied to a function**: toán tử sizeof không thể áp dụng cho hàm
95. **Size of 'identifier' is unknown or zero**: kích thước của 'identifier' không biết hoặc là 0
96. **Size of the type is unknown or zero**: kích thước của kiểu không biết hoặc là 0
97. **Statement missing ';'**: thiếu dấu ‘;’
98. **Structure size too large**: kích thước của struct quá lớn
99. **Switch selection expression must be of integral type**: biểu thức chọn của switch phải là kiểu nguyên
100. **unexpected end of file**: thiếu }
101. **xxx undeclared identifier**: thiếu khai báo xxx

Mục 3: KỸ THUẬT CHẠY DEBUG ĐỂ GỠ RỐI CHƯƠNG TRÌNH

3.1 Xét CT xuất ra bảng mã ASCII của 256 ký tự:

```
// Các thu vi?n
#include "stdafx.h"
#include "conio.h"          // Ch?a getch() d? d?ng CT xem k?t
#include "stdio.h"          // Ch?a getch() d? d?ng CT xem k?t

// Hàm chính
int main(int argc, char* argv[])
{
    // Khai báo các bi?n
    int k;
    printf ("-----[ MaASCII, KyTu]-\n");
    for (k = 0; k < 256; k++);
        printf (" %c %d", k, k);

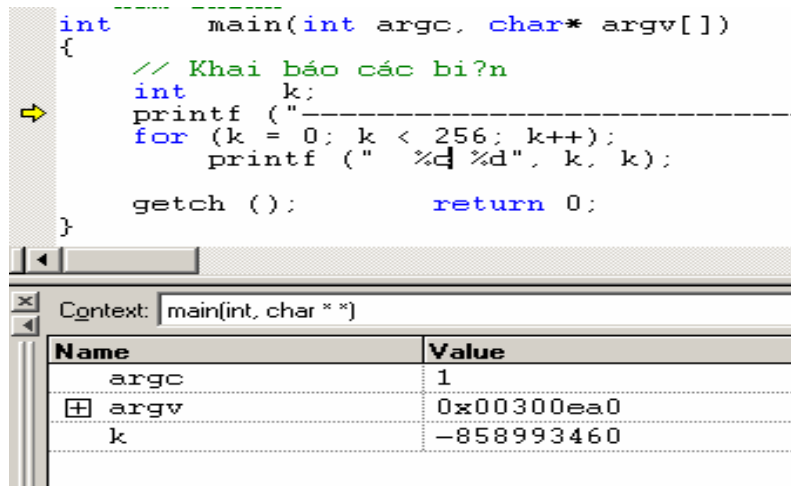
    getch ();              return 0;
}
```

Khi chạy CT, ta có kết quả sai

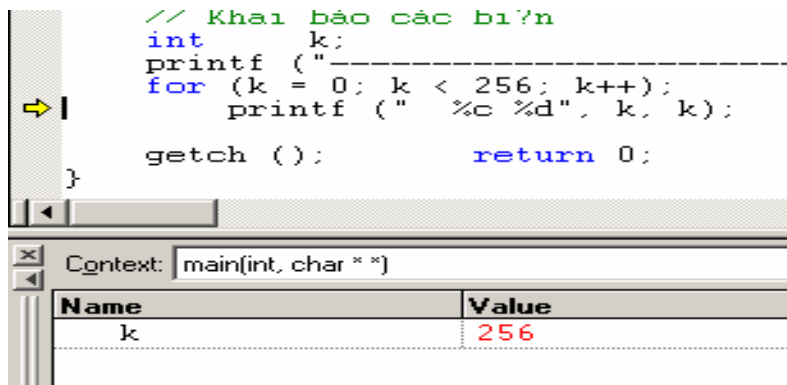


Để gỡ rối CT, ta ấn F10 để chạy CT từng bước và quan sát giá trị của các biến ở cửa sổ bên dưới:

B1) Ấn F10



B2) Ấn F10

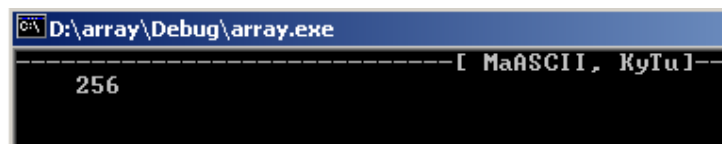


Ta thấy, giá trị bây giờ của k là 256 mặc dù ta muốn ứng với mỗi k từ 0 đến 255 thực hiện lệnh

`printf (" %c %d", k, k);`

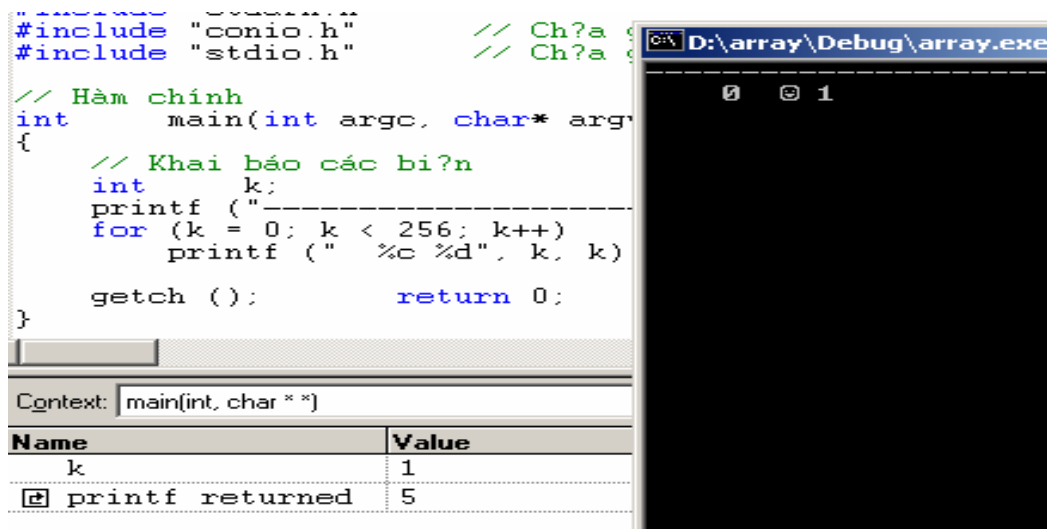
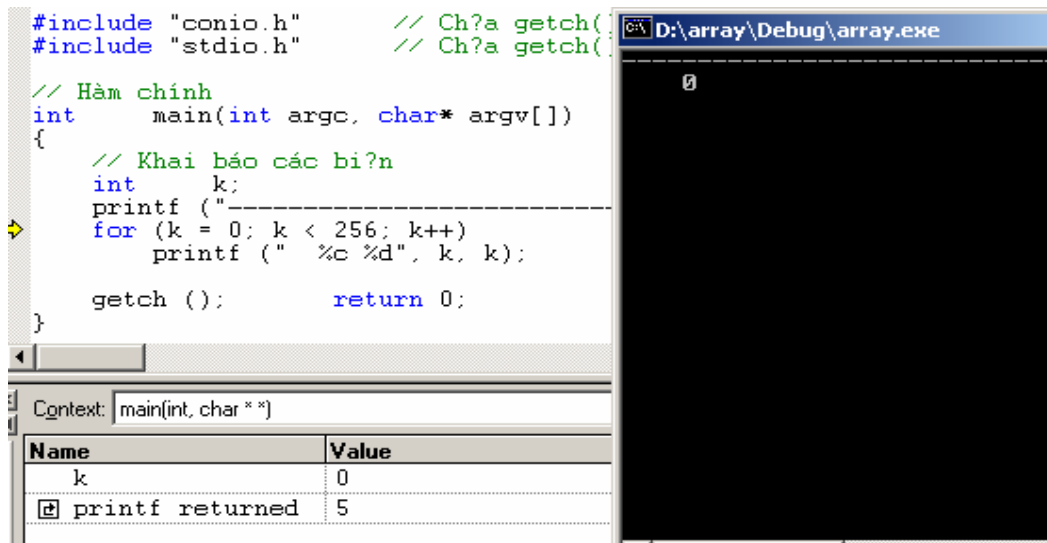
Điều đó, chứng tỏ k lặp từ 0 đến 255 mà không thực hiện gì.

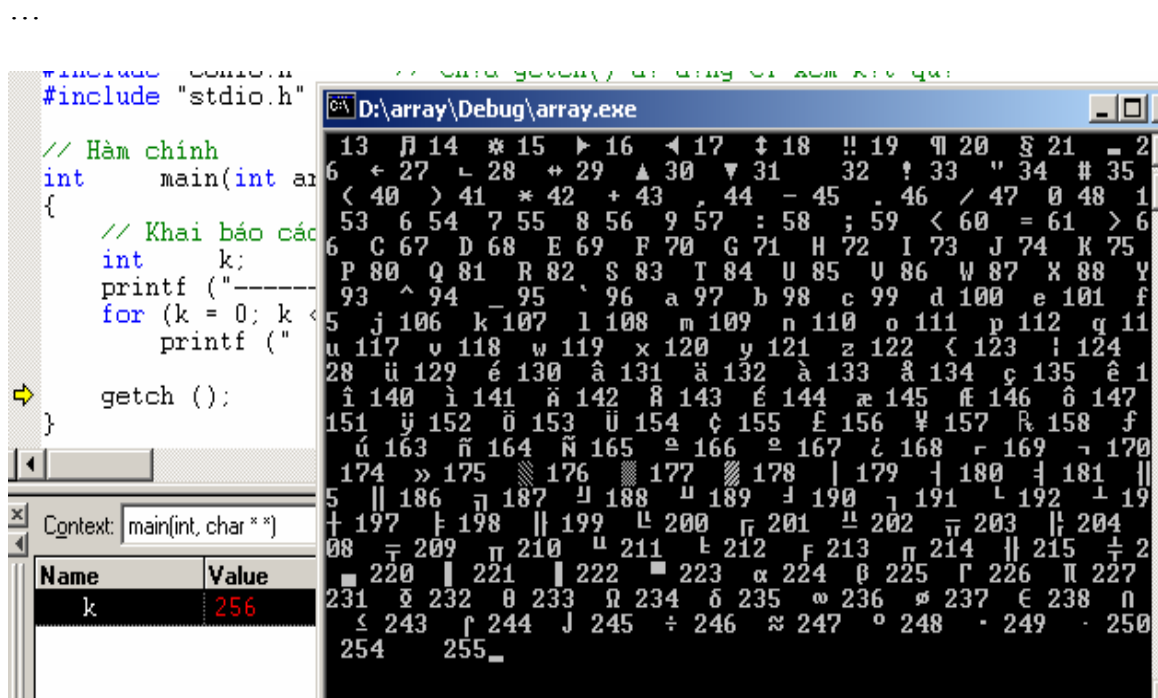
Thật vậy, ấn F10, ta có kết quả của CT chỉ là con số 256 mà không có ký tự nào được in ra



Lỗi sai của CT trên là do dấu ';' ở cuối vòng lặp for: CT thực hiện 256 lần lệnh ';' mà không làm gì cả. Kết thúc vòng lặp giá trị của k là 256, CT xuất ra số 256 (dĩ nhiên không có ký tự nào có mã ASCII là 256).

Sửa: Bỏ dấu ';' ở cuối for ta có kết quả chạy từng bước như sau



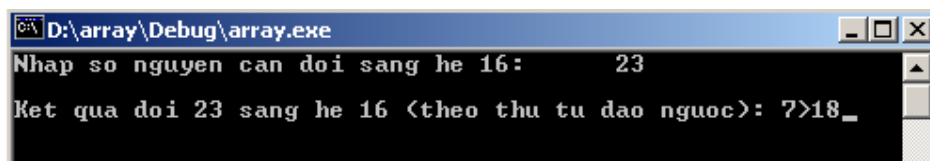


Ấn một phím để kết thúc CT.

3.2. Xét CT đổi số sang hệ 16:

```
// Hàm chính
int main(int argc, char* argv[])
{
    int    so, du, b;
    printf( "Nhap so nguyen can doi sang he 16:\t"); scanf ("%d", &so);
    b = so;
    printf( "\nKet qua doi %d sang he %d (theo thu tu dao nguoc): ", so, 16);
    do {
        du = b % 16;
        b = b / 16;
        switch (du) {
            case 0: case 1: case 2: case 3: case 4: case 5:
            case 6: case 7: case 8: case 9:
                printf( "%c", du + '0');
            default:
                printf( "%c", 'A' + du - 10);
        }
    } while (b != 0);
    getch (); return 0;
}
```

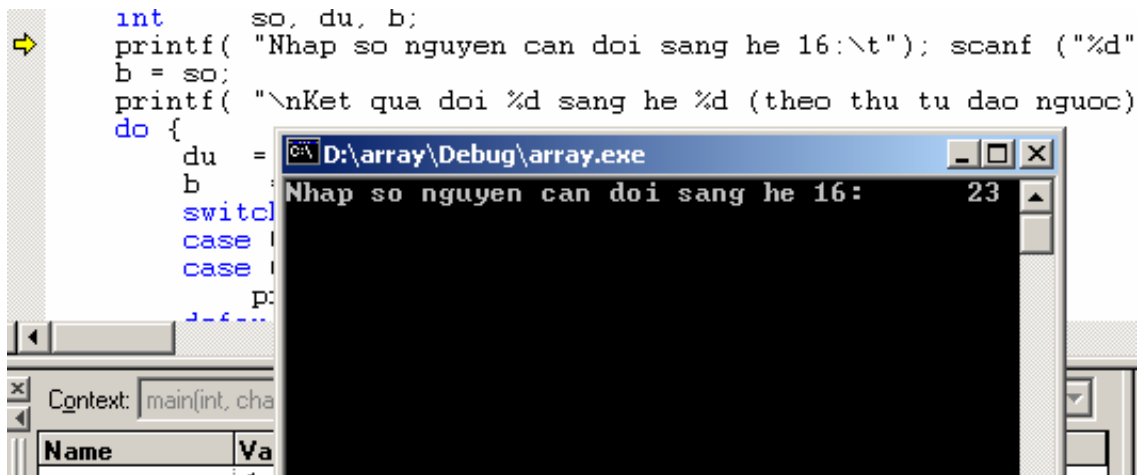
Kết quả chạy CT đổi số 23 sang hệ 16 là “7>18”:



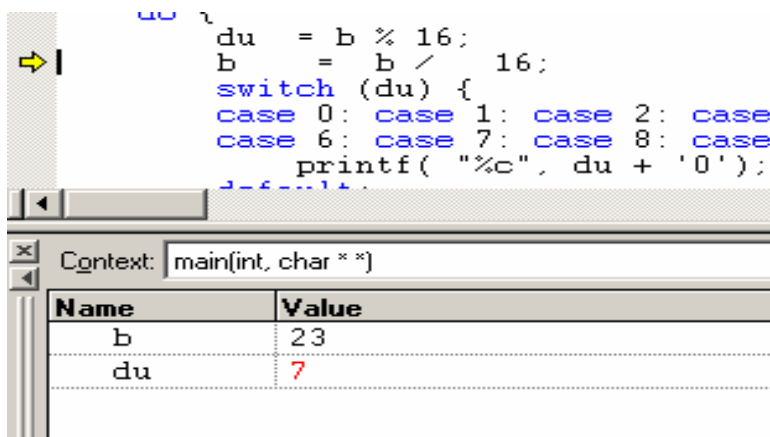
Kết quả sử dụng Calculator (một ứng dụng tính toán của Windows) là 17. Do đó kết

quả của CT phải là 71 (thuật toán xuất theo thứ tự đảo ngược). Nhưng ở đây lại là “7>18”. Ta thử tìm lý do.

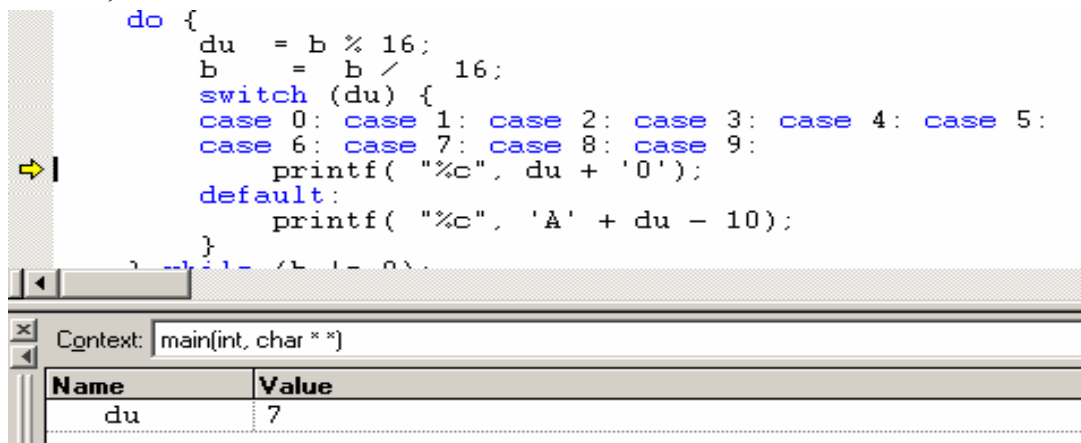
Ấn F10, để chạy từng bước CT.



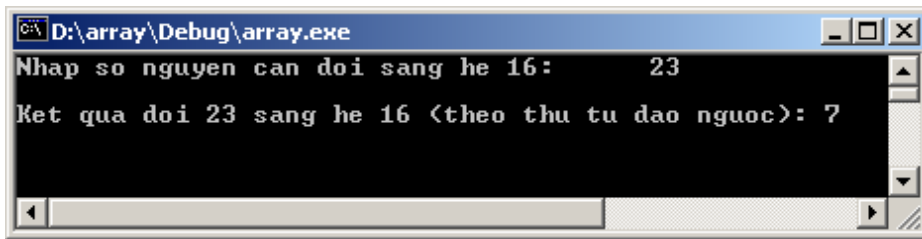
Ấn F10,



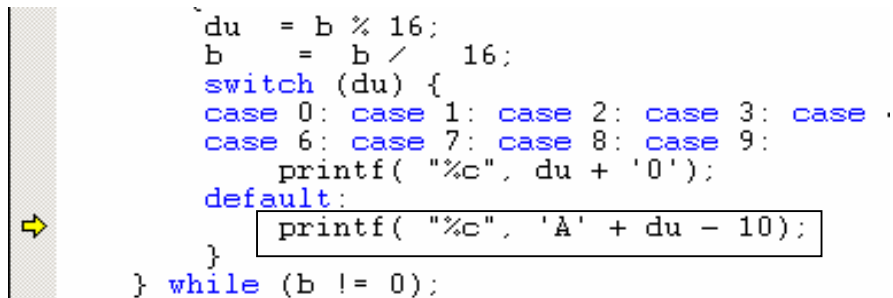
Ấn F10,



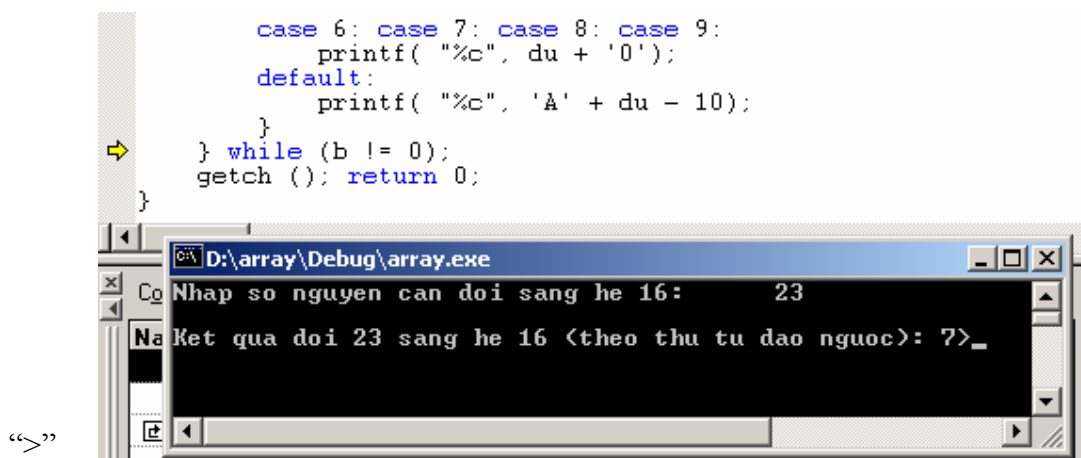
Kết quả hiện tại trên màn hình:



Ấn F10,



CT chạy đến câu lệnh `printf("%c", 'A' + du - 10);` tương ứng với default. Kết quả hiện ra màn hình là



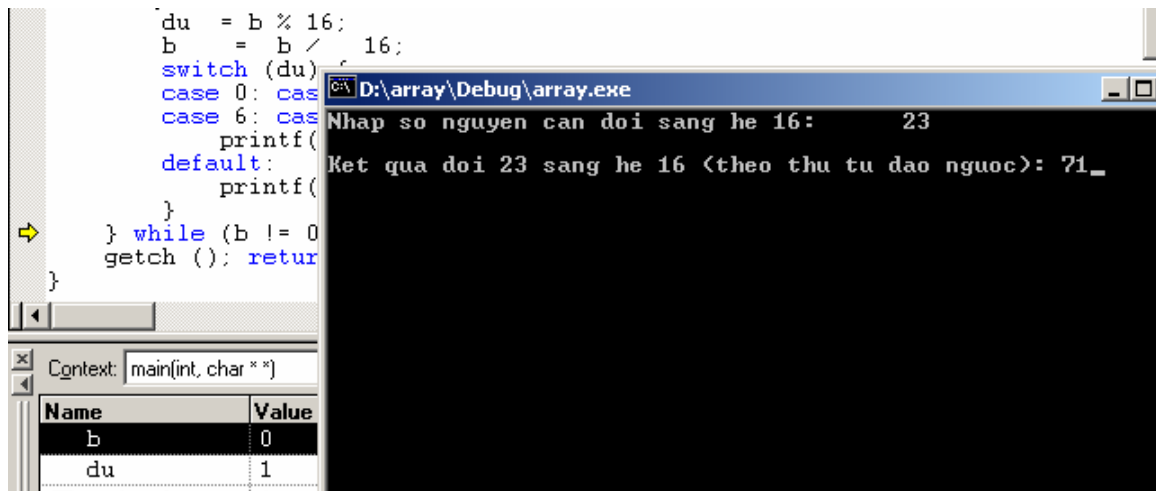
chính là kết quả quy đổi số dư 7 sang ký tự tương ứng ở hệ 16.

Điểm sai của CT là thiếu break để ngăn CT tiến xuống default sau khi đã rơi vào

`case 0: case 1: case 2: case 3: case 4: case 5:`

`case 6: case 7: case 8: case 9:`

Sau khi **dừng việc chạy từng bước** CT bằng **Debug \ Stop Debugging**, ta thêm vào break bị thiếu trước default, kết quả đổi số 23 là 71



3.3 Xét CT đổi số từ hệ 10 sang hệ b dùng hàm:

```
int main(int argc, char* argv[])
{
    int so, b;

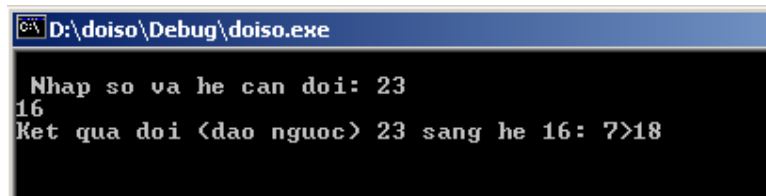
    printf ("\n Nhap so va he can doi: ");
    scanf ("%d%d", &so, &b);
    doi_so (so, b);

    getch (); return 0;
}

void    xuat_ky_so (int du)
{
    switch (du) {
        case 0: case 1: case 2: case 3: case 4: case 5:
        case 6: case 7: case 8: case 9:
            printf( "%c", du + '0');]
        default:
            printf( "%c", 'A' + du - 10); break;
    }
}

void    doi_so (int so, int he)
{
    printf( "Ket qua doi (dao nguoc) %d sang he %d: ", so, he);
    do {
        int du = so % he;
        so = so / he;
        xuat_ky_so(du);
    } while (so != 0);
}
```

Kết quả chạy CT đổi số 23 sang hệ 16 là

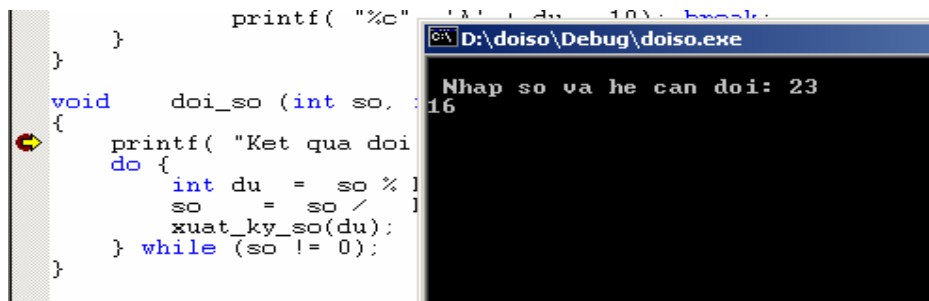


Kết quả của CT phải là 71 (thuật toán xuất theo thứ tự đảo ngược). Ta thử tìm lý do.

. Đặt điểm debug vào hàm doi_so bằng cách ấn F9 ở bên trong hàm này.

```
void doi_so (int so, int he)
{
    printf( "Ket qua doi (dao nguoc) %d sang he %d: ", so, he);
    do {
        int du = so % he;
        so = so / he;
        xuat_ky_so(du);
    } while (so != 0);
}
```

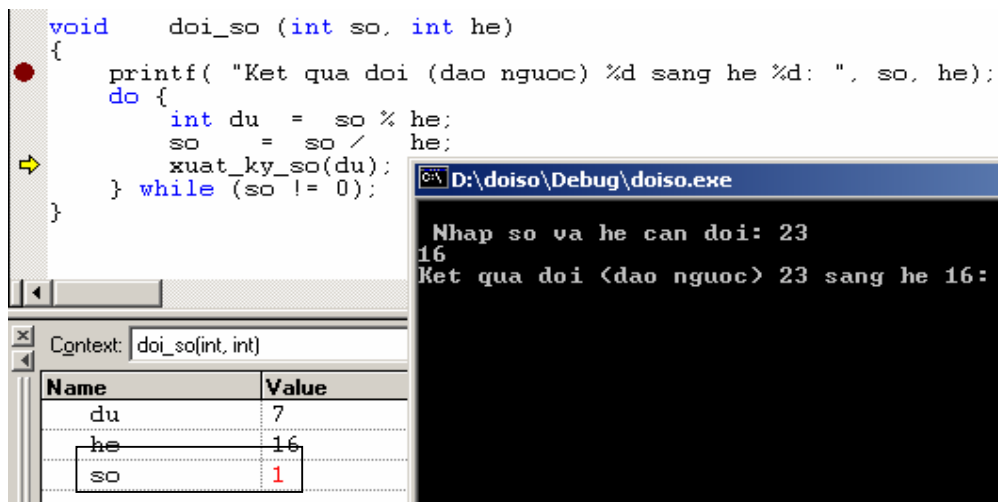
Ấn F5 để chạy CT. CT sẽ dừng ở bên trong hàm doi_so.



```
void doi_so (int so, int he)
{
    printf( "Ket qua doi (dao nguoc) %d sang he %d: ", so, he);
    do {
        int du = so % he;
        so = so / he;
        xuat_ky_so(du);
    } while (so != 0);
}
```

Console output: Nhap so va he can doi: 23, 16

Dùng F10, ta điều khiển CT đến:

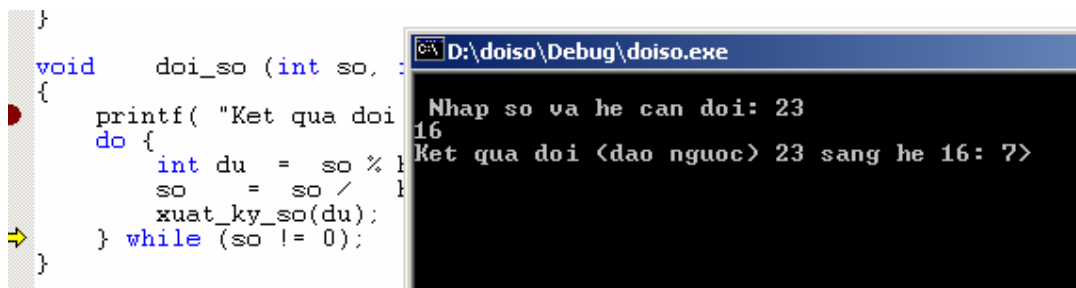


```
void doi_so (int so, int he)
{
    printf( "Ket qua doi (dao nguoc) %d sang he %d: ", so, he);
    do {
        int du = so % he;
        so = so / he;
        xuat_ky_so(du);
    } while (so != 0);
}
```

Console output: Nhap so va he can doi: 23, 16, Ket qua doi (dao nguoc) 23 sang he 16:

Name	Value
du	7
he	16
so	1

Số dư lúc này là 7. Ấn F10, ta có kết quả:



```
void doi_so (int so, int he)
{
    printf( "Ket qua doi (dao nguoc) %d sang he %d: ", so, he);
    do {
        int du = so % he;
        so = so / he;
        xuat_ky_so(du);
    } while (so != 0);
}
```

Console output: Nhap so va he can doi: 23, 16, Ket qua doi (dao nguoc) 23 sang he 16: 7>

Tương tự 3.2, ta biết rằng hàm `xuat_ky_so` không đúng. Stop debug (Debug/ Stop debugging).

. Đặt điểm debug vào hàm `doi_so` bằng cách ấn F9 ở bên trong hàm `xuat_ky_so`.

```
void    xuất_ky_so (int du)
{
    switch (du) {
        case 0: case 1: case 2: case 3: case 4: case 5:
        case 6: case 7: case 8: case 9:
            printf( "%c", du + '0');
        default:
            printf( "%c", 'A' + du - 10); break;
    }
}

void    doi_so (int so, int he)
{
    printf( "Ket qua doi (dao nguoc) %d sang he %d: ", so, he);
    do {
        int du = so % he;
        so = so / he;
        xuất_ky_so(du);
    } while (so != 0);
}
```

Ấn F5, và ấn F10 cho đến khi CT chạy vào trong điểm debug ở hàm `xuat_ky_so`

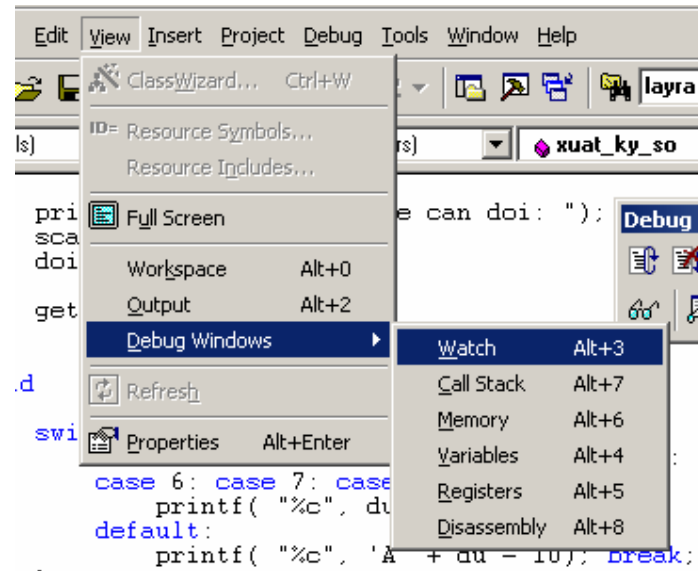
```
void    xuất_ky_so (int du)
{
    switch (du) {
        case 0: case 1: case 2: case 3: case 4: case 5:
        case 6: case 7: case 8: case 9:
            printf( "%c", du + '0');
        default:
            printf( "%c", 'A' + du - 10); break;
    }
}

void    doi_so (int so, int he)
{
    printf( "Ket qua doi (dao nguoc) %d sang he %d: ", so, he);
    do {
        int du = so % he;
        so = so / he;
        xuất_ky_so(du);
    } while (so != 0);
}
```

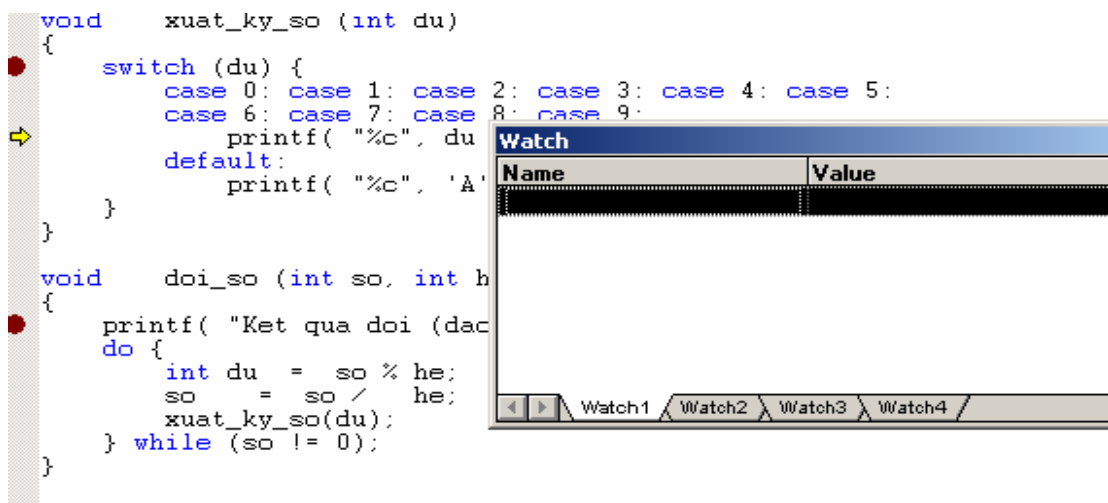
Ấn F10 cho đến khi

```
void    xuất_ky_so (int du)
{
    switch (du) {
        case 0: case 1: case 2: case 3: case 4: case 5:
        case 6: case 7: case 8: case 9:
            printf( "%c", du + '0');
        default:
            printf( "%c", 'A' + du - 10); break;
    }
}
```

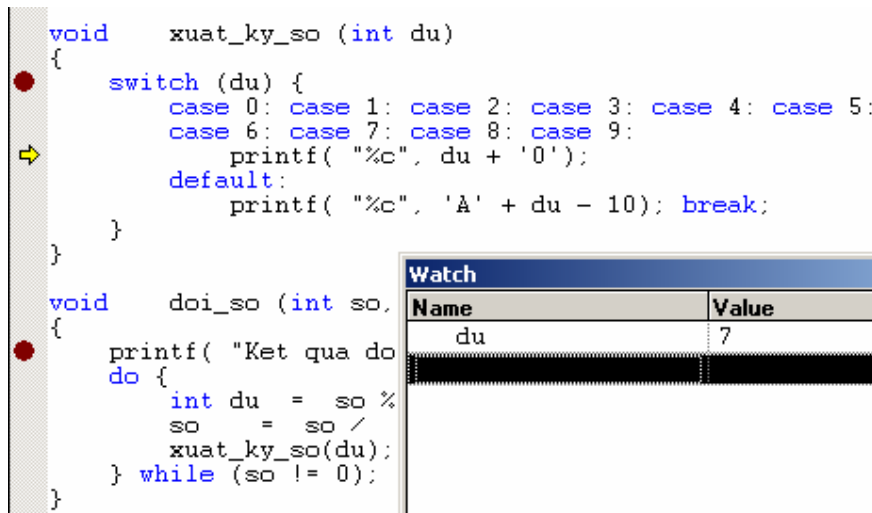
Xem giá trị số dư bằng cách



ta có,

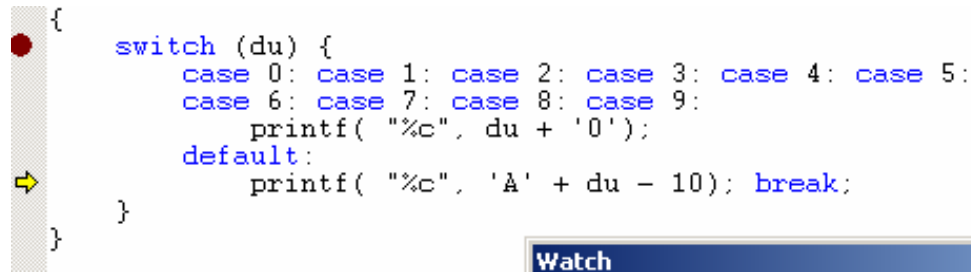


Kích đôi chuột vào ô Name và đặt vào “du”:



Ta có giá trị của du là 7, CT thực hiện câu lệnh “**printf("%c", du + '0');**” là hợp lý. Do đó kết quả xuất ra màn hình là 7.

Nhưng thử ấn F10 một lần nữa, ta thấy



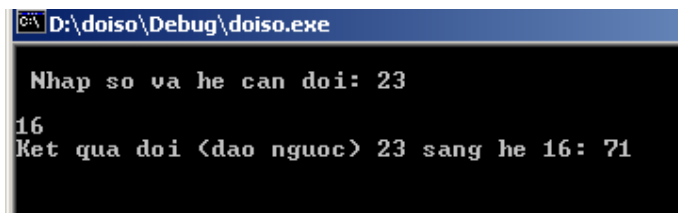
vì hàm xuất_ky_so thiếu **break;** sau “**printf("%c", du + '0');**” (tương tự 2.2).

Stop debugging, ta thêm “**break;**” vào sau “**printf("%c", du + '0');**”. Bỏ các điểm debug trong CT, bằng cách ấn F9 vào các điểm debug.

```
void    xuất_ky_so (int du)
{
    switch (du) {
        case 0: case 1: case 2: case 3: case 4: case 5:
        case 6: case 7: case 8: case 9:
            printf( "%c", du + '0');
            break;
        default:
            printf( "%c", 'A' + du - 10); break;
    }
}

void    doi_so (int so, int he)
{
    printf( "Ket qua doi (dao nguoc) %d sang he %d: ", so, he);
    do {
        int du = so % he;
        so = so / he;
        xuất_ky_so(du);
    } while (so != 0);
}
```

Chạy lại CT, ta có (kết quả đúng)



Mục 4: Một kỹ thuật kiểm chứng tự động CT trên các bộ dữ liệu được sinh ngẫu nhiên

Một vấn đề quan trọng trong lập trình là kiểm tra tính đúng đắn của một giải thuật hay một CT. Về mặt lý thuyết, ta có thể chứng minh được tính đúng của một CT hay một

giải thuật, tuy nhiên không phải lúc nào cũng dễ dàng, mà phải thực hiện các biến đổi toán học phức tạp. Về mặt thực hành, ta cần phải thử nghiệm CT trên một số lượng lớn các bộ dữ liệu thử. Các bộ dữ liệu thử cần phải được sinh tự động một cách ngẫu nhiên, vì nếu ta nhập từ bàn phím thì mất quá nhiều thời gian.

Trong mục này, ta sẽ kiểm chứng thuật toán sắp xếp tăng dần một dãy các số nguyên trên N (đủ lớn) bộ dữ liệu được sinh ngẫu nhiên. Hoạt động của CT như sau:

0. lần_thử = 0
1. Tăng lần_thử lên 1
2. Sinh ngẫu nhiên một mảng các số nguyên A.
3. Sắp xếp tăng dần A.
4. Kiểm tra A có được sắp tăng dần không ?
5. Nếu A không được sắp tăng dần thì thông báo “thuật toán sắp xếp tăng dần dãy không đúng với A”;
ngược lại, nếu lần_thử = N thì dừng CT (dùng kỹ thuật debug nêu trong mục 2 để dò lỗi và sửa lại thuật toán sắp xếp); ngược lại quay lại bước 1.

CT như sau:

```
#include ...
#include "stdlib.h" // Chứa các hàm: srand, rand
#include "time.h" // Lấy thời gian hệ thống
#define MAX_SO_PHAN_TU 20
#define MAX_PHAN_TU 99

// Định nghĩa kiểu mảng và khai báo các tiêu đề hàm

int main(int argc, char* argv[])
{
    int N, hat_giong, lan_thu = 0;
    mang A; int nA;

    printf("Xác định một số nguyên không âm làm hạt giống (0: hạt giống được tính
        theo thời gian của hệ thống) và số lần thử nghiệm: ");
    scanf("%d%d", &hat_giong, &N);
    if (hat_giong > 0) srand(hat_giong);
    else srand( time(0) );

    for(int k=1; k<=N; k++) {
        sinh_mang_ngau_nhien (A, nA) ;
        sap_xep_tang (A, nA) ;
        if (!tang_dan (A, nA))
        {
            printf("\n Thuật toán sắp xếp tăng không đúng khi áp dụng vào dãy");
            xuat_mang (A, nA);
            break;
        }
        else printf(" %d ", k);
    }
}
```

```

    getch(); return 0;
}
void sap_xep_tang (mang A, int nA)
{
    // ???
}
bool tang_dan (mang A, int nA)
{
    for (int i=1; i<=nA-1; i++)
        if (A[i] > A[i+1])
            return false;
    return true;
}
void sinh_mang_ngau_nhien (mang A, int &nA)
{
    nA = rand () % MAX_SO_PHAN_TU + 1;
    for (int i=1; i<=nA; i++)
        A[i] = rand () % MAX_PHAN_TU + 1;
}
void xuat_mang (mang A, int nA)
{
    // ???
}

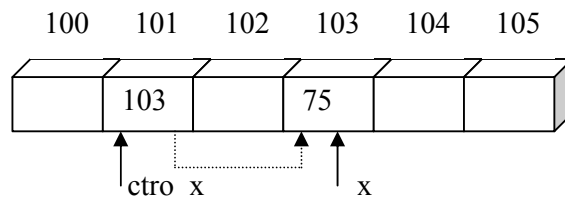
```

Mục 5: Con trỏ

1. Định nghĩa, khai báo, khởi tạo và sử dụng con trỏ

Con trỏ là một trong những công cụ mạnh của C/C⁺⁺, cho phép LTV xử lý bộ nhớ máy tính một cách trực tiếp. Con trỏ là một biến kiểu nguyên lưu giữ địa chỉ của (trỏ đến) một biến khác.

Ví dụ: ta có con trỏ `ctro_x` trỏ đến biến `x`



Cú pháp khai báo biến con trỏ:

*kiểu *tênBiếnConTrỏ;*

trong đó, kiểu có thể là số (nguyên, thực), ký tự, struct, mảng, ...

Sau khi khai báo, ta cần gán địa chỉ của biến mà con trỏ trỏ đến cho biến con trỏ, nếu chưa biết chính xác địa chỉ này thì ta gán trị NULL.

Ví dụ:

```
char *ch1, *ch2;      // ch1, ch2 là hai biến con trỏ trỏ đến biến kiểu char.
float *gia_tri, x;    // gia_tri là con trỏ trỏ đến biến kiểu float.
```

Con trỏ chỉ có tác dụng khi nó đang lưu trữ địa chỉ của một biến trong CT. Để lấy địa chỉ của một biến ta dùng toán tử &, do đó, câu lệnh lấy địa chỉ của biến x cho vào biến con trỏ ctro_x có dạng:

```
ctro_x = &x;
```

Khi đặt dấu * trước biến con trỏ ta sẽ có được giá trị của biến mà con trỏ đang lưu giữ địa chỉ.

2. Cấp phát và thu hồi vùng nhớ bằng con trỏ

Một phần bộ nhớ RAM dành cho CT khi chạy được chia thành bốn vùng:

- . Stack: lưu trữ các biến cục bộ.
- . Heap: chứa các biến được cấp phát động bằng toán tử new.
- . Static: chứa các biến ngoài hoặc biến tĩnh.
- . Code: chứa mã CT và các hằng.

Toán tử new cấp phát vùng nhớ cho một biến động có dạng:

```
contro = new kiểuBiếnConTrỏTrỏĐến;
```

Khi thành công, toán tử new trả về địa chỉ (của ô nhớ đầu tiên) của vùng nhớ. Nếu thất bại, toán tử new sẽ trả về trị NULL.

Ví dụ:

```
#include ...
int main (...)
{
    int bien_cuc_bo = 5;
    int *p_bien_cuc_bo = &bien_cuc_bo;
    printf("\nTrị của biến cục bộ: %d", bien_cuc_bo);
    printf("\nTrị của biến cục bộ truy cập bằng con trỏ: %d", *p_bien_cuc_bo);

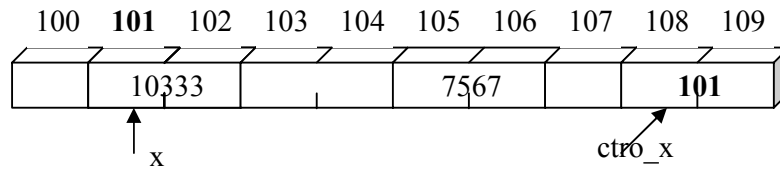
    int *pHeap = new int;
    if (pHeap == NULL)
        printf("\n Het bo nho Heap.");
    else
    {
        printf("\nĐịa chỉ của vùng nhớ vừa được cấp phát bằng new: %d", pHeap);
        delete pHeap;
    }
    getch(); return 0;
}
```

3. Toán tử tăng / giảm trên biến con trỏ

Khi tăng / giảm biến con trỏ n đơn vị tức là tăng / giảm giá trị của nó với một lượng bằng kích thước của kiểu của biến mà con trỏ trỏ đến nhân với n.

Ví dụ:

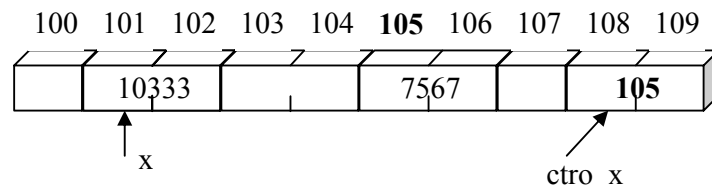
```
int x = 10333;
int *ctro_x = &x;
ctro_x = ctro_x + 2;
printf("\n%d", *ctro_x);    // giá trị là 10333
```



Sau khi thực hiện lệnh:

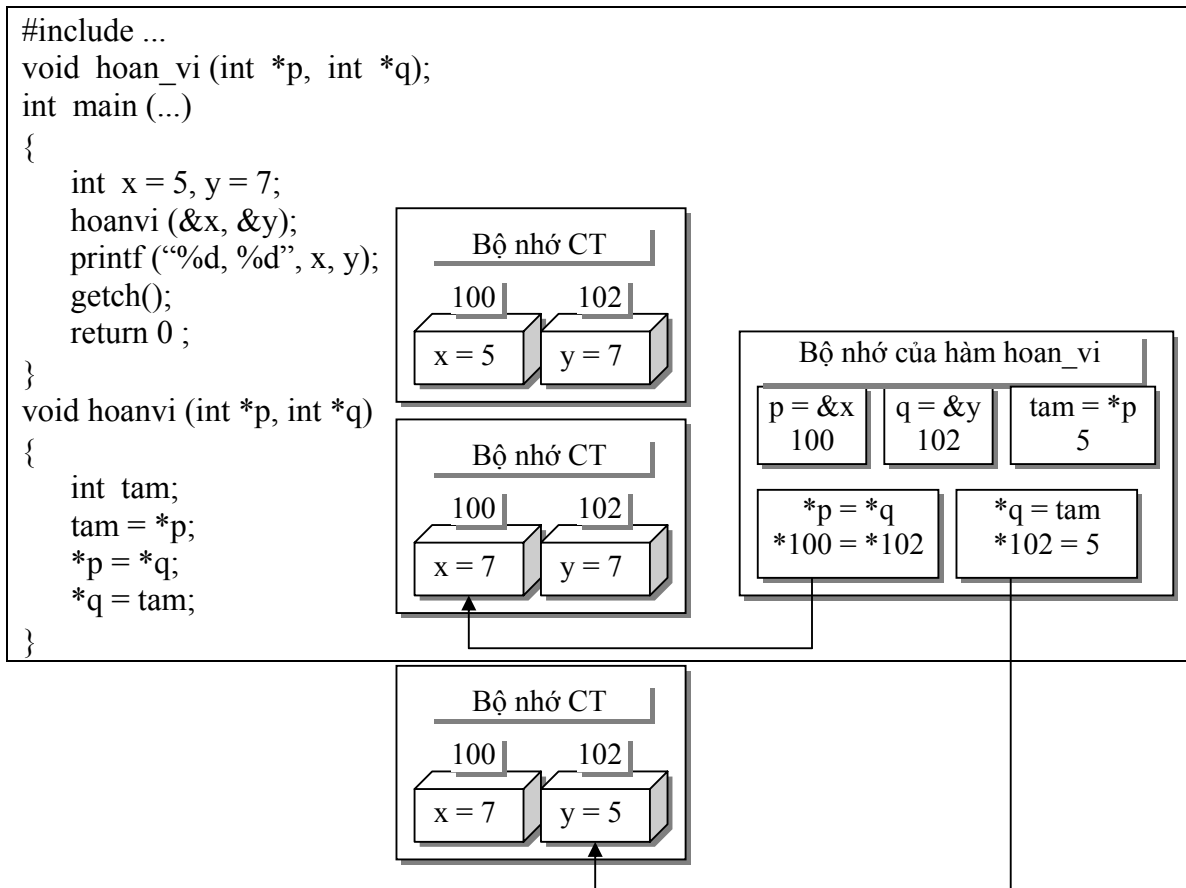
```
ctro_x = ctro_x + 2;
```

ta có:



Lệnh `printf("\n%d", *ctro_x);` sẽ in ra giá trị là 7567.

4. Một ứng dụng con trỏ để hoán vị giá trị hai biến



Mục 6: Tìm hiểu một số hàm xử lý chuỗi trong thư viện string.h

STT	TÊN HÀM	CHỨC NĂNG	VÍ DỤ
1	int strlen(char s[]);	Trả về độ dài của chuỗi s.	<pre>char *s = "International"; printf("Do dai %d",strlen(s));</pre> Kết quả: Do dai 3
2	strcpy(char dest[], char src[]);	Sao chép nội dung chuỗi src vào chuỗi dest.	<pre>char dest[10]; char *src = "abcdefghi"; strcpy(dest, src); printf("%s\n", dest);</pre> Kết quả: abcdefgh
3	strncpy(char dest[], char src[], int n);	Chép n ký tự từ chuỗi src sang chuỗi dest. Nếu chiều dài src < n thì hàm sẽ điền khoảng trống cho đủ n ký tự vào dest.	<pre>char dest[4]; char *src = "abcdefghi"; strncpy(dest, src, 3); printf("%s\n", dest);</pre> Kết quả: abc
4	strcat(char s1[], char s2[]);	Nối chuỗi s2 vào chuỗi s1.	<pre>char *s1 = "Khoa "; char *s2 = "CNTT"; strcat(s1, s2); printf("%s\n", s1);</pre> Kết quả: Khoa CNTT
5	strncat(char s1[], char s2[], int n)	Nối n ký tự đầu tiên của chuỗi s2 vào chuỗi s1.	<pre>char *s1 = "Khoa "; char *s2 = "CNTT"; strncat(s1, s2, 2); printf("%s\n", s1);</pre> Kết quả: Khoa CN
6	int strcmp(char s1[], char s2[])	So sánh 2 chuỗi s1 và s2 theo nguyên tắc thứ tự từ điển. Phân biệt chữ hoa và thường.	<pre>char *s1 = "abcd"; char *s2 = "abCD"; if(strcmp(s1, s2)==0) printf("Giống nhau"); else if(strcmp(s1, s2) < 0) printf("s1 < s2 "); else printf("s1 > s2 ");</pre> Kết quả: Khác nhau
	int strncmp(char s1[], char s2[], int n)	Tương tự như strcmp(), nhưng chỉ so sánh n ký tự đầu tiên của hai chuỗi.	<pre>char *s1 = "abcd"; char *s2 = "abef"; if(strncmp(s1, s2, 2)==0) printf("Giống nhau"); else printf("Khác nhau");</pre> Kết quả: Giống nhau

8	<code>int stricmp(char s1[], char s2[])</code>	Tương tự như <code>strcmp()</code> , nhưng không phân biệt hoa thường.	<code>char *s1 = "abcd";</code> <code>char *s2 = "abCD";</code> <code>if(stricmp(s1, s2)==0)</code> <code>printf("Giống nhau");</code> <code>else</code> <code>printf("Khác nhau");</code> Kết quả: <i>Giống nhau</i>
9	<code>int strnicmp(char s1[], char s2[], int n);</code>	Tương tự như <code>strcmp()</code> , nhưng chỉ so sánh n ký tự đầu tiên của hai chuỗi.	<code>char *s1 = "aBcd";</code> <code>char *s2 = "Abef";</code> <code>if(strnicmp(s1, s2, 2)==0)</code> <code>printf("Giống nhau");</code> <code>else</code> <code>printf("Khác nhau");</code> Kết quả: <i>Giống nhau</i>
10	<code>char *strchr(char s[], char c);</code>	Tìm lần xuất hiện đầu tiên của ký tự c trong chuỗi s. Trả về: • NULL: nếu không có. • Địa chỉ c: nếu tìm thấy.	<code>char s[15];</code> <code>char *ptr, c = 'm';</code> <code>strcpy(s, "Vi du tim ky tu");</code> <code>ptr = strchr(s, c);</code> <code>if (ptr)</code> <code>printf("%c tai %d", c, ptr-</code> <code>s);</code> <code>else</code> <code>printf("Khong tim thay");</code> Kết quả: <i>m tai 8</i>
11	<code>char *strstr(char s1[], char s2[]);</code>	Tìm sự xuất hiện đầu tiên của chuỗi s2 trong chuỗi s1. Trả về: • NULL: nếu không có. • Ngược lại: Địa chỉ bắt đầu chuỗi s2 trong s1.	<code>char *s1 = "International";</code> <code>char *s2 = "nation", *ptr;</code> <code>ptr = strstr(s1, s2);</code> <code>printf("Chuoi con: %s\n",</code> <code>ptr);</code> Kết quả: <i>Chuoi con: national</i>

BÀI TẬP

1. Tìm lỗi của đoạn CT sau:

```
{
    int *pInt;
    *pInt = 9;
    int so = 9, *pSo = &so;
```

2. Cho biết kết quả (lỗi, kết quả in ra màn hình, tác động, ...) của từng dòng lệnh trong các đoạn CT sau:

a) `int n = 1, *pi = &n;`

```
printf(“ %d, %d, %d”, n, *(&n), *pi);
```

- b)

```
int m, x = 1, *pi;
pi = &x; m = *pi;
printf(“ %d”, m);
int y1 = x + 1, y2 = *(&x) + 1, y3 = *pi + 1;
printf(“ %d, %d, %d”, y1, y2, y3);
```
- c)

```
int x = 5, *px = &x; px = 3;
```
- d)

```
int n = 22, *p = &n;
printf(“ %d ”, p);
```
- e)

```
int n = 22, *p = &n, *q = p;
delete q; printf(“ %d ”, *p);
```
- f)

```
int *p;
*p = 5; printf(“ %d ”, *p);
```
- g)

```
int n = 22; double x = 3.1416;
int *p = &n; double *q = &x;
printf(“ %d ”, q - p);
```
- h)

```
int *p = NULL;
*p = 5; printf(“ %d ”, *p);
```
- i)

```
int *p = new int;
*p = 5; printf(“ %d , %d”, *p, &(*p));
```
- j)

```
int *p = new int; *p = 72; printf(“ %d ”, *p);
int *q = new int; *q = 84; printf(“ %d ”, q);
q = p;
```
- k)

```
float x = 5, *px = new float;
px = &x;
```
- l)

```
float x = 5, *px = &x;
printf(“ %f, %f, %f, %f, %d”, x, *px, *(&(*px)), *(&(*(&(*px)))), &(*px));
```
- m)

```
char s[] = “ABCDEFGH”;
char *p = &s[3]; printf(“ %c”, *p);
++p; printf(“ %c”, *p);
p += 3; printf(“ %c”, *p);
p -= 6; printf(“ %c”, *p);
—p; printf(“ %c”, *p);
```

3. Cho định nghĩa kiểu

```
struct ngay {
    int    thu;
    chat   thang[10];
    int    nam;
};
```

Hãy cho biết kết quả (lỗi, kết quả in ra màn hình, tác động, ...) của từng dòng lệnh trong các đoạn CT:

- a)

```
ngay x = {2, “7”, 1969};
```



```

ngay *p = &x;
printf(" %d ", (*p).nam);
printf(" %d ", p → nam);
printf(" %d ", *p);
b) ngay x = {2, "7", 1969}, *p1, ds[4];
p1 = &ds[0];
for (int i=0; i<=4; i++)
    p1[i].nam = 1960 + i;
for (int i=0; i<=4; i++)
    printf(" %d ", p1[i]);
for (int i=0; i<=4; i++)
    *(p1+i).nam = 1950 + i;
for (int i=0; i<=4; i++)
    printf(" %d ", *(p1+i));
for (int i=0; i<=4; i++)
    (p1+i) → nam = 1940 + i;
for (int i=0; i<=4; i++)
    printf(" %d ", (p1+i) → nam);
p1 = &ds[2];
p1 += 2; cout << *p1;
--p1;    printf(" %d ", (*p1).nam);
++p1;    printf(" %d ", p1 → nam);

```

Viết các CT sau (và nếu được hãy áp dụng kỹ thuật kiểm chứng tự động)

4. Vẽ các tam giác sau:

AAAA	A	A	AAAA
A A	A A	AA	A A
AA	A A	A A	A A
A	AAAAAA	AAAA	AAAA

5. Một số hoàn thiện là một số có tổng các ước số của nó (không kể nó) bằng chính nó. Hãy liệt kê các số hoàn thiện nhỏ hơn 5000.

Ví dụ: số 6 là số hoàn thiện vì tổng các ước số là $1+2+3=6$.

6. In ra dãy số Fibonacci

```

f1 = f0 = 1 ;
fn = fn-1 + fn-2 ; (n>1)

```

7. Nhập vào giờ, phút, giây. Kiểm tra xem giờ, phút, giây đó có hợp lệ hay không, nếu hợp lệ cho biết giờ sau và trước đó 1 giây là bao nhiêu.

Ví dụ: Nhập 01:59:59

Giờ sau đó 1 giây 02:00:00

Giờ trước đó 1 giây 01:59:58

8. In ra bảng cửu chương từ 2 đến 9.

9. Nhập số nguyên dương n ($0 \leq n < 1000$) và in ra cách đọc của n.

Ví dụ: Nhập $n = 105$. In ra màn hình: *Mot tram le nam*.

10. Tính tiền thuê máy dịch vụ Internet và in ra màn hình kết quả với dữ liệu nhập vào là giờ bắt đầu thuê (GBD), giờ kết thúc thuê (GKT), số máy thuê (SoMay).
 - Điều kiện cho dữ liệu nhập: $6 \leq \text{GBD} < \text{GKT} \leq 21$. Giờ là số nguyên.
 - Đơn giá: 2500đ cho mỗi giờ máy trước 17:30 và 3000đ cho mỗi giờ máy sau 17:30.
11. Nhập số nguyên dương n gồm 5 chữ số, kiểm tra xem các chữ số n có phải là số đối xứng hay không.
Ví dụ: Đối xứng: 13531
Không đối xứng: 13921
12. Nhập số nguyên dương n gồm k chữ số ($0 < k \leq 5$):
 - a. đếm xem n có bao nhiêu chữ số chẵn và bao nhiêu chữ số lẻ.
 - b. đếm xem n có bao nhiêu chữ số là số nguyên tố.
 - c. tính tổng các ước số dương của n.
 - d. tìm ước số lẻ lớn nhất của n.
 - e. kiểm tra xem các chữ số của n có toàn lẻ hay toàn chẵn không.
 - f. sắp xếp các chữ số của n theo thứ tự tăng dần.
 - g. tìm vị trí xuất hiện của chữ số có giá trị x trong n.
 - h. kiểm tra xem các chữ số của n có được sắp thứ tự không.
 - i. tính giá trị trung bình các chữ số chẵn trong n.
13. Nhập 2 số a, b sao cho: số lớn nhất trong 2 số phải là một số dương và chia hết cho 7. Nếu nhập sai phải yêu cầu nhập lại cho đến khi đúng.
14. In ra màn hình tất cả các ngày (dưới dạng ngày/tháng) của năm hiện tại, in ra màn hình thời gian trong ngày (dưới dạng giờ:phút:giây).
15. Sắp xếp mảng:
 - a. theo thứ tự tăng dần của các phần tử là số nguyên tố.
 - b. các phần tử lẻ tăng dần.
 - c. các phần tử chẵn giảm dần.
 - d. các phần tử chẵn nằm bên trái theo thứ tự tăng dần còn các phần tử lẻ bên phải theo thứ tự giảm dần.
 - e. các phần tử âm giảm dần từ trái sang phải, phần tử dương tăng dần từ phải sang trái.
16. Xoá:
 - a. phần tử tại vị trí lẻ trong mảng.
 - b. phần tử có giá trị lớn nhất trong mảng.
 - c. xoá tất cả các phần tử có giá trị nhỏ hơn X.
 - d. xoá phần tử có giá trị gần X nhất.
17. Chèn phần tử có giá trị X vào:
 - a. vị trí đầu tiên của mảng.
 - b. phía sau phần tử có giá trị lớn nhất trong mảng.
 - c. trước phần tử có giá trị là số nguyên tố đầu tiên trong mảng.
 - d. phía sau tất cả các phần tử có giá trị chẵn trong mảng.
18. Tách 1 mảng các số nguyên thành 2 mảng a và b, sao cho mảng a chứa toàn số lẻ và mảng b chứa toàn số chẵn.
Ví dụ: Mảng ban đầu: 1 3 8 2 7 5 9 0 10
Mảng a: 1 3 7 5 9

Mảng b: 8 2 10

19. Cho 2 mảng số nguyên a và b kích thước lần lượt là n và m. Viết chương trình nối 2 mảng trên thành mảng c theo nguyên tắc chèn ở đầu mảng và lẻ ở cuối mảng.

Ví dụ: Mảng a: 3 2 7 5 9

Mảng b: 1 8 10 4 12 6

Mảng c: **6 12 4 10 8 2 3 7 5 9 1**

20. Nhập vào mảng A gồm n phần tử, trong quá trình nhập kiểm tra các phần tử nhập vào không được trùng, nếu trùng thông báo và yêu cầu nhập lại.

21. Tính tổng của từng dãy con giảm trong mảng.

22. Chỉ ra số hạng lớn thứ k của mảng.

Ví dụ: Mảng a: 6 3 1 10 11 18

k = 2

Kết quả: 11

23. Liệt kê các bộ 4 số a, b, c, d trong mảng các số nguyên (có ít nhất 4 phần tử và đôi một khác nhau) sao cho $a + b = c + d$.

24. Tính trung bình cộng của các tổng các dãy tăng dần có trong mảng các số nguyên.

Ví dụ: 1 2 3 4 2 3 4 5 6 4 5 6 $\Rightarrow TB = 15$.

25. Tính tổng tất cả các phần tử xung quanh trên mảng các số nguyên. (Phần tử xung quanh là hai phần tử bên cạnh cộng lại bằng chính nó (Ví dụ: 1 3 2 \rightarrow 1,2 là hai phần tử xung quanh của 3).

Ví dụ: 1 3 2 5 3 9 6 \rightarrow tổng 17

26. Nhập vào hai số lớn a, b nguyên (a, b có từ 20 chữ số trở lên). Tính tổng, hiệu, tích, thương của hai số trên.

27. Tính tổng các phần tử là số Armstrong (số Armstrong là số có đặc điểm như sau: số có k ký số, tổng của các lũy thừa bậc k của các ký số bằng chính số đó.

Ví dụ: 153 là số có các ký số $1^3 + 5^3 + 3^3 = 153$ là một số Armstrong).

28. Tìm và xóa tất cả các phần tử trùng với x trong mảng một chiều các số nguyên, nếu không tồn tại phần tử x trong mảng thì trả về -1.

29. Tìm số lẻ nhỏ nhất lớn hơn mọi số chẵn có trong mảng, giá trị chẵn nhỏ nhất nhỏ hơn mọi giá trị lẻ trong mảng các số nguyên.

30. Tìm mảng con tăng dần có tổng lớn nhất trong mảng một chiều.

31. Tách 1 mảng các số nguyên thành 2 mảng a và b, sao cho kết quả thu được là:

- Mảng a chứa toàn số lẻ tăng dần.

- Mảng b chứa toàn số chẵn giảm dần.

(Không dùng sắp xếp)

Hướng dẫn: Tìm vị trí chèn thích hợp khi trích phần tử từ mảng ban đầu.

Ví dụ: Mảng ban đầu: 9 3 8 2 7 5 1 0 10

Mảng a: 1 3 5 7 9

Mảng b: 10 8 2

32. In ra tam giác Pascal.

33. Sinh ngẫu nhiên mảng các số nguyên gồm 10.000 phần tử, mỗi phần tử có giá trị từ 0 đến 32.000 và xây dựng hàm thống kê số lần xuất hiện các phần tử trong mảng, sau đó cho biết phần tử nào xuất hiện nhiều lần nhất.

Ví dụ: Mảng: 5 6 11 4 4 5 4

5 xuất hiện 2 lần

6 xuất hiện 1 lần

11 xuất hiện 1 lần

4 xuất hiện 3 lần

34. Cho mảng A có n phần tử. Nhập vào số nguyên k ($k \geq 0$), dịch phải xoay vòng mảng A k lần.

Ví dụ: Mảng A: 5 7 2 3 1 9

Nhập k = 2

Dịch phải xoay vòng mảng A: 1 9 5 7 2 3

35. Viết chương trình tìm kiếm tên trong chuỗi họ tên. Nếu có thì xuất ra là tên này đã nhập đúng, ngược lại thông báo là đã nhập sai.

36. Viết chương đảo vị trí của từ đầu và từ cuối.

Ví dụ: nhập “bo an co” xuất ra “co an bo”

37. Nhập một chuỗi bất kỳ, sau đó hỏi người dùng cần tách bắt đầu từ đâu trong chuỗi trở về sau.

Ví dụ: Nhập chuỗi S1: “trường Đại học Đà Lạt – Khoa Toán - Tin”. Người nhập muốn tách bắt đầu từ chữ “Khoa” thì sẽ xuất ra chuỗi “Khoa Toán - Tin”.

38. Kiểm tra xem trong chuỗi có ký tự số hay không nếu có tách ra thành một mảng số riêng.

39. Đảo ngược thứ tự các từ có trong chuỗi

Ví dụ: Nhập Truong CD CNTT TpHCM

Xuất ra màn hình là: TpHCM CNTT CD Truong

40. Nhập một ma trận:

- in ra những phần tử có ký số tận cùng là 5.
- in ra các phần tử nằm trên 2 đường chéo.
- in ra các phần tử nằm phía trên / dưới đường chéo phụ của ma trận vuông các số nguyên.
- in ra các phần tử nằm phía trên / dưới đường chéo chính của ma trận vuông các số nguyên.
- tính tổng các phần tử trên cùng một dòng, cùng một cột, các phần tử chẵn, số nguyên tố, nằm trên đường chéo chính của ma trận vuông.
- tính tổng các giá trị lớn nhất trên mỗi dòng, cột.
- tính giá trị trung bình của các phần tử nhỏ nhất trên mỗi cột.

41. Khởi tạo ma trận A chứa các số thực ngẫu nhiên có kích thước $m \times n$ và sắp xếp A:

- theo thứ tự tăng dần từ trên xuống dưới và từ trái qua phải.
- theo thứ tự giảm dần từ trên xuống dưới và từ trái sang phải.
- các dòng theo thứ tự tăng dần, các cột theo thứ tự giảm dần.

42. Nhập một ma trận A:

- Hoán vị 2 dòng / cột của A.
- Bỏ dòng i và cột j của A.

- c. Giả sử A là ma trận vuông có kích thước $n \times n$. Tách A thành 4 ma trận con $A^{1,1}$, $A^{1,2}$, $A^{2,1}$, $A^{2,2}$, sao cho:

$$A = \begin{bmatrix} A^{1,1} & A^{1,2} \\ A^{2,1} & A^{2,2} \end{bmatrix}$$

- d. phát sinh ma trận B là ma trận lật ngược của ma trận A.

Ví dụ:

$$\begin{array}{cccc} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{array} \Rightarrow \begin{array}{cccc} 4 & 3 & 2 & 1 \\ 8 & 7 & 6 & 5 \\ 12 & 11 & 10 & 9 \\ 16 & 15 & 14 & 13 \end{array}$$

- e. phát sinh ma trận B sao cho các phần tử của B là trung bình cộng các phần tử trong hình vuông 3×3 tâm tại (i, j) .

Ví dụ:

$$\begin{array}{ccc} 1 & 3 & 2 \\ 6 & 4 & 3 \\ 2 & 5 & 7 \end{array} \Rightarrow \begin{array}{ccc} 1 & 4 & 7 \\ 6 & 4 & 7 \\ 7 & 7 & 7 \end{array}$$

43. Nhập một ma trận A, in ra tất cả các đường chéo phụ / chính.

Ví dụ: các đường chéo phụ

$$\begin{array}{cccc} 1 & 3 & 7 & 4 \\ 9 & 5 & 6 & 2 \\ 3 & 4 & 7 & 5 \\ 2 & 3 & 1 & 6 \end{array} \Rightarrow \begin{array}{cccc} 1 & & & \\ 9 & 3 & & \\ 3 & 5 & 7 & \\ 2 & 4 & 6 & 4 \\ 3 & 7 & 2 & \\ 1 & 5 & & \\ 6 & & & \end{array}$$

44. Cho một mảng các phân số (**PHANSO**) gồm n phần tử ($n \leq 50$). Hãy viết chương trình nhập và xuất danh sách các phân số sau đó tìm phân số có giá trị lớn nhất, tổng và tích các phân số, nghịch đảo giá trị các phân số trong mảng và sắp xếp danh sách các phân số theo thứ tự giảm dần.

45. Tổ chức dữ liệu quản lý danh mục các bộ phim VIDEO, các thông tin liên quan đến bộ phim này như sau:

- Tên phim (tựa phim).
- Thể loại (3 loại : hình sự, tình cảm, hài).
- Tên đạo diễn.
- Tên diễn viên nam chính.
- Tên diễn viên nữ chính.
- Năm sản xuất.
- Hãng sản xuất

Viết chương trình thực hiện những công việc sau:

- Nhập vào bộ phim mới cùng với các thông tin liên quan đến bộ phim này.

- Nhập một thể loại. In ra danh sách các bộ phim thuộc thể loại này.
- Nhập một tên nam diễn viên. In ra các bộ phim có diễn viên này đóng.
- Nhập tên đạo diễn. In ra danh sách các bộ phim do đạo diễn này dàn dựng.

46. Một cửa hàng cần quản lý các mặt hàng, thông tin một mặt hàng bao gồm:

- Mã hàng.
- Tên mặt hàng.
- Số lượng.
- Đơn giá.
- Số lượng tồn.
- Thời gian bảo hành (tính theo đơn vị tháng).

Viết chương trình thực hiện những công việc sau:

- Hãy nhập vào một danh sách các mặt hàng.
- Tìm mặt hàng có số lượng tồn nhiều nhất.
- Tìm mặt hàng có số lượng tồn ít nhất.
- Tìm mặt hàng có giá tiền cao nhất.
- In ra những mặt hàng có thời gian bảo hành lớn hơn 12 tháng.
- Sắp xếp các mặt hàng theo thứ tự tăng dần của số lượng tồn.

47. Viết chương trình quản lý vé tàu, thông tin một vé tàu như sau :

- Ngày giờ khởi hành, ngày giờ đến.
- Ga đi, ga đến.
- Loại tàu, loại chỗ ngồi (ngồi, nằm, cứng, mềm).
- Số toa, số ghế.

Viết chương trình thực hiện những công việc sau:

- nhập vào danh sách các vé tàu.
- In danh sách các vé tàu có ga đến là Huế.
- In danh sách các vé tàu có ga đến là Hà Nội và đi ngày 8/6/2005.
- Đếm xem có bao nhiêu khách đi tàu loại chỗ ngồi là nằm cứng.

48. Tính tiền điện hàng tháng của các hộ gia đình, thông tin các khách hàng như sau:

- Kỳ thu, từ ngày.....đến ngày.
- Tên khách hàng, mã khách hàng.
- Địa chỉ.
- Điện năng tiêu thụ (Kwh).

Viết chương trình thực hiện những công việc sau:

- Nhập vào danh sách các khách hàng.
- Xuất danh sách hoá đơn theo thứ tự tăng dần của điện năng tiêu thụ.
- Tính tiền điện của các khách hàng theo quy định sau.
 - 100 kw đầu tiên là 550 đ / kw
 - 50 kw tiếp theo là 900 đ / kw
 - 50 kw tiếp theo là 1210 đ / kw
 - Thuế 10 % trên tổng số tiền phải trả
- Tính tổng số tiền thu được của các khách hàng.

TÀI LIỆU THAM KHẢO

- [1] A. B. Downey, *How to think like a computer scientist, C++ Version, First Edition*, 1999.
- [2] H. M. Deitel, P. J. Deitel, *C++ How to Program*, Fifth Edition, Prentice Hall, 2005.
- [3] S. R. Davis, *C++ for Dummies 5th Edition*, Willey Publishing, Inc, 2004.
- [4] Trần Minh Thái, *Giáo trình bài tập kỹ thuật lập trình C*, Trường CD Công Nghệ Thông Tin Tp. Hồ Chí Minh.