1 Санах ойн хаяг

```c
#include <stdio.h>
int main() {
  int n = 5;
  printf("%u\n", &n); // n-iin haygiig hevlene
  return 0;
}
```

int n; зарлагдах үед Си хэл stack дээр 4byte санах ой нөөцөлнө. n-ийн хаягийг хэвлэхдээ %u гэж зааж өгч байгаа нь unsigned int төрлийг хэвлэж байгааг Си гаралтын системд хэлж өгч байна.

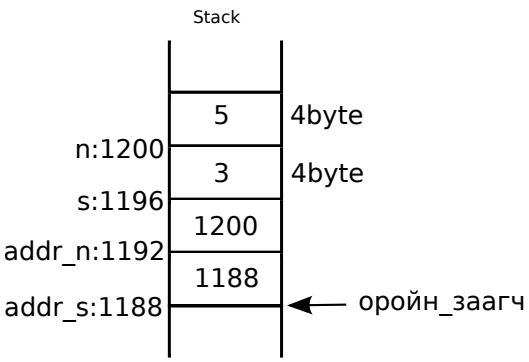
Санах ой 32bit-ийн үйлдлийн систем дээр $0..2^{32}-1$ хүртэл дугаарлагдсан байдаг. Тиймээс санах ойн хаяг 4byte буюу 32bit санах ойд багтана.

int төрөл 2^{31}-ийг хасах тоонд, ахиад 2^{31}-ийг нэмэх тоонд, нийтдээ $2^{31}+2^{31}=2*2^{31}=2^{32}$ зэрэгт буюу 4byte-ийг эзэлдэг. Хэрвээ бид санах ойн хаягийг int төрөлд хадгалах гэж байгаа бол unsigned int төрөлд хадгалбал орон хэтрэл үүсэхгүй. Учир нь int төр $2^{31}-1$ хүртэлх нэмэх тоонуудыг хадгалж чаддаг байхад санах ой $2^{32}-1$ хүртэл дугаарлагдсан байх юм. Харин unsigned int хасах тоо хадгалдаггүй учраас бүх 4byte-ийг эерэг тоо хадгалахад зориулдаг тул $2^{32}-1$ хүртэл тоог хадгалж чадна.

```c
#include <stdio.h>
int main() {
  int n;
  unsigned int addr = (unsigned int) &n;
  scanf("%d", addr);  // scanf("%d", &n); ene uildeltei ijil
                      // uchir ni scanf-ruu n-iin haygiig damjuulna
  printf("%d\n", n);
}
```

```c
#include <stdio.h>

int main() {
    int n = 5;
    int s = 3;
    unsigned int addr_n = (unsigned int) &n;
    unsigned int addr_s = (unsigned int) &s;
    printf("%u\n", addr_n - addr_s);
    return 0;
}
```



Зураг 1: stack

Stack хамгийн сүүлд орсон элементийг эхэлж гардаг өгөгдлийн бүтэц. Stack-ийг нарийн хоолойтой зүйрлэвэл хамгийн доор байгаа зүйлийг авахын тулд дээр нь байгаа бүгдийг гаргана. **Stack-ийн оройн заагч**, хамгийн сүүлд орсон элементийн хаягийг заадаг.

Зураг 1-д хувьсагчийн нэрийн ард тодорхойлох цэг тавиад тухайн хувьсагчийн санах ойн хаягийг бичигдэж, тухайн хаяг доторх утга дөрвөлжин дотор бичигдсэн. Си хэлэнд stack дээрээс доошоо өсдөг. Дээрх

хоёр хувьсагчийн хувьд n түрүүлж зарлагдсан учраас түрүүлж stack дээр үүсч, stack-ийн хэмжээ 4byte-аар нэмэгдэх ба stack-ийн оройн заагчийн хаяг 4byte-аар хорогдоно (доошоо өсдөг). n, s нь int, stack дээр дараалаад үүссэн учраас хаягуудынх нь хоорондын зөрөө 4byte.

1.1 Хаяган хувьсагч

Санах ойн хаяг 4byte-ийг хадгалах зориулагдсан хувьсагчдыг хаяган хувьсагчид гэнэ. Хаяган хувьсагч утгаараа санах ойн хаяг авдаг.

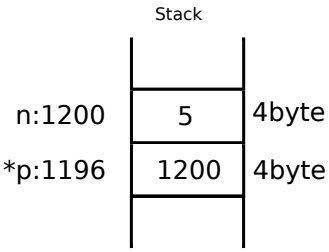
Дүрэм:

төрөл *нэр;

Жирийн хувьсагчаас ялгахдаа нэрийнх нь өмнө од (*) тавина.

```
#include <stdio.h>
int main() {
    int n = 5;
    int *p;
    p = &n;
    printf("%u\n", &n);
    printf("%u\n", p);
    return 0;
}
```

n-ийн хаяг, хаяган хувьсагч p-ийн утга хоёр ижил байна.



Зураг 2: хаяган хувьсагч

Хаяган хувьсагч санах ойн хаяг болох 4byte тоог хадгалахаас гадна тухайн санах ойруу хандах боломжийг олгодог. Хаяган хувьсагчийн нэрийн өмнө од (*) тавихад, хадгалж байгаа хаяган доторх утгаруу хандана. Үүнийг **дам хандалт** гэж нэрлэдэг.

```
#include <stdio.h>

int main() {
    int n = 5;
    int *p;
    p = &n;
    printf("%d\n", *p);
```

```
    *p = 3;
    printf("%d\n", n);
    return 0;
}
```

Бүтэц төрлийн хаягийг хадгалах хаяган хувьсагчийн хувьд доорх байдлаар гишүүдэд нь хандаж болдог.

```
struct Student {
    char name[20];
    float mark;
};
int main()
{
    struct Student bat;
    struct Student *p = &bat;
    p->mark = 100;
    strcpy(p->name, "Bat");
    //...
}
```

2 Дасгалууд

2.1 Ангид

1. x хувьсагчид хаягаар нь дамжуулан утга оноо.

```
int main()
{
     int x;
     // Utga onoox uildel ....

     printf("%d\n", x);
     return 0;
}
```

2. *even()* функцийг тодорхойл. Уг функц нь n тэгш тоо байвал дамжуулсан int төрлийн хаяг доторх утгад 1-г, үгүй бол 0-г онооно.

```
void even(int n, int *p)
{
     //....
}
int main()
{
     int x, n;
     scanf("%d", &n);
     even(n, &x);
     if (x == 1)
           printf("tegsh\n");
     else
           printf("sondgoi\n");
```

```
    return 0;
}
```

3. Параметрээр дамжуулсан хүснэгтээс тэгш тоонуудын тоог олж, параметрт дамжуулсан *addr* хаягт, сондгой тоонуудын тоог олж *addr1* хаягт тус тус хадгал.

```
void find(int a[], int n, int *addr, int *addr1);

int main()
{
    int a[100];
    read(a, 5);   // lab 9.3-iin read, print funkts
    print(a, 5);
    int x, y;
    find(a, 5, &x, &y);
    printf("tegsh toonuudiin too: %d\n", x);
    printf("sondgoi toonuudiin too: %d\n", y);
    return 0;
}
```

4. Хоёр тооны утгыг сольдог доорх функцийг бич.

```
void swap(int *a, int *b);
int main()
{
    int x = 1, y = 2;
    swap(&x, &y);
    printf("%d %d\n", x, y);
    return 0;
}
```

5. Triangle бүтцэн төрөлд гараас утга авах дараах функцийг бич.

```
struct Triangle {
    int a, b, c;
};
void read(struct Triangle *p)
{
    //...
}
int main()
{
    struct Triangle g;
    read(&g);
    //...
}
```

6. Оюутан бүтцэд гараас утга авах *read_ student()* функц, нэрийг нь өөрчлөх *change_ name()* функцүүдийг тус тус тодорхойл.

```
struct Student {
    char fname[20], lname[20], id[10];
```

```
    float golch;
};
void read(struct Student *s)
{
    //...
}
// oyutnii medeelel hewlex funkts
void print(struct Student s)
{
    //...
}

void set_fname(struct Student *p, char ner[]);
void set_lname(struct Student *p, char ovog[]);
void set_id(struct Student *p, char id[]);
void set_golch(struct Student *p, float golch);
int main()
{
     struct Student bat, t;
     read(&bat);
     print(bat);
     set_fname(&t, "dorj");
     set_lname(&t, "bold");
     set_id(&t, "1324");
     set_golch(&t, 3);
     print(t);
     return 0;
}
```

2.2 Гэрт

1. Хүснэгтийн хамгийн их ба бага элементүүдийг параметрт ирсэн хаягт хадгалах доорх функцийг хэрэгжүүл.

```
    void find(int a[], int n, int *min, int *max);
```

2. Хүснэгтээс *value* элементийг хайж, олдсон хаягийг буцаах доорх функцийг хэрэгжүүл. Олдохгүй бол NULL хаяг буюу 0-г буцаана.

```
int *search(int a[], int value, int size);
int main()
{
     int a[100];
     read(a, 5);
     print(a, 5);
     int *p;
     p = search(a, 3, 5);
     if (p == NULL)
         printf("Oldsongui\n");
     else
         printf("%d too %d bairlald oldloo\n", *p, p - a);
     return 0;
```

```
}
```

3. *Rational* бүтэц нь энгийн бутархайн хүртвэр хуваарийг хадгалах бол тэдгээрийн хооронд үйлдэл хийх дараах функцүүдийг хэрэгжүүл.

```
struct Rational {
  int d, n;  // d/n butarxai
};
Rational add(const Rational *a, const Rational *b);
Rational sub(const Rational *a, const Rational *b);
Rational mult(const Rational *a, const Rational *b);
Rational div(const Rational *a, const Rational *b);
void simplify(Rational *a); // xuraax
void read(Rational *a); // unshih
void print(Rational *a); // xevlex
```