**NAME: PHẠM QUỐC DUY**

**STUDENT ID: 21127031**

**WEEK 4 HOMEWORK**

**4-7:**

Thân hàm như sau:

*char* *\**nonwhite (*char* *\**str)

{

*char* c;

    if (\*str == '\0') return NULL;

    while (\*str != '\0')

    {

        c = \*str;

        if ( c == '\t' || c == ' ') str++;

        else return str;

    }

    return NULL;

}

Nếu xâu cần quét là xâu rỗng hoặc không có kí tự nào là không phải là ký tự không phải khoảng trắng thì hàm sẽ trả về NULL, còn lại sẽ trả về con trỏ trỏ đến ký tự đầu tiên không phải là khoảng trống

**4-8:**

Ba phiên bản của hàm swap sẽ được triển khai cùng thân hàm main như sau:

1. Version 1: Swap by pointer
   1. Hàm:

*void* Swap1(*int* *\**a, *int* *\**b)

{

*int* c = \*a;

    \*a = \*b;

    \*b = c;

}

* 1. Hàm main:

*int* main()

{

*int* i=5, j=7;

*int* \*pi = &i;

*int* \*pj = &j;

    Swap1(pi,pj);

    cout << i << " " << j;

}

1. Version 2: Swap by reference
   1. Hàm:

*void* Swap2(*int* *&*a, *int* *&*b)

{

*int* c = a;

    a = b;

    b = c;

}

* 1. Hàm main:

*int* main()

{

*int* i=5, j=7;

    Swap2(i,j);

    cout << i << " " << j;

}

1. Version 3: Swap pointer by reference
   1. Hàm:

*void* Swap3(*int* *\*&*a, *int* *\*&*b)

{

*int* \*c = a;

    a = b;

    b = c;

}

* 1. Hàm main:

*int* main()

{

*int* i=5, j=7;

*int* \*pi = &i;

*int* \*pj = &j;

    Swap3(pi,pj);

    cout << \*pi << " " << \*pj;

}

**4-9:**

Với đoạn code cout << &a[2] << “ “ << a <<endl;

Output sẽ lần lượt là địa chỉ của phần tử a[2] của mảng và địa chỉ của phần tử a[0] (a) của mảng

Với đoạn code cout << &a[2] – a;

Output sẽ là 2 vì: phép trừ hai địa chỉ này sẽ cho ra độ chênh lệch giữa vị trí của hai phần tử trong mảng được tính như sau: (add1 – add2) / byteofdatatype. Tại đây, mảng là float a[3] = {1,2,3} nên độ lệch giữa hai địa chỉ là (2\*4 – 0) / 4 = 2

Tương tự với mảng double b[3] = {1,2,3} phía dưới, đoạn code cout << &b[2] << “ “ << b << endl; sẽ lần lượt xuất ra địa chỉ của phần tử b[2] và phần tử b[0] (b)

Đoạn code cout << &b[2] – b; sẽ xuất ra 2 vì theo tương tự như trên, độ lệch vị trí giữa b[2] và b[0] là (2\*8 – 0) / 8 = 2

**4-11:**

**Hàm Sort và các hàm liên quan sẽ được triển khai như sau:**

*int\** Sort(*int* *\**arr, *int* n,*int\** (\*funccall)(*int* *\**, *int*))

{

*int* \*exe;

    exe = (\*funccall)(arr, n);

    return exe;

}

*int\** Ascending(*int* *\**arr, *int* n)

{

    for (*int* i=0; i<n; ++i)

    {

        for (*int* j=i+1; j<n; ++j)

        {

            if (\*(arr+i) > \*(arr+j)) swap(\*(arr+i), \*(arr+j));

        }

    }

    return arr;

}

*int* *\**Descending(*int* *\**arr, *int* n)

{

    for (*int* i=0; i<n; ++i)

    {

        for (*int* j=i+1; j<n; ++j)

        {

            if (\*(arr+i) < \*(arr+j)) swap(\*(arr+i), \*(arr+j));

        }

    }

    return arr;

}

Các hàm Ascending và Descending sẽ trả con trỏ về vị trí đầu tiên của mảng sau khi sắp xếp xong. Khi truyền qua hàm Sort, con trỏ exe sẽ được truyền qua các hàm tùy lựa chọn của người dùng.

**Các hàm khác bổ trợ cho chương trình:**

*int* *\**init(*int* *&*n)

{

    cout << "Number of integers you want to store: ";

    cin >> n;

*int* \*p;

    p = new (nothrow) *int*[n];

    if (p == NULL)

    {

        cout << "Memory allocation error";

    }

    else

    {

        for (*int* i=0; i<n; ++i)

        {

            cout << "Enter number: ";

            cin >> \*(p+i);

        }

    }

    return p;

}

*void* PrintArr(*int* *\**arr, *int* n)

{

    for (*int* i=0; i<n; ++i)

    {

        cout << \*(arr+i) << " ";

    }

}

Hàm cấp phát bộ nhớ cho mảng arr sẽ trả về con trỏ NULL nếu như có lỗi trong việc cấp phát bộ nhớ, nếu cấp phát thành công thì tiến hành nhập mảng.

Hàm xuất các phần tử của mảng để theo dõi kết quả sau khi chạy chương trình.

**Hàm xuất menu và hàm main:**

*int* Menu()

{

*int* choice;

    cout << "                      ===========================ARRAY SORTING PROGRAM ==============================               " << endl;

    cout << "     ======================== CHOOSE THE CORRESPONDING NUMBER TO SORT THE ARRAY IN SUCH ORDER ===========           " << endl;

    cout << " 1. Sort the array in an increasing order   " << endl;

    cout << " 2. Sort the array in a decreasing order    " << endl;

    cin >> choice;

    return choice;

}

*int* main()

{

*int* n;

*int* \*arr = init(n);

*int* choice = Menu();

    if (choice == 1) arr = Sort(arr,n,Ascending);

    else arr = Sort(arr,n,Descending);

    PrintArr(arr,n);

    delete[] arr;

}

Hàm xuất menu cho phép người dùng chọn và ghi nhận lựa chọn của người dùng, ở hàm main, tùy vào lựa chọn của người dùng mà con trỏ đến hàm Ascending hay Descending được truyền vào hàm Sort. Sau khi hoàn tất, giải phóng bộ nhớ đã được cấp phát trước đó.

**4-12:**

1. char \*ex1
2. char \*\*ex2
3. char \*ex3[10];
4. char (\*ex4)[30]
5. char (\*ex5[10])[500]
6. int \* const ex6
7. const int \*ex7
8. char \*ptr (char \*pc)

{

return pc;

}

1. float \*Changefl (int \*pi)

{

return (float \*) pi;

}

1. void OddEven(int \*pi)

{

if ( (\*pi) % 2 == 0) cout << “The value pointed to by pi is even”;

else cout << “The value pointed to by pi is odd”;

}

1. Các hàm liên quan đến thao tác xử lý chính: hàm func1,2,3 để xuất các xâu tương ứng và hàm Imp để gọi tương ứng với giá trị được người dùng nhập

*char* *\**Imp (*char* *\**(\*funccall)())

{

*char* \*call = (\*funccall)();

    return call;

}

*char* *\**func1()

{

*char* \*str = "Catfish";

    Print(str);

    return str;

}

*char* *\**func2()

{

*char* \*str = "chrisname";

    Print(str);

    return str;

}

*char* *\**func3()

{

*char* \*str = "devonrevenge";

    Print(str);

    return str;

}

Các hàm thao tác phụ: hàm nhập lựa chọn và hàm in chuỗi:

*int* Choice()

{

*int* n;

    cout << "Pick a number between 1 and 3: ";

    cin >> n;

    while (n<1 || n>3)

    {

        cout << "Invalid choice, try again";

        cout << "Pick a number between 1 and 3: ";

        cin >> n;

    }

    return n;

}

*void* Print (*char* *\**str)

{

    while (\*str != '\0')

    {

        cout << (\*str);

        str++;

    }

}