#include <map>

map<char, bool> letters;

letters[input[i]] = true;

#include <set>

set<string> setIns;

set<int, greater<int> > s1; // descending order

#include <unordered\_map>

unordered\_map<string, int> umap;

#include <stack>

stack<int> stackArr;

stackArr.push(arr[i]);

int a = stackArr.top();

stackArr.pop();

#include <queue>

queue<int> queueArr;

queueArr.push(input);

queueArr.empty()

queueArr.front()

queueArr.pop();

Array2Ddynamis.cpp

#include <stdlib.h>  // malloc

#include <string.h>  // memcpy, memset

#include <algorithm> // fill

using namespace std;

void foo(int \*\*x)

{

    printf("%d", x[2][3]);

}

int main()

{

    int i;

    // === PAKAI C++ NEW ===

    int \*\*arr = new int \*[5];

    for (i = 0; i < 5; i++)

        arr[i] = new int[10];

    for (i = 0; i < 5; i++)

        delete[] arr[i];

    delete[] arr;

    // === PAKAI MALLOC ===

    // int \*a = (int \*)malloc(sizeof(int) \* 10);

    int \*\*a = (int \*\*)malloc(sizeof(int \*) \* 10); // dynamis 2d array with malloc

    for (i = 0; i < 10; i++)

        a[i] = (int \*)malloc(sizeof(int \*) \* 20);

    // ...

    // membebaskan memory dari a

    for (i = 0; i < 10; i++)

        free(a[i]);

    free(a);

    return 0;

}

#include <stdlib.h>  // malloc

#include <string.h>  // memcpy, memset

#include <algorithm> // fill

using namespace std;

int main()

{

    int a[5]; // array statis -> stack mamory

    printf("%d\n", sizeof(a)); // keluar 20 byte (int size dikali 5)

    int n = 5;

    int \*arr = (int \*)malloc(sizeof(int) \* n); // buat 4 \* 5 byte pake heap memory

    printf("%d\n", sizeof(arr)); // keluar size dari pointer arr (8 byte)

    // malloc keluar void\* (void pointer)

    int \*arr3 = new int[n]; // cari c++ lebih lambat

    fill(arr3, arr3 + n, 1234);

    memset(arr, -1, sizeof(int) \* n); // inisialisasi nilai arr 0 per byte paling cepat

    // for (int i = 0; i < n; i++)

    // {

    //     arr[i] = i \* 10;

    // }

    int \*arr2 = (int \*)malloc(sizeof(int) \* n);

    // printf("%d\n", sizeof(arr));

    memcpy(arr2, arr3, sizeof(int) \* n); // copy arr to arr2 with low level

    for (int i = 0; i < n; i++)

    {

        printf("%d ", arr2[i]);

    }

    return 0;

}

// memset, memcpy itu sangat cepat

POINTER

#include <stdio.h>

void foo(int \*p) // passing by reference

{

    \*p = \*p + 10;

    printf("%d\n", \*p);

}

void foo2(int x) // passing by value

{

    x = x + 10;

    printf("%d\n", x);

}

void foo3(int &x) // passing by reference (ga usah pake aja)

{

    x = x + 10;

    printf("%d\n", x);

}

int main()

{

    int angka = 100;

    foo(&angka);

    foo2(angka); // angka tidak berubah

    foo3(angka);

    printf("angka: %d\n", angka);

    return 0;

}

#include <stdio.h>

#include <vector>

#include <algorithm>

using namespace std;

int mapSimbol[256], mapNilai[256];

class Card

{

public:

    char text[3];

    // int simbol, nilai;

    int value;

    Card(char s, char n)

    {

        // simbol = mapSimbol[s];

        // nilai = '2' <= n && n <= '9' ? n - '0' : mapNilai[n];

        value = mapSimbol[s] \* 15 + ('2' <= n && n <= '9' ? n - '0' : mapNilai[n]);

        text[0] = s;

        text[1] = n;

        text[2] = 0;

    }

    bool operator<(const Card &other) const;

};

int main()

{

    mapSimbol['C'] = 0;

    mapSimbol['D'] = 1;

    mapSimbol['S'] = 2;

    mapSimbol['H'] = 3;

    mapNilai['T'] = 10;

    mapNilai['J'] = 11;

    mapNilai['Q'] = 12;

    mapNilai['K'] = 13;

    mapNilai['A'] = 14;

    char s, n;

    char first;

    int i, j;

    // while (scanf("%c\n", &dealer) && dealer != '#')

    while (scanf("%c\n", &first), first != '#')

    {

        switch (first)

        {

        case 'N':

            j = 3;

            break;

        case 'E':

            j = 0;

            break;

        case 'S':

            j = 1;

            break;

        case 'W':

            j = 2;

            break;

        }

        vector<Card> hand[4];

        for (i = 0; i < 52; i++)

        {

            scanf("%c%c\n", &s, &n);

            hand[j].push\_back(Card(s, n));

            j = (j + 1) % 4;

        }

        char arr[] = {'S', 'W', 'N', 'E'};

        for (j = 0; j < 4; j++, printf("\n"))

        {

            printf("%c:", arr[j]);

            sort(hand[j].begin(), hand[j].end());

            for (i = 0; i < hand[j].size(); i++)

                printf(" %s", hand[j][i].text);

        }

    }

    return 0;

}

bool Card::operator<(const Card &other) const

{

    // if (simbol == other.simbol)

    //     return nilai < other.nilai;

    // return simbol < other.simbol;

    return value < other.value;

}