Bài 3. Mảng và xâu kí tự kiểu C

Mục tiêu:

1. *Luyện tập sử dụng mảng và xâu kí tự dạng mảng (C-string)*
2. *Thử nghiệm lỗi tràn mảng/xâu*
3. *Luyện tập sử dụng các cấu trúc điều khiển để cài đặt một số thuật toán đơn giản: sắp xếp nổi bọt.*

Giới hạn: *chỉ được dùng mảng và C-string, không dùng C++ string hay thư viện stl (chẳng hạn vector, algorithm)*

Yêu cầu nộp bài: sản phẩm code phần A; ít nhất 03 bài phần C hoặc 01 bài (**).

A. Thực hành

1. ***Khởi tạo mảng một chiều***. Hãy viết một chương trình thử dùng các cách khai báo và khởi tạo mảng dưới đây. Với mỗi cách, nếu không có lỗi thì dùng vòng lặp in toàn bộ nội dung của mảng ra màn hình để xem kết quả khởi tạo, nếu có lỗi thì chuyển khai báo đó thành dạng comment
    a. Khai báo mảng trong **và** ngoài hàm main() và không khởi tạo giá trị cho mảng
    b. Khai báo và khởi tạo trong **và** ngoài hàm main() dạng int a[N] = {1, 2, 3, 4} với N lớn hơn **và** nhỏ hơn 4.
    c. Khai báo và khởi tạo trong **và** ngoài hàm main() dạng int a[ ] = {1, 2, 3, 4}.

    Hãy chạy đi chạy lại trường hợp khai báo bên trong hàm main() và không khởi tạo để quan sát hiện tượng mỗi lần chạy lại in ra một kết quả khác cho nội dung mảng. Rút kinh nghiệm tránh lỗi không khởi tạo giá trị cho biến.

2. ***Khởi tạo C-string***. Hãy viết một chương trình thử dùng các cách khởi tạo C string dưới đây, mỗi lần lại in string ở hai dạng: (1) lần lượt tất cả phần tử trong mảng, và (2) xâu kí tự, ra màn hình để xem kết quả khởi tạo và so sánh hai dạng output đó:
    a. Khai báo dạng mảng trong **và** ngoài hàm main() và không khởi tạo giá trị.
    b. Khai báo và khởi tạo trong **và** ngoài hàm main() dạng char a[N] = “abcd”; với N lớn hơn **và** nhỏ hơn 4, N bằng 4.
    c. Khai báo và khởi tạo trong **và** ngoài hàm main() dạng char a[ ] = “abcd”;. Kích thước mảng thực là bao nhiêu? hãy dùng hàm sizeof để tính.

    Mục tiêu: cần quan sát được hiệu ứng của kí tự null chặn cuối xâu kí tự khi in ra ở dạng string: string có thể ngắn hơn mảng, có thể dài hơn kích thước khai báo của mảng (tràn ra ngoài). Mảng kích thước N lưu được string độ dài tối đa là bao nhiêu?

3. ***Khởi tạo mảng nhiều chiều***. Hãy viết một chương trình thử dùng các cách khởi tạo mảng dưới đây.

```
char daytab[2][12] = {
  {31,28,31,30,31,30,31,31,30,31,30,31},
```

```
  {31,29,31,30,31,30,31,31,30,31,30,31}
};
char daytab[2][12] = {
  31,28,31,30,31,30,31,31,30,31,30,31,
  31,29,31,30,31,30,31,31,30,31,30,31
};
```

a. Với mỗi cách, in mảng thành dạng bảng 2 dòng 12 cột để xem hiệu quả khởi tạo.
b. Với mỗi cách trên, hãy thử chỉ khởi tạo một phần của mảng bằng cách xóa bớt các giá trị khởi tạo xem kết quả như thế nào
c. Với mỗi cách ban đầu, hãy thử bỏ giá trị kích thước (số dòng / số cột / cả hai) trong khai báo mảng (tương tự tại khai báo int a[ ] = {1, 2, 3, 4}) xem kết quả như thế nào

4. ***Tràn mảng***. Hãy viết một chương trình thử nghiệm hậu quả của việc truy nhập ra ngoài mảng. Gợi ý: khai báo một mảng và một số biến (nên cùng kiểu char để dễ quan sát hiệu ứng).
    a. *Đọc tràn*: Dùng một chuỗi giá trị liên tiếp, gán trị cho các phần tử trong mảng và các biến nói trên (nên gán bằng khởi tạo). Hãy thử đọc giá trị các phần tử mảng có chỉ số -1, N, N+1… và in ra màn hình. Có bị lỗi không? có dấu hiệu gì có vẻ như bạn đang đọc trúng các biến không phải mảng hay không? Thử thay đổi thứ tự khai báo các biến và mảng xem sao.
    b. *Ghi tràn*: (Không bắt buộc và bạn tự chịu trách nhiệm về hậu quả) gán trị cho các phần tử mảng với chỉ số -1, N, N+1… sau đó in giá trị của các biến được khai báo xung quanh mảng ra màn hình xem sao. Nếu việc ghi tràn không thấy lỗi hệ thống, chẳng hạn chương trình của bạn bị sập, thì bạn có thể thấy các biến đó bị sửa giá trị.

5. ***Tràn string.*** Viết một chương trình thử nghiệm việc đọc string quá kích thước được khai báo. Khai báo một C-string kích thước N, dùng cin để đọc từ bàn phím vào C-string đó rồi in nó ra màn hình (nhớ chặn đuôi bằng một kí tự đặc biệt để ta có thể nhìn thấy cuối string, chẳng hạn cout << "_" << str << "_";).
   Hãy chạy thử: nhập một chuỗi kí tự (không chứa kí tự trắng như space, tab, newline vì cin sẽ ngừng đọc tại đó) có độ dài nhỏ hơn N, bằng đúng N, và lớn hơn N.

6. ***Sắp xếp nổi bọt.*** Viết một chương trình lần lượt thực hiện các bước sau: (1) tạo một mảng chứa 30 số ngẫu nhiên trong khoảng từ 1 đến 100. (2) In mảng ra màn hình trên 1 dòng. (3) dùng thuật toán sắp xếp nổi bọt (xem trong bài giảng) để sắp xếp mảng theo thứ tự tăng dần, (4) in mảng kết quả ra màn hình trên một dòng.
   Gợi ý cách sinh số ngẫu nhiên: hàm rand() của thư viện cstdlib trả về một giá trị "ngẫu nhiên" trong khoảng từ 0 đến RAND_MAX (giá trị cụ thể tùy từng cài đặt thư viện, ít nhất là 32767). Bạn có thể dùng kết quả trả về của hàm này để tạo ra giá trị cho các phần tử mảng, dùng phép đồng dư % để lấy được giá trị trong một khoảng mong muốn.

Giá trị trả về của rand() thực ra chỉ là giả ngẫu nhiên. Tuy chuỗi kết quả của một chuỗi lệnh gọi rand() có phân bố ngẫu nhiên (bạn có thể thấy mảng được tạo ra trông khá ngẫu nhiên), nhưng từng giá trị lại bị quyết định bởi hạt giống (seed) mà ta chọn khi gọi hàm srand(giá trị hạt giống) trước khi bắt đầu chạy chuỗi rand(). Hãy thử dùng một seed cố định, chẳng hạn **srand(2016);**, bạn sẽ thấy lần nào bạn cũng tạo ra một mảng giống hệt lần chạy trước.

Để có chương trình thực sự ngẫu nhiên, mỗi lần chạy lại sinh ra một mảng khác lần trước, bạn cần dùng giá trị hạt giống khác nhau cho mỗi lần chạy. Một lựa chọn phổ biến là dùng thời gian hiện tại làm seed cho srand(): **srand (time(NULL))**;.Hàm time() cần include thư viện ctime.
(Xem thêm tại: http://www.cplusplus.com/reference/cstdlib/rand/)

B. Câu hỏi

1. Khai báo nào sau đây là hợp lệ:
   i. double a[];
   ii. double[] a;
   iii. double a{5};
   iv. double a[5];
2. Khai báo nào sau đây là hợp lệ
   i. int a[3] = { 1, 5, 8, 4};
   ii. int a[5] = { 1, 5, 8, 4};
   iii. int[] a = { 1, 5, 8, 4};
   iv. int a[] = { 1, 5, 8, 4};
3. Khai báo nào sau đây là khai báo một mảng 2 chiều
   i. array a[20][20];
   ii. int a[20][20];
   iii. int a[20, 20];
   iv. char a[20];
4. Mảng foo có 100 phần tử, câu lệnh nào sau đây dung truy cập phần tử thứ 7:
   i. foo[6];
   ii. foo[7];
   iii. foo(7);
   iv. foo{6};
5. Câu lệnh nào sau đây trả về địa chỉ của phần tử đầu tiên trong mảng foo
   i. foo[0];
   ii. &foo[1];
   iii. &foo;
   iv. foo;
6. Hai chỉ số [5] & [4] trong int a[5]; & a[4]++; diễn tả:
   i. Cả hai chỉ số diễn tả độ dài của mảng a
   ii. Cả hai chỉ số diễn tả chỉ số phẩm tử của mang a
   iii. [5] diễn tả độ dài của mảng a & [4] diễn tả chỉ số phần tử của mảng a

iv. [5] diễn tả chỉ số phần tử của mảng a & [4] diễn tả độ dài của mảng a

7. Mảng được truyền như một đối số cho hàm được hiểu như thế nào
   i. Mảng được truyền
   ii. Giá trị của phần tử đầu tiên của mảng được truyền
   iii. Địa chỉ của phần tử đầu tiên của mảng được truyền
   iv. Số phần tử của mảng được truyền.
8. Những phát biểu nào xong đây là chính xác
   i. Độ dài của mảng phải xác định ngay khi khai báo
   ii. Không thể thay đổi độ dài của mảng bằng cách khai báo lại mạng
   iii. Khi truyền mảng cho hàm, độ dài của mảng cũng cần được truyền theo
   iv. Tất cả các phát biểu trên
9. Có lỗi nào xảy ra với đoạn chương trình sau

```
int sampleArray[10];
for (int index = 1; index <= 10; index++)
        sampleArray[index] = 3*index;
```

   i. Lỗi khi dịch: chưa khởi tạo mảng
   ii. Lỗi khi dịch: truy cập phần tử vượt khoảng cho phép
   iii. Lỗi khi chạy: truy cập phần tử vượt khoảng cho phép
   iv. Không có lỗi nào trong khi dịch và chạy
10. Những khai báo nào xong đây là chính xác
   i. void clearBoard(char aChar[][]);
   ii. void clearBoard(char[][] aChar);
   iii. void clearBoard(char[][10] aChar);
   iv. void clearBoard(charaChar[][10]);
11. Cho hàm tripler và 2 khai báo như sau, lệnh gọi hàm nào không đúng

```
void tripler(int& n){ n = 3*n; }
int a[3] = {4, 5, 6}, number = 2;
```

   i. triple(a[2]);
   ii. triple(a[number]);
   iii. triple(a);
   iv. triple(number);
12. Đoạn mã sau in ra như thế nào:

```
double a[3] = {1.1, 2.2, 3.3};
cout <<a[0]<<" "<<a[1]<<" "<<a[2]<< endl; a[1] = a[2];
cout <<a[0]<<" "<<a[1]<<" "<<a[2]<< endl;
```

   i. 1.12.23.3
      1.12.23.3
   ii. 1.1 2.2 3.3
      1.1 2.2 3.3
   iii. 1.1 2.2 3.3
      2.2 2.2 3.3
   iv. 1.1 2.2 3.3
      1.1 3.3 3.3
13. Đoạn mã sau in ra như thế nào:

```
int a[3] = {5, 10, 15};
for (int i = 2; i >= 0; i--)
        cout << a[i] << " ";
```

i. 5 10 15
ii. 15 10
iii. 15 10 5
iv. 2 1 0

14. Sau khi chạy đoạn mã sau giá trị của phần tử thứ 2 trong mảng là gì:

```
int a[5];
for (int i = 0; i <5; i++)
{
  a[i] = i + 2;
  if (i >= 2) a[i-1] = a[i] + 3;
}
```

i. 2
ii. 3
iii. 7
iv. 8

15. Mảng a có 100 phần tử, lệnh nào có thể dùng để in giá trị a[0], a[2], a[4],…

i. for (i=0; i<5000; i=i+2) cout << beta[i] << endl;
ii. for (i=0; i<2500; i++) cout << beta[i*2] << endl;
iii. for (i=0; i<2500; i++) cout << beta[i]*2 << endl;
iv. tất cả các đoạn lệnh trên

16. Đếm số lần xuất hiện các ký tự ASCII (256 ký tự), khai báo nào là hợp lý

i. int freq[char];
ii. char freq[256];
iii. int freq[256];
iv. char freq[int];

17. Cho đoạn chương trình sau, khai báo hàm nào là hợp lý nhất:

```
char a[200];
char b[200];
copy(a, b, 200); // sao toàn bộ nội dung mảng b sang mảng a
```

i. copy(char a1[], char a2[], int size);
ii. copy(const char a1[], char a2[], int size);
iii. copy(const char a1[], const char a2[], int size);
iv. copy(char a1[], const char a2[], int size);

18. Cho khai báo hàm như sau, phát biểu nào không chính xác:

```
void SapXep(int a[], int N);
```

i. Độ dài của mảng truyền vào là N
ii. Mảng truyền vào có N phần tử
iii. Phải truyền vào độ dài của mảng trong ngoặc vuông [] cùng tên mảng
iv. Mảng được truyền vào theo kiểu truyền giá trị

19. Khai báo hàm cho việc đổi chỗ 2 phần tử của mảng trong quá trình sắp xếp, khai báo nào sau đây là đúng:

i. int swap(int a, int b);
ii. void swap(int a, int b);
iii. int swap(int &a, int &b);
iv. void swap(int &a, int &b);

20. Truyền mảng 2 chiều cho hàm, khai báo nào sau là đúng:
i. void timkiem(int a[][]);
ii. void timkiem(int a[10][]);
iii. void timkiem(int a[][10]);
iv. void timkiem(int[10][10] a);

C. Bài tập

1. ***Tìm lặp.*** Viết chương trình nhập một số nguyên dương N <= 10000 và một chuỗi gồm N số trong khoảng từ 1 đến N, xác định xem trong chuỗi đó hai số nào bằng nhau hay không. Nếu có thì in ra "Yes", nếu không thì in ra "No".
Gợi ý: ý tưởng là dùng một tập hợp ban đầu rỗng, duyệt từng số trong chuỗi, nếu số đó chưa có trong tập hợp thì bỏ nó vào trong tập hợp, nếu nó có rồi nghĩa là chuỗi có ít nhất hai số cùng giá trị → kết luận có trùng. Duyệt đến hết chuỗi rồi mà chưa kết luận trùng nghĩa là không có lặp. Làm thế nào để mô hình hoá một tập hợp các số từ 1 đến N? Dùng mảng bool, seen[i] = true nghĩa là đã gặp giá trị i rồi.

2. ***Xâu đối xứng.*** Viết chương trình nhập từ bàn phím một xâu kí tự độ dài tối đa 100, sau đó kiểm tra xem xâu kí tự đó có đối xứng hay không. Chẳng hạn "abcba", "abba" là đối xứng, còn "abcda" không đối xứng. Nếu có thì in ra "Yes", nếu không thì in ra "No".
Gợi ý: cho i chạy từ đầu xâu sang phải, đồng thời j chạy từ cuối xâu sang trái cho đến khi i và j gặp nhau hoặc khi s[i] != s[j]. Kết luận tuỳ theo tình trạng kết thúc.

3. ***Số đối gương.*** Cho các số A và B, đếm số các số nguyên N sao cho A <= N <= B và N là một số đối gương.
Ví dụ, các số đối gương: 121, 11, 11411
Các số không đối gương: 122, 10
Input:
Dữ liệu đọc từ input chuẩn (bàn phím). Dòng đầu chứa số tự nhiên T là số test case, N dòng tiếp theo, mỗi dòng chứa hai giá trị A và B trên cùng một dòng
Output:
Với mỗi test case, in giá trị cần thiết trên một dòng.
Ràng buộc: 1 <= T <= 100
0 <= A <= B <= 10^5

Sample Input	Sample Output
2	1

10 13 20 30	1

Nguồn: http://www.hackerearth.com/problem/algorithm/palindromic-numbers-7/ (chạy test tại đó)

4. ***Dò mìn***. Viết chương trình nhận input từ bàn phím gồm một cặp số m, n nguyên dương nhỏ hơn 10 và một bản đồ mìn dạng bảng, * nghĩa là có mìn, . là không có mìn, các kí tự cách nhau bởi 01 space. In ra màn hình một bảng m dòng n cột sao cho giá trị của mỗi ô là một dấu * nếu như ở đó có mìn, nếu không thì là một giá trị trong khoảng 0...8 là số mìn nằm trong các ô xung quanh. Ví dụ

Sample Input	Sample Output
3 5 * * . . . . * . . .	* * 1 0 0 3 3 2 0 0 1 * 1 0 0

Thảo luận: Thuật toán rất đơn giản, duyệt từng ô, với mỗi ô chỉ việc kiểm tra có mìn hay không và nếu không thì đếm số mìn xung quanh. Bài này chỉ phải nghĩ chút xíu về cách lưu trữ input. Bạn định dùng cách nào? m xâu kí tự? mảng char [m][n]? int [m][n]?... hãy chọn cách phù hợp nhất cho thuật toán mà bạn sẽ dùng.

5. ***Bảng xoắn ốc***. Viết chương trình nhận input từ bàn phím gồm một cặp số x, y nguyên dương nhỏ hơn 10 và in ra màn hình một bảng x dòng y cột chứa các giá trị từ 1 đến x*y theo dạng xoắn ốc. Ví dụ

Sample Input	Sample Output
4 5	1 2 3 4 5 14 15 16 17 6 13 20 19 18 7 12 11 10 9 8

6. ***Ma trận kì ảo.*** Viết chương trình nhập một số nguyên dương lẻ N và in ra ma trận kì ảo kích thước N * N. Ma trận kì ảo là bảng vuông chứa các số từ 1 đến N*N có tính chất như sau: mỗi số xuất hiện đúng một lần, tổng các số trên mỗi hàng, cột, đường chéo chính, đường chéo phụ đều bằng nhau.
Ví dụ

N = 3	N = 5
8 1 6 3 5 7	17 24 1 8 15 23 5 7 14 16

```
4 9 2                                  4  6 13 20 22
                                      10 12 19 21  3
                                      11 18 25  2  9
```

Gợi ý: thuật toán đơn giản như sau, lần lượt xếp các số theo thứ tự tăng dần từ 1 đến N*N vào bảng vuông. Bắt đầu bằng số 1 tại ô nằm chính giữa hàng trên cùng, với mỗi số vừa được xếp, xếp số tiếp theo vào ô kề phía trên bên phải ô vừa xếp (hướng đông bắc), nếu ô đó đã bị điền rồi thì điền vào ô kề dưới (hướng nam). Xem chi tiết tại https://en.wikipedia.org/wiki/Siamese_method.
Để cho chỉ số mảng chạy vòng quanh mảng khi tăng dần hoặc giảm dần, nên dùng phép đồng dư. Chẳng hạn, thay vì i=i+1, dùng i = (i+1) % N, i sẽ chạy đến cuối mảng rồi vòng về đầu mảng theo kiểu ...N-2, N-1, 0, 1, …

7. ***Banner.*** Viết một chương trình nhập vào một từ tiếng Anh độ dài không quá 12 chữ cái và hiển thị lại từ đó ra màn hình ở dạng "chữ to" được vẽ bằng nhiều dòng text. Ví dụ: với input là "Kevin" thì output là:

```
#    #  ######  #    #      #    #    #
#   #   #       #    #      #    ##   #
####     #####   #    #      #    # #  #
#  #     #       #    #      #    #  # #
#   #    #       #  #  #      #    #   ##
#    #   ######    ##        #    #    #
```

Bạn tự thiết kế font chữ nhé. Nếu có nhu cầu đổi chữ thường thành chữ hoa thì có thể dùng hàm toupper() trong thư viện cctype - xem link tại
http://www.cplusplus.com/reference/cctype/
Gợi ý: bắt chước cách hiển thị các hình đồ họa: dùng một mảng hai chiều đại diện cho màn hình (6 dòng và bao nhiêu cột?) "vẽ" từng chữ cái tại vị trí của nó trên "màn hình", sau khi vẽ xong thì in "màn hình" ra output chuẩn.

8. ***Lịch khám bệnh (**)***. Một bác sĩ có N bệnh nhân đến khám. Bác sĩ khám cho bệnh nhân thứ i sẽ hết một khoảng thời gian Ti. Cả n bệnh nhân đều có mặt ở phòng khám từ đầu buổi khám bệnh. Hãy giúp bác sĩ tìm cách sắp xếp một thứ tự khám cho các bệnh nhân để tổng thời gian chờ đợi của N bệnh nhân là ít nhất. In ra tổng thời gian chờ đợi nếu dùng thứ tự đó.
Input: từ bàn phím: dòng đầu là số N, N dòng sau là N số nguyên là khoảng thời gian khám tính bằng phút cho các bệnh nhân, lần lượt từ bệnh nhân thứ nhất đến bệnh nhân thứ N.
Output: ra màn hình: 01 số nguyên là tổng thời gian đợi với lịch khám tối ưu mà bạn tìm được.
Giới hạn dữ liệu: Ti trong khoảng kiểu int. Còn N? Bạn thử xem bubblesort ở máy bạn chạy được N tối đa bao nhiêu nếu thời gian chạy không quá 5 giây.

9. Viết chương trình nhập vào một dãy số. In ra giá trị trung bình (μ) và phương sai (σ^2) của dãy số đó.

10. Viết chương trình nhập vào một số N là số phần tử của dãy số, sau đó nhập vào một dãy số nguyên gồm N phần tử. In ra:
    a. Phần tử nhỏ nhất
    b. Phần tử lớn nhất
    c. Tổng các phần tử chẵn
    d. Số lượng các phần tử lẻ

11. Viết chương trình nhập vào một dãy số gồm N phần tử, mỗi phần tử trong khoảng từ 0 đến 9. Đếm số lượng các sô 0, 1, 2, …, 8, 9 trong dãy số đó.

12. Viết chương trình in ra tam giác Pascal bậc n, ví dụ với n = 6

```
1
1  1
1  2  1
1  3  3  1
1  4  6  4  1
1  5 10 10  5  1
```

Cần căn lề các cột.

13. Viết chương trình đọc vào điểm Toán, Lý & Anh của các sinh viên. Chương trình cần kiểm tra điểm số trong khoảng 0-10. Sắp xếp danh sách sinh viên dựa trên điểm Toán, Anh và trung bình, ví dụ:

Theo điểm môn Toán:

	Toán	Lý	Anh
Sinh Vien 1	9.0	6.0	7.5
Sinh Vien 3	8.0	8.5	6.0
Sinh Vien 2	6.0	7.0	4.5

Theo điểm môn Lý:

	Toán	Lý	Anh
Sinh Vien 3	8.0	8.5	6.0
Sinh Vien 2	6.0	7.0	4.5
Sinh Vien 1	9.0	6.0	7.5

Theo điểm trung bình:

	TB	Toán	Lý	Anh
Sinh Vien 1	7.5	9.0	6.0	7.5
Sinh Vien 3	7.5	8.0	8.5	6.0
Sinh Vien 2	5.8	6.0	7.0	4.5

D. Bài tập lớn

Đưa ra tình huống, bạn đi siêu thị với một danh sách hàng hóa và số lượng cần mua. Bạn mang theo một số tiền nhất định. Nếu mang theo đủ tiền bạn có thể mua toàn bộ lượng hàng hóa trong danh sách. Tuy nhiên nếu không mang theo đủ tiền, bạn phải quyết định mua mặt hàng nào. Đồng thời khi đã quyết định mua mặt hàng nào, bạn sẽ mua toàn bộ số lượng của mặt hàng đó trong danh sách. Trong trường hợp không đủ tiền, bạn có thể mua một phần số lượng của mặt hàng cuối cùng quyết định mua trước khi hết tiền.

1. Viết các hàm sau:
    i. Tính tổng tiền mỗi mặt hàng cần mua:
        double xxx(int SoLuong, double GiaThanh)
    ii. Kiểm tra xem có đủ tiền mua hàng không:
        bool xxx(double Tien, int SoLuong[], double GiaThanh[], int)
2. Trong trường hợp không đủ tiền, bạn có thể mua hàng theo 2 phương án sau:
    i. Mua tất cả mặt hàng có giá thành trên một đơn vị sản phẩm giá rẻ nhất
    ii. Mua tất cả mặt hàng có tổng giá trị (trên danh sách cần mua) là rẻ nhất

Viết các hàm cần thiết để giải quyết 2 trường hợp trên. Chú ý danh sách mặt hàng bao gồm tên, giá thành, số lượng, giá trị mỗi mặt hàng phải được in ra và được sắp xếp theo thứ tự tên sản phẩm (A, B, C,…)

Có một số mặt hàng thiết yếu trong danh sách phải mua trước khi quyết định mua các mặt hàng khác. Viết các hàm mới dựa trên các hàm của 2.i và 2.ii

Sửa đổi và/hoặc viết thêm hàm để đáp ứng được bài toán thực tế là số lượng sản phẩm có thể thuộc kiểu số thực (double) ví dụ 0,5kgs khoai tây. Giả sử số lượng những sản phẩm kiểu này được làm tròn đến 1 chữ số phần thập phân còn giá mỗi đơn vị sản phẩm (/1chiếc, /0.1kgs) và số tiền mang theo được làm tròn đến đơn vị nghìn đồng