# Хаяган ба Заалтан хувьсагчийн тухай ойлголт (Лаборатори №2)

Д.Самдан (16B1SEAS2873)

ХШУИС, ПХ-ийн 3-р түвшин, pri.samdan@gmail.com

# 1. ОРШИЛ

Хаяган хувьсагч болон заалтан хувьсагчийн талаар судалж, тэдгээрийг хэрхэн ашиглах талаар өгүүлнэ. Энэ лабораторийн ажилаар заагч болон заалтыг ашиглан функц рүү хаягаар болон заалтан хувьсагчаар аргумент дамжуулана. Мөн хаяган хувьсагч ба new оператор ашиглан санах ой нөөцөлж авах боломжтой болсон ч зөв хэрэглэхгүй бол санах ойн цоорхой үүсгэх аюултай. Сүүлд нь inline функц гэж юу болохыг тайлбарлана.

# 2. ЗОРИЛГО

Бид өмнөх C хэлний хичээл дээр энгийн төрлийн хувьсагчууд буюу бүхэл тоо, бодит тоо, тэмдэгт илүү судалж байсан бол энэ лабораторийн ажлаар хаяган болон заалтан хувьсагчуудын талаар судалж, ашиглаж сурах.

1. Хаяган хувьсагчийн тухай судалж тодорхойлох
2. Заалтан хувьсагчийн тухай судалж тодорхойлох
3. Хаяган хувьсагчийг ашиглан хэрхэн санах ой нөөцлөх тухай
   1. Ойн операторуудыг судлах
      1. new болон delete операторуудын үүрэг ба хэрэглээ

3.2 Санах ойн цоорхойн талаар судлах

1. Классыг хэрэглэсэн функц тодорхойлох.

inline функцийн талаар судаlж бичнэ inline функцийн давуу болон сүл талаад сурах.

# 3. ОНОЛЫН СУДАЛГАА

## **Хаяган хувьсагч**

## Хаяг бол тоон утга тул түүнийг хувьсагч руу хадгалж болох ба хаяг хадгалах хувьсагчийг хаяган хувьсагч гэнэ.

## **Хаяган хувьсагч тодорхойлох:**

## Програм нь хаягийн зааж байгаа утга ямар төрлийнх болохыг мэдэж байх ёстой. Учир нь, (char) тэмдэгтийн хэрэглэж байгаа ойн хаяг бүтцийн хувьд (double) бодит тооныхтой адил байвч тэдгээрийн хэрэглэж байгаа ойн хаяг бүтэцийн хувьд компьютерийн дотоод дүрслэлийн хувьд хоорондоо ялгаатай байдаг. Иймд хаяган хувьсагчийн тодорхойлохдоо түүнд ямар утга хадаглахыг зааж өгөх ёстой.

## int \*int\_ptr;

## Тус хувьсагчийг хаягийн хувьсагч гэдэгийг нь (\*) од, хаяг нь бүхэл тоо утгынх болохыг int тус тус заана. Харин int\_ptr нь хаяган хувьсагчийг нэр юм.[1]

## C++:

## char \*char\_ptr;

## flaot \*float\_ptr;

## double \*double\_ptr

## long \*long\_ptr;

## **Дам хандалтын оператор**:

## Од (\*) оператор нь хаягийн хувьсагчийн эсвэл хаяган тогтмолын зүүн талд хэрэглэглэж байвал түүнийг дам хандалтын оператор гэнэ. Хэрэв ptr нь хаягийн хувьсагч бол дам хандалтын оператор бүхийн \*ptr илэрхийлэл нь уг хувьсагчийн заах ойн утгыг авчирч өгнө.[1]

## int x = 255;

## int \*ptr = &x;

## cout << \*ptr << endl; // 255-г хэвлэнэ

## **Хаягийн арифметик**:

## Жишээ нь int\_ptr гэсэн бүхэл тоон өгөгдийг хаяглах хувьсагч бөгөөд түүний утга нь (int \* ) 0xa000 гэсэн хаяг байна гэж үзье.Тэгвэл (int\_ptr + 1) хаяг нь ptr\_int хувьсагчийн хаяглах өгөглийн байт хэмжээгээр ptr\_int-c алсап орших ойг заана.

## **Динамик ойн одератор**

## С хэлний хуюьд malloc(), free() функцүүп бөгөөд энний функцээр нь ой булдэж хэрэглээд буцааж чөлөөлөхдөө улаач функчийг нь хэрэглэнэ. Эдгээрийг ашиглахд alloc.h, stdlib.h толгой файлуудын аль нэгийг препроцессорын #include командаар зааж өгөх ёстой. Мөн malloc() нь хэрэглэх санах ойн хэмжээ тодорхойлох зэрэг олон нэмэлт алахмууд хэрэгтэй байдаг.

## Дээрх хоёр функчийг С++ хэлэнд шууд хэрэглэж болно. Мөн malloc() функцийг орлох new, харин free() функчийг орлох delete гэсэн хоёр энгийн хэрэглээний оператор бий болсон. [1]

## **new оператор**:

## Дрограм ажиллах явцдаа (OS) үйлдэлийн системээс нэмэлт ой хүсэхдээ new операторыг хэрэглэж авна.

## new Type;

## Type төрөлийн объектод хүрэлцэх ойг бэлдэх хүсэлтийг системд тавих ба систем түүнийг биелүүлж чадвал бэлдсэн ойнхоо эхлэл хаягийг, ой бэлдэх боломжгүй бол хоёртын тэгийг new операторт буцааж өгнө. Энэ оператороос буцаж ирэх хаягийг төрөл нь тохирох хаяган хувьсагчид хадгалж болох учир уг операторыг голдуу хаяган хувьсагчтай холбож хэрэглэнэ.[1]

## Хүснэгт үед:

## new Type[урт] гэж зарална.

## **Delete** **оператор:**

## new нь сул ойн сангаас тодорхой хэмжээний ойн мужийг авч хэрэглэх хүсэлт бол delete нь ойн мужийг сул оан санд буцааж бүртгэхтэй холбоотой хүсэлт юм.[1]

## delete Хаяг

## Тухайн хаягийг чөлөөлнө.

## Хүснэгт үед

## delete [] Хаяг []-тай ашиглна

## **Заалтан хувьсагч**

## Заалтан хувьсагчийг тодорхой нэр бүхий хувьсагчтай холбож үүсгэх бөгөөд хэрэв x\_ref нь x хувьсагчийн заалтан нэр бол x ба x\_ref нь нэг ойнхоёр өөр нэр болох учир эдгээр нэрийн алийг ч хэрэглэж болно.

## int x = 10;

## int &x\_ref = x;

## x\_ref бол х хувьсагчтай холбоостой заалтан хувьсагч үүснэ. Заалтан хувьсагч гарааны утга оноохоос өөр аргаар утга оноож болдоггүй.

* **Inline функц**

C++ хэлний inline функц бол програмын хурдыг нэмэгдүүлэх зориулалтаар бүтээгдсдэн боломж юм. Дотоод мөр функц ба ерийн Си функц хооронд байх үндсэн ялгаа нь тэдгээрийг програмд хэрхэн холбож өгдөгт оршино. Си функцийг өөр функц дотроос дуудах үед дараах үйлдлүүд дараалан хийгддэг. Тухайлбал,

* Дараагийн ээлжинд хийгдэх командын хаягийг хадгалах
* CPU дотоод регистрийн утгыг хадгалах
* Дуудагдсан функц рүү дамжуулж өгч байгаа аргументийг стек рүү хадгалах
* Програмын удирдлага дуудагдсан функц рүү шилжин очиж улмаар аргументээ стекээс авах
* Функц ажиллаж дуусахад түүнээс буцах утгыг үндсэн програм, эх функц нь гаргаж авахаар CPU регистр эсвэл стек рүү түр хадгалах
* Програмын удирдлага эх функц рүү шилжихдээ өмнө хадгалсан буцах хаяг, CPU регистрийн утгыг стекээс буцааж гаргах

Функц хэрэглэх үед дээрх бүх үйлдэлд нэмэлт CPU цаг шаардагдана. Дотоод функц хэрэглэвэл түүний командуудыг функцийг хэрэглэсэн командын мөр бүрд хувилж үүсгэнэ. Энэ бол дотоод функцийн сул тал, учир нь програмын хэмжээ ихсэх талтай. Гэхдээ буцах хаяг хадгалах мэтээр дээр дурдсан бүх үйлдлийг хийх шаардлагагүй болох тул нэмэлт CPU цаг шаардагдахгүй, ингэснээр програмын хурд нэмэгдэх боломжтой болдог.

Хэрэглэгчийн функцийг дотоод функц болгохдоо inline гэсэн тусгай үгийг хэрэглэнэ.

Функц бүр inline функц болж чадахгүй. Тухайлбал, С++ компайлер нь хэт том функц, бас рекурсив функцийг дотоод функц болгож чадахгүй. Мөн С++ компайлерт inline шинж байхгүй бол ямар ч функц дотоод функц болж чадахгүй.[1]

* **Ойн цоорхой**

Ойн цоорхой (memory leaks) бол new, delete операторуудыг дэс дараатайгаар олон удаа хэрэглэх програмд гарч болох ойлголт юм. [1]

## Хаяган хувьсагч зарлаж түүнд new оператороор буцаж ирсэн хаягаа хадгалан. Гэвч delete оператор ашиглж тухай санах ой чөлөөлөхөөс өмнө тухай хувьсагчаа алдах үед санах ойн цоорхой бий болно. Жишээ нь:

## 9-р хавсралтын жишээгээр харуулсан. 2 удаа for давталт ашиглаж хоёр бүхэл тоо нөөцөлсөн гэвч багц мөрийн хаалт хаагдан хадгалж авсан хувьсагч устаад delete хийх боломжгүй болно.

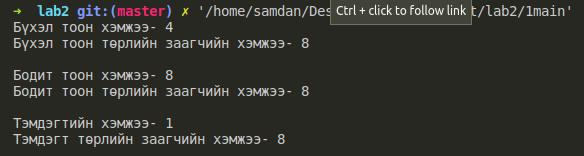
# 4. ХЭРЭГЖҮҮЛЭЛТ

**1-р асуулт:**

Хаяган хувьсагчийн талаар онолын хэсэг дээр судалж бичсэн.

Хаяган хувьсагчийн хэмжээг Ubuntu 18.04 64bit үйлдлийн систем дээр тодорхойлсон нь:

а гэсэн бүхэл тоо, b гэсэн бодит тоо, c гэсэн тэмдэг гэсэн гурван хаяган хувьсагчуудыг зарлаж өгсөн. Тэдгээрийн санах ойд эзлэх хэмжээг нь sizeof() гэсэн оператор ашиглан олж cout ашиглан хэвлэсэн. Үүнээс үзэхэд бүгд адилхан 8byte байна.



**2-р асуулт:**

Хаяган хувьсагчийн хэмжээг Ubuntu 18.04 64bit үйлдлийн систем дээр тодорхойлж түүний учрыг тайлбарласан нь:

1-р асуултын адил бүхэл тоо, бодит тоо, тэмдэгт гэсэн гурван хаяган хувьсагчуудыг зарлаж тэдгээрийн санах ойд эзлэх хэмжээг нь sizeof оператор ашиглаж олсон. Үүний хариуд бүгд хавсралтын 16-р мөрт 8 гэж хэвлэгдэнэ.

Шалтгаан:

Хаяган хувьсагч нь бидний хэрэглэж буй үйлдлийн систем 32, 64 bit байхаас хамаарна.

32 bit тохиолдолд бидний хэрэглэх хаяган хувьсагч 2^32 ширхэг санах ойн үүрүүдийг тоолж чадах ёстой учраас 4 byte буюу 32 bit байна. Харин 64 bit үед 2^64 ширхэг санах ойн үүрүүдтэй ба түүнийг хадгалахад 8 byte буюу 64 bit санах ой хэрэгцээтэй. (1 byte = 8bit)

**3-р асуулт:**

# 3-р хавсралтын 7-р мөр: а гэсэн бүхэл тоо зарлаж, түүнд 125 гэсэн анхны утга оноож өгнө

# 3-р хавсралтын 10-р мөр: p гэсэн бүхэл тоон хаяган хувьсагч зарлаж, а-н хаягийг p-д оноож өгнө

# 3-р хавсралтын 13-р мөр: p буюу а-н хаягийг хэлнэ

# 3-р хавсралтын 16-р мөр: р-н утга буюу а-н хаягийг дам хандалтын оператор ашиглан тухайн хаяг дээр байгаа утгыг хэвлэнэ - 125

# 3-р хавсралтын 19-р мөр: p хувьсагч дээр 1-г нэмэх ба энэ заагч нь бүхэл тоон төрөл 4 byte-р нэмэгдэж &a + 4 болно.

# 3-р хавсралтын 22-р мөр: р-н шинэ хаягийг хэвлэнэ

# 3-р хавсралтын 25-р мөр: Дам хандалтын оператор ашиглан шинэ хаягт байгаа утгыг хэвлэнэ

**4-р асуулт:**

# 4-р хавсралтын 7-р мөр: Бүхэл тоон 5 элементтэй хүснэгт зарлана

# 4-р хавсралтын 10-р мөр: Бүхэл тоон хаяган хувьсагч зарлана

# 4-р хавсралтын 13-р мөр: р-д бүхэл тоон хүснэгтийн анхны элементийн хаягийг өгнө

# 4-р хавсралтын 16-р мөр: р-н анхны утгыг дам хандалтын оператороор хандаж 10 болж байна.

# 4-р хавсралтын 19-р мөр: р-н утга буюу number-н эхний элементийн хаягийг 1-ээр нэмж дараагийн элементийн хаягийг авна

# 4-р хавсралтын 22-р мөр: number хүснэгтийн 2-р элементэд 20 гэсэн утга онооно

# 4-р хавсралтын 25-р мөр: р-д number-н 3-р элементийн хаягийг өгнө

# 4-р хавсралтын 28-р мөр: number-н 3-р элементийг 30 болгоно

# 4-р хавсралтын 31-р мөр: р-д number хүснэгтийн 4-р элементийн утгыг оноож өгнө

# 4-р хавсралтын 34-р мөр: number-н 4 дахь элементийг 40 болгоно

# 4-р хавсралтын 37-р мөр: р-д number хүснэгтийн анхны элементийн хаягийг өгнө

# 4-р хавсралтын 41-р мөр: р-д эхний хаяг дээр sizeof(int) \* 4-р нэмж number хүснэгтийн 5 дахь элементийн утгыг 50 болгоно

# 4-р хавсралтын 44-46-р мөр: number-н эхний 5 элементийг хэвлэж харуулна

# Үр дүн:

# 

**5-р асуулт:**

Бодох арга: Хоёр тооны утгыг хооронд нь солиход ямар нэгэн temp буюу түр аль нэг утгыг нь хадгалах хувьсагч зарлаж өгнө. Хоёр тооны эхнийх нь утгыг temp-д хадгалж авна. Түүний дараа эхний хувьсагчид хоёр дахь хувьсагчийн утгыг нь оноож өгнө. Хамгийн сүүлд temp-д хадгалж авсан эхний тооны утгыг хоёр дахь тоонд оноож өгнө.

# temp = a;

# a = b;

# b = temp;

# Энэ асуулт дээр хаяган хувьсагч ашиглан шинэ функц тодорхойлж хоёр тооны утгыг нь солино. 5-р хавсралтын 14-р мөрт функцийн параметр нь хаяган хувьсагч ба мөр 17, 19, 21-т дам хандалтын оператор ашиглан дээр тайлбарласан бодох аргын дагуу хоёр бүхэл тоон утгыг сольсон.

# 

# Оролт: a = 10, b = 96 гэсэн хоёр бүхэл тоо

# Үйлдэл: Тухайн хоёр тоог дээр дурдсаны дагуу утгыг нь солино. a-д 10 байсан бол 96 гэсэн утгатай болно. Харин b нь 96 гэсэн утгатай байсан бол 10 гэсэн утгатай болсон.

# Гаралт: Void

**6-р асуулт:**

# 5-р асуулттай адил бодох аргатай.

# Гэвч энэ асуултанд хаяган хувьсагч биш заалтан хувьсагч ашиглан хоёр тооны утгыг сольсон.

# Оролт гаралт, үйлдэл нь 5-р асуулттай адилхан.

# 5. ДҮГНЭЛТ

Хаяган хувьсагч нь хувьсагчийн нэрнээс гадна хаягаар нь хандах боломжийг олгодог. Харин заалтан хувьсагч нь нэг зарласан хувьсагчид өөр нэр оноож ашиглах боломжтой байдаг. Динамик ойн new оператор нь хялбараар санах ой нөөцөлдөг бол delete оператор нь нөөцөлсөн санах ойг чөлөөлдөг. Чөлөөлөөгүй үед санах ойн цоорхой үүснэ. Ийм учраас delete оператор ашиглан чөлөөлөх нь зүйтэй юм. Inline функц нь ажиллаж буй програмын хурдыг хурдасгах давуу талтай ч програмын хэмжээг ихэсгэх сул талтай.

# 6. АШИГЛАСАН МАТЕРИАЛ

1. Объект хандлагат технологийн С++ програмчлал, Ж.Пүрэв, 2008, Улаанбаатар.

# 7. ХАВСРАЛТ

**1-р асуулт**

#include <iostream>;

using *namespace* std;

*int* main()

{

    // Бүхэл тоон заагч

*int* \*a;

    // Бодит тоон заагч

*float* \*b;

    // Тэмдэгт төрлийн заагч

*char* \*c;

    cout << "Бүхэл тоон хэмжээ- " << sizeof(*int*) << endl;

    cout << "Бүхэл тоон төрлийн заагчийн хэмжээ- " << sizeof(a) << endl

         << endl;

    cout << "Бодит тоон хэмжээ- " << sizeof(*double*) << endl;

    cout << "Бодит тоон төрлийн заагчийн хэмжээ- " << sizeof(b) << endl

         << endl;

    cout << "Тэмдэгтийн хэмжээ- " << sizeof(*char*) << endl;

    cout << "Тэмдэгт төрлийн заагчийн хэмжээ- " << sizeof(c) << endl

         << endl;

}

**2-р асуулт**

#include <iostream>;

using *namespace* std;

*int* main()

{

    // Тэмдэгт тоон заагч зарласан

*char* \*p1;

    // Бүхэл тоон заагч зарласан

*int* \*p2;

    // давхаp нарийвчлалтай бодит тоон заагч зарласан

*double* \*p3;

    // p1, p2, p3 - ийн хэмжээг хэвлэж байна.

    cout << sizeof(p1) << " " << sizeof(p2) << " " << sizeof(p3);

    // OS-ийн 32bit / 64 bit ээс хамааран 4 / 8 байна.36+

}

**3-р асуулт**

#include <iostream>;

using *namespace* std;

*int* main()

{

    // а гэсэн бүхэл тоо зарлаж, түүнд 125 гэсэн анхны утга оноож өгнө

*int* a = 125;

    // p гэсэн бүхэл тоон хаяган хувьсагч зарлаж, а-н хаягийг p-д оноож өгнө

*int* \*p = &a;

    // p буюу а-н хаягийг хэлнэ

    cout << p << endl;

    // р-н утга буюу а-н хаягийг дам хандалтын оператор ашиглан тухайн хаяг дээр байгаа утгыг хэвлэнэ - 125

    cout << \*p << endl;

    // p хувьсагч дээр 1-г нэмэх ба энэ заагч нь бүхэл тоон төрөл 4 byte-р нэмэгдэж &a + 4 болно.

    p++;

    // р-т оноогдсон шинэ хаягийг хэвлэнэ

    cout << p << endl;

    // Дам хандалтын оператор ашиглан шинэ хаягт байгаа утгыг хэвлэнэ

    cout << \*p << endl;

}

**4-р асуулт**

#include <iostream>;

using *namespace* std;

*int* main()

{

    // Бүхэл тоон 5 элементтэй хүснэгт зарлана

*int* numbers[5];

    // Бүхэл тоон хаяган хувьсагч зарлана

*int* \*p;

    // р-д бүхэл тоон хүснэгтийн анхны элементийн хаягийг өгнө

    p = numbers;

    // р-н анхны утгыг дам хандалтын оператороор хандаж 10 болж байна.

    \*p = 10;

    // р-н утга буюу number-н эхний элементийн хаягийг 1-ээр нэмж дараагийн элементийн хаягийг авна

    p++;

    // number хүснэгтийн 2-р элементэд 20 гэсэн утга онооно

    \*p = 20;

    // р-д number-н 3-р элементийн хаягийг өгнө

    p = &numbers[2];

    // number-н 3-р элементийг 30 болгоно

    \*p = 30;

    // р-д number хүснэгтийн 4-р элементийн утгыг оноож өгнө

    p = numbers + 3;

    // number-н 4 дахь элементийг 40 болгоно

    \*p = 40;

    // р-д number хүснэгтийн анхны элементийн хаягийг өгнө

    p = numbers;

    // р-д эхний хаяг дээр sizeof(int) \* 4-р нэмж number хүснэгтийн

    // 5 дахь элементийн утгыг 50 болгоно

    \*(p + 4) = 50;

    // number-н эхний 5 элементийг хэвлэж харуулна

    for (*int* n = 0; n < 5; n++)

        cout << numbers[n] << ", ";

}

**5-р асуулт**

#include <iostream>;

using *namespace* std;

/\*

    2 тооны утгуудыг хаяган хувьсагч ашиглан солино

    Params:

    int \*a - утгыг нь солих тооны хаяг

    int \*b - утгыг нь солих тоона хаяг

    Return:

    void

\*/

*void* swap\_nums\_with\_ponter(*int* \*a, *int* \*b)

{

    // temp-т а-н утгыг хадгална

*int* temp = \*a;

    // а-т b-н утгыг оноож өгнө

    \*a = \*b;

    // b-m хадгалж авсан а-н утгыг оноож өгнө

    \*b = temp;

}

*int* main()

{

    // утгыг нь солих 2 бүхэл тоо зарлах

*int* a = 10;

*int* b = 96;

    // функцээ дуудан ажиллуулах

    swap\_nums\_with\_ponter(&a, &b);

    // хоёр тоогоо хэвлэнэ

    cout << "a: " << a << endl;

    cout << "b: " << b << endl;

}

**6-р асуулт**

#include <iostream>

using *namespace* std;

/\*

    2 тооны утгуудыг заалтан хувьсагч ашиглан солино

    Params:

    int a - утгыг нь солих тоо

    int b - утгыг нь солих тоо

    Return:

    void

\*/

*void* swap\_nums\_with\_refrence(*int* &a, *int* &b)

{

    // temp-т а-н утгыг хадгална

*int* temp = a;

    // а-т b-н утгыг оноож өгнө

    a = b;

    // b-m хадгалж авсан а-н утгыг оноож өгнө

    b = temp;

}

*int* main()

{

    // утгыг нь солих 2 бүхэл тоо зарлах

*int* a = 10;

*int* b = 96;

    // функцээ дуудан ажиллуулах

    swap\_nums\_with\_refrence(a, b);

    // хоёр тоогоо хэвлэнэ

    cout << "a: " << a << endl;

    cout << "b: " << b << endl;

}

**9-р асуулт**

#include <iostream>

using *namespace* std;

*int* main()

{

    for (*int* i = 0; i < 2; i++)

    {

        // а гэсэн бүхэл тоон хаяган хувьсагч зарлаж түүнд new int-с нөөцөлсөн хаягийг оноож өгнө

*int* \*a = new *int*;

        // p\_a-т а-н хаягийг өгнө

        p\_a = a;

        // а-н хаяган дээр байгаа утгыг i болгоно

        \*a = i;

        // а-н хаяган дээр байгаа утгыг хэвлэнэ

        cout << "\*a = " << \*a << endl;

    }

    return 0;

}