C++ хэлний хаяган хувьсагч, заалтан хувьсагч
Лаборатори 2

Э.Түвшин-Эрдэнэ(23B1NUM0869)
МТЭС, МКУТ-ын “Программ хангамж” хөтөлбөрийн оюутан, tuvshin.erdene25@gmail.com
2025 он

1.Оршил

Энэ лабораторийн ажлаар C++ хэлний хаяган ба заалтан хувьсагч, “new” оператор, санах ойн цоорхой (memory leak) зэрэг ойлголтуудыг судлан, холбогдох C++ кодуудыг оруулав.

2.Зорилго

Санах ойн хаяг, хаяган ба заалтан хувьсагч, “new” оператор зэрэг ойлголтуудыг жишээтэйгээр тайлбарлан, программчлалын C++ хэл дээр энгийн бодлогууд бодож харуулах зорилготой. Түүнчлэн өргөнөөр тохиолддог санах ойн цоорхой(memory leak)-г тайлбарлана.

3. Онолын судалгаа

3.1)Санах ойн хаяг

Санах ойн хаяг нь RAM дээр хадгалагдсан өгөгдлийн байршил юм.[4]

Орчин үеийн компьютерууд ихэвчлэн 64 битийн санах ойн хаягтай(хэрэв 32 битийн үйлдлийн системтэй бол санах ойн хаяг нь 32 бит байна.)[5]

C хэлэнд санах ойн хаягт хандахдаа “&” операторыг хэрэглэдэг:

```
int myAge = 43;
printf("%p", &myAge); // Outputs 0x7ffe5367e044
```

3.2)Динамик ой

Программ гүйцэтгэх явцад санах ойд үүсэж болох, утгын хэмжээг тодорхойлж болох объектыг динамик ой гэнэ. Заагч ашиглан динамик ойд хандаж болно.[3]

Статик ба глобал хувьсагчид нь программыг дахин compile хийх үед өөрчлөгддөг. Харин динамикаар байгуулсан хувьсагчдад ашиглагддаг нөөц ой болох heap ойг программ гүйцэтгэх явцад байгуулах, устгах боломжтой.[3]

Динамик санах ой ашиглах алхам:

1. Өөрт хэрэгцээтэй хэмжээгээр санах ой тогтоох
2. Санах ойг хуваарилах
3. Заагчаар санах ойг эзэмших
4. Санах ойг чөлөөлөх

3.3)Хаяган хувьсагч

Хаяг бол тоон утга тул түүнийг хувьсагч руу хадгалж болох ба хаяг хадгалах хувьсагчийг хаяган хувьсагч гэнэ. Программ нь хаягийн зааж байгаа утга ямар төрлийнх болохыг мэдэж байх ёстой. Учир нь тэмдэгтийн хэрэглэж байгаа ойн хаяг бүтцийн хувьд бодит тооныхтой адил боловч тэдгээрт шаардагдах ой нь тоо хэмжээ болон компьютерийн дотоод дүрслэлийн хувьд хоорондоо ялгаатай байдаг. Иймд хаяган хувьсагч тодорхойлохдоо түүнд ямар утга хадгалахыг зааж өгөх ёстой.[2]

Хаяган хувьсагч тодорхойлох:

<хадгалах өгөгдлийн төрөл> * <хаяган хувьсагчийн нэр>

Тухайн хаяган хувьсагчийн зааж буй өгөгдлийн төрлөөс үл хамааран 4 байтын хэмжээтэй байна.

3.4)Заалтан хувьсагч

Заалтан хувьсагч нь C хэлэнд байдаггүй, C++ хэлний нэг онцлог бөгөөд тодорхой нэр бүхий хувьсагчтай холбож үүсгэнэ. Заалтан нэр болон зааж буй хувьсагчид нь хоёулаа ижил нэг ойн хоёр өөр нэрүүд болно.[2]

Заалтан хувьсагч тодорхойлох:

```
Int x;
Int &x_ref = x;
```

Заалтан хувьсагчийг функцийн параметрээр хэрэглэхэд нэн тохиромжтой байдаг. Ингэснээр ой хэмнэхээс гадна эх өгөгдөл рүү нь шууд хандан бодолт хийх боломжтой болно.

Заалтан хувьсагчийг зарлахдаа гарааны утгыг заавал оноох хэрэгтэй.

Заалтан хувьсагч параметрээр дамжуулах:

1. Void func_name(int &x); //ил арга
2. Void func_name(int &); //далд арга

Хэрэв функцээр параметрийн утгыг өөрчлөхгүй бол параметрээр авахдаа “const” түлхүүр үгийг ашиглах хэрэгтэй.

```
Void func_name (const int &x)
```

Мөн функц заалтан утга буцаадаг байж болно:

```
Int & func_name(int &z)
```

3.5)New оператор

“New” нь “malloc” командыг бодвол энгийн, нэмэлт үйлдэл шаарддаггүй оператор юм. Сул ойг нөөцлөөд, авсан ойн эхний хаягийг буцаана.

Тухайн объектод хүрэлцэх ойг бэлдэх хүсэлтийг системд тавих ба амжиллтай бэлдвэл эхний хаягийг нь, эсрэг тохиолдолд хоёртын тэгийг буцаана.

Энэ операторoос буцаж ирэх хаягийг төрөл нь тохирох хаяган хувьсагчид хадгалж болох учир уг операторыг голдуу хаяган хувьсагчтай холбож хэрэглэдэг.[2]

```
Int *intPtr;
intPtr = new int ;
```

Хүснэгттэй ашиглах:

```
Int *arrayPtr;
arrayPtr = new int[10];
```

3.6)Delete оператор

Delete операторыг ашиглан new операторaар нөөцөлсөн санах ойг чөлөөлнө.[2]

```
Delete pointerVariable;
Delete [arraySize] arrayPointerVariable;
```

3.7)Санах ойн цоорхой

Санах ойн цоорхой(memory leak) нь санах ойг динамикаар нөөцлөөд ашиглаж дууссаны дараа чөлөөлөөгүйгээс үүдэн тэр нөөцөлсөн санах ойд хандах боломжгүй болох асуудал

юм. Хамгийн муу нөхцөл байдалд хэтэрхий их санах ойг нөөцлөн, чөлөөтэй санах ойн хэмжээ багасна.[1]

Үүнээс сэргийлэхийн тулд динамикаар нөөцөлсөн санах ойг чөлөөлж байх хэрэгтэй.

Санах ойн хог(garbage) - Санах ойд зай эзэлж байгаа ч гэсэн хандах заагч байхгүй хувьсагч.[1]

4.Хэрэгжүүлэлт

4.1)Хаяган хувьсагчийн хэмжээ

```
#include<iostream>
using namespace std;
int main(){
    char *p1;
    int *p2;
    double *p3;
    cout << sizeof(p1)<<endl<<sizeof(p2)<<endl<<sizeof(p3)<<endl;
    return 0;
}
```

```
tuvshinerdene@PC:~/sem4/OOP/OOP/lab2$ ./exercise5
8
8
8
```

Энэ жишээнд өгөгдлийн төрлөөс үл хамааран хаяган хувьсагчдын хэмжээ тогтмол 8 байт байна. Учир нь энэ үйлдлийн систем 64 битийнх.

4.2)Хаяган хувьсагчийн утгыг өөрчлөх

```
#include<iostream>

using namespace std;
int main(){
    int a=125; //a хувьсагч зарлан 125 утгыг оноов
    int *p = &a;//p хаяган хувьсагч зарлан a хувьсагчийн хаягийг оноов.
    cout<<p<<endl;//p-ийн зааж буй хувьсагчийн хаяг
    cout<<*p<<endl;//p-ийн зааж буй хувьсагчийн утга
    p++;//Хаягийн утгыг нэмэгдүүлж байгаа тул санах ойн дараагийн хаягийг заана.(энэ хаягт өмнө нь хадгалагдаж байсан garbage утгыг харуулна.)
```

```
    cout<<p<<endl;
    cout<<*p<<endl;
    return 0;
}
```

```
tuvshinerdene@PC:~/sem4/OOP/OOP/lab2$ ./exercise5_1
0x7ffc9c7b497c
125
0x7ffc9c7b4980
-1669641856
```

4.3)Хаяган хувьсагчаар хүснэгтэд утга оноох

```
#include<iostream>
using namespace std;
int main(){

    int numbers[5];//5 элементтэй бүхэл тоон хүснэгт зарлав
    int * p;//бүхэл тоон хаяган хувьсагч
    p = numbers; //хаяган хувьсагчид хүснэгтийн хаягийг оноов
   *p = 10;//хаяган хувьсагчийн утга буюу хүснэгтийн эхийн элементэд 10 гэсэн утга оноов.
    p++; *p = 20;//p-ийн утгыг нэгээр нэмэгдүүлснээр дараагийн элемент рүү заана.
    p = &numbers[2]; *p = 30;//хүснэгтийн 3-р элементийн хаягийг оноогоод 30 гэсэн утгыг оноосон.
    p = numbers + 3; *p = 40;//эхний элементийн хаяг дээр 4-ийг нэмснээр хүснэгтийн 3-р элемент рүү заана.
    p = numbers;*(p+4) = 50; //p-ийн утгыг хүснэгтийн эхний элемент рүү заан 5-ыг нэмснээр хүснэгтийн сүүлийн элементэд утга онооно.

    for(int n = 0; n < 5; n++){
        cout<<numbers[n]<<", ";//хүснэгтийн элементүүдийг хэвлэн гаргав.
    }

    return 0;

}
```

```
tuvshinerdene@PC:~/sem4/OOP/OOP/lab2$ ./exercise6
10, 20, 30, 40, 50, tuvshinerdene@PC:~/sem4/OOP/OOP/lab2$
```

Дээрх жишээнд хүснэгтийн элементэд хандах өөр аргуудыг харуулав.

4.4)Хаяган хувьсагч параметрээр авах

Хаяган хувьсагч параметрээр аван утгуудыг нь солих хэрэглэгчийн функц бичив:

```cpp
// Хаяган хувьсагч ашиглан хоёр хувьсагчийн утгыг солих хэрэглэгчийн
функц.
#include<iostream>
using namespace std;

void swap(int *a, int *b){
    int temp = *a;
    *a = *b;
    *b = temp;
}

int main(){
    int a = 10, b = 20;
    cout<<"Before swapping: "<<"a = "<<a<<", b = "<<b<<endl;

    int *p1 = &a, *p2 = &b;
    swap(p1, p2);

    cout<<"After swapping: "<<"a = "<<a<<", b = "<<b<<endl;

    return 0;
}
```

```
tuvshinerdene@PC:~/sem4/OOP/OOP/lab2$ g++ exercise7.cpp -o exercise7
tuvshinerdene@PC:~/sem4/OOP/OOP/lab2$ ./exercise7
Before swapping: a = 10, b = 20
After swapping: a = 20, b = 10
```

4.5)Заалтан хувьсагч параметрээр авах

Заалтан хувьсагч параметрээр аван утгуудыг нь солих хэрэглэгчийн функц бичив:

```cpp
//8. Заалтан хувьсагч хоёр хувьсагчийн утгыг солих хэрэглэгчийн функц
#include<iostream>
using namespace std;

void swap(int &a, int &b){
    int temp = a;
    a = b;
    b = temp;
}

int main(){
    int a = 10, b = 20;
    cout<<"Before swapping: "<<"a = "<<a<<", b = "<<b<<endl;

    swap(a, b);

    cout<<"After swapping: "<<"a = "<<a<<", b = "<<b<<endl;

    return 0;
}
```

```
tuvshinerdene@PC:~/sem4/OOP/OOP/lab2$ g++ exercise8.cpp -o exercise8
tuvshinerdene@PC:~/sem4/OOP/OOP/lab2$ ./exercise8
Before swapping: a = 10, b = 20
After swapping: a = 20, b = 10
```

5.Дүгнэлт

C++ хэл дээр заалтан хувьсагч ашиглах нь хаяган хувьсагчаас санах ойн хэрэглээ багатайгаараа давуу талтай байна. Мөн хаяган болон заалтан хувьсагчаар функц руу утга дамжуулснаар тухайн хувьсагчийн утгыг өөр функц дотор өөрчлөх боломж үүсч байгаа тул үүнийг ашиглан функцээс олон үр дүн гаргаж авах боломжтой санагдав. Энэ лабораториос харахад C++ хэл нь санах ойн уян хатан зохион байгуулалттай ч гэсэн анхааралтай ажиллахгүй бол санах ойн цоорхой үүсэх эрсдэлтэй. Цаашид хаяган болон заалтан хувьсагчийн илүү ахисан түвшинд(хаяган хувьсагчийн хаягийг ашиглах гэх мэт) судлах хэрэгтэй гэж бодогдлоо.

6.Ашигласан материал

1. https://www.geeksforgeeks.org/what-is-memory-leak-how-can-we-avoid/.
2. “Объект хандлагат технологийн С++ програмчлал” Ж.Пүрэв 2008, Улаанбаатар
3. “Програмчлалын хэл Си” Н.Соронзонболд 2011, Улаанбаатар
4. https://www.computerhope.com/jargon/m/memoaddr.htmAddress?
5. https://computer.howstuffworks.com/c23.htmrstanding Memory Addresses - The Basics of C Programming | HowStuffWorks
6. https://www.w3schools.com/c/c_memory_address.phpdress

7.Хавсралт