Санах ой

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.

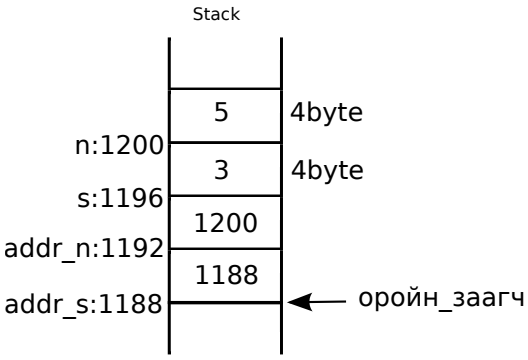
Санах ой 32bit-ийн үйлдлийн систем дээр $0..2^{32}-1$ хүртэл дугаарлагдсан байдаг. Тиймээс санах ойн хаяг 4byte буюу 32bit санах ойд багтана.

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() {
```

```c
        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

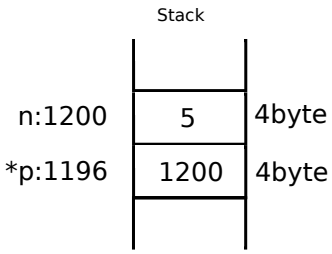
Санах ойн хаяг 4byte-ийг хадгалах зориулагдсан хувьсагчдыг хаяган хувьсагчид гэнэ. Хаяган хувьсагч утгаараа санах ойн хаяг авдаг.

Дүрэм:
төрөл *нэр;

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

```c
#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.

```c
#include <stdio.h>

int main() {
        int n = 5;
        int *p;
        p = &n;
        printf("%d\n", *p);
        *p = 3;
        printf("%d\n", n);
        return 0;
}
```

1.2 Хүснэгт, хаягийн арифметик

Санах ойд дараалаан байрласан элементүүдийн цуваag хүснэгт гэж нэрлэдэг.

Дүрэм:

төрөл нэр[хэмжээ];

Санах ойд дараалаан байрлана гэдэг нь хүснэгтийн элементүүд, хэрэв хүснэгт int төрөл бол дөрөв, дөрвөн byte-ийн зайтай, хэрэв char төрөл бол нэг, нэг byte-ийн зайтай санах ойд байрлана гэсэн үг.

```c
#include <stdio.h>
int main() {
        int ai[100];    // int husnegt zarlaj bna
        char ac[100];   // char husnegt zarlaj bna
        printf("%u", &ai[0]); // 0-r elementiin hayg
        printf("%u", &ai[1]); // 1-r elementiin hayg
        printf("%u", &ai[2]); // 2-r elementiin hayg
```

```
        printf("%u", &ac[0]); // 0-r elementiin hayg
        printf("%u", &ac[1]); // 1-r elementiin hayg
        printf("%u", &ac[2]); // 2-r elementiin hayg
        return 0;
}
```

Хүснэгтийн санах ойд эзлэх нийт хэмжээ нь **хэмжээ * төрлийн_хэмжээ** байна.

```
#include <stdio.h>
int main() {
        int a[100];
        char s[100];
        printf("%u\n", sizeof(a));  // 400
        printf("%u\n", sizeof(s));  // 100
        return 0;
}
```

sizeof функц нь компайлдах үед дамжуулсан аргумент, хэдий хэмжээний хэсэг ой байгааг буцаадаг.

Хүснэгт файлийн хүрээнд зарлагдвал хэмжээ нь заавал тогтмол байна.

```
#include <stdio.h>
int size = 10;
int array[size]; // Aldaa: filiin hureend huvisah hemjeetei
    husnegt
char s[100];      // Zov: togtmol hemjeetei husnegt
int main() {
        int n;
        scanf("%d", &n);
        int a[n]; // Zov: n hemjeetei husnegt main() funkts
             dotor.
        return 0;
}
```

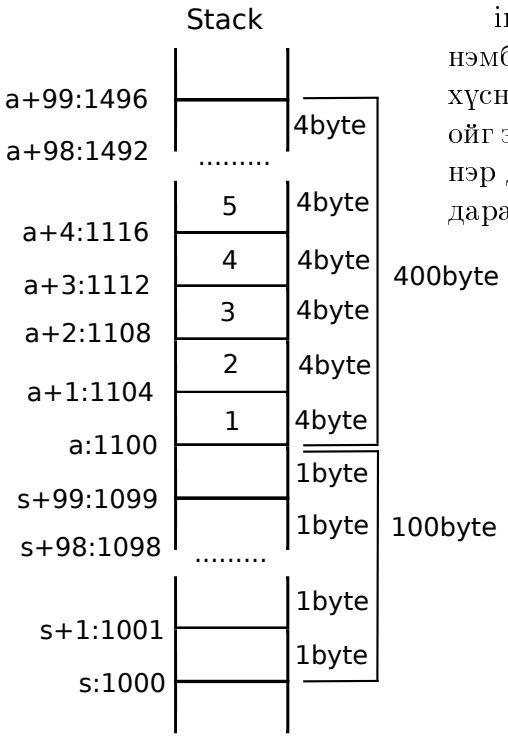
Хүснэгтийн нэр өөрөө хамгийн эхний элементийнхээ санах ойн хаягийн хадгална.

```
#include <stdio.h>
int main() {
        int a[100] = {1, 2, 3, 4, 5,};   // int husnegt
            zarlaj bna
        printf("%u", a); // 0-r elementiin hayg
        printf("%u", a + 1); // 1-r elementiin hayg
        printf("%u", a + 2); // 2-r elementiin hayg
        printf("%u", &a[2]);
        char s[100];
        printf("%u", s); // 0-r elementiin hayg
        printf("%u", s + 1); // 1-r elementiin hayg
        printf("%u", s + 2); // 2-r elementiin hayg
        printf("%u", &s[2]);
```

```
        return 0;
}
```

Зураг 3-т үзүүлснээр **a** хүснэгтийн санах ой 1000-аас эхэлж байна.

int төрлийн хүснэгтийн нэр дээр 1-ийг нэмбэл санах ой нь 4byte-аар нэмэгдэн хүснэгтийн дараагийн элементийн санах ойг заана. Харин char төрлийн хүснэгтийн нэр дээр 1-ийг нэмбэл 1byte-аар нэмэгдэн дараагийн элементийг зааж байна.



Зураг 3: хүснэгт

Үүнээс дүгнэхэд тухайн төрлийн хаягийг n-ээр нэмэгдүүлэхэд үнэндээ хаяг нь **n * төрлийн_хэмжээ** byte-аар нэмэгдэнэ.

Тэгэхээр доорх үйлдлүүд нь бүрэн боломжтой.

```
#include <stdio.h>

int main() {
        int a[100];
        *a = 1;         // a[0] = 1;
        *(a + 1) = 2;   // a[1] = 2;
        2[a] = 3;       // a[2] = 3;
        a[3] = 4;

        int *p = a;
        *p = 0;         // a[0] = 0;
        p[4] = 5;
```

```
        *(p + 2) = 0;  // a[2] = 0;
        return 0;
}
```

Жишээ: Тэмдэгт мөрийг %s хэрэглэлгүй хэвлэх.

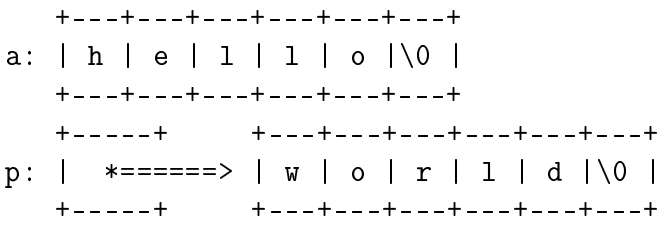
```
#include <stdio.h>
int main() {
        char s[] = "sain"; // temdegt mor (" ") dotor
            bichigddeg
        // togsohdoo zaaval '\0' temdegteer togsdog
        printf("%u\n", sizeof(s)); // yagaad 5 ve?
        int i;
        char *p;
        for (p = s;  // p-d s husnegtiin ehleliin haygiig
            hadgalj bna
          *p != '\0'; // sanah oid temdegt mor togsol
              zaagch bga eseh
          p++)        // sanah oig 1-eer nemegduuleh
             printf("%c", *p); // p sanah oid bga neg
                 temdegtiig hevleh

        printf("\n");
        return 0;
}
```

1.3 Би *char a[]* нь *char *a*-тай адилхан гэж бодсон юм?

Ерөөсөө үгүй. Хүснэгтүүд бол хаяган хувьсагчид (заагчууд) биш. *char a[6]* гэдэг бол *a* гэж нэрлэгдэх 6 ширхэг тэмдэгтийн зайг үүсгэхийг зарлаж байна. Энэ нь *a* гэж нэрлэгдсэн газар 6-н ширхэг тэмдэгт сууж болно гэсэн үг. Харин *char *p* нь *p* гэж нэрлэгдэх хаяган хувьсагчийг хадгалах зайг хүсэж байна. Энэ хаяган хувьсагч нь бараг бүхий л санах ойг зааж чадна: ямар ч *char*, *char* төрлийн дараалсан ямар ч хүснэгт, эсвэл хаашаа ч үгүй.

```
char a[] = "hello";
char *p = "world";
```

Энэ нь доор харагдаж байгаа шиг бүтцийг үүсгэх болно:



x[3] гэсэн код x нь хаяган хувьсагч уу эсвэл хүснэгт үү гэдгээс хамааруулан өөр өөр код үүсгэнэ гэдгийг мэдэх хэрэгтэй. Компайлар *a[3]* үйлдэлд a хаягаас эхлэн, хойш 3 яваад, тэнд байгаа тэмдэгтийг авах код гаргана. Харин *p[3]* үйлдэлд p хаягаас эхлэн, тэнд байгаа хаяган утгыг авч, тэр хаягаас эхлэн, хойш 3 яваад, тэнд байгаа тэмдэгтийг авах код гаргана. Энд хоёул 'l' тэмдэгтийг буцааж байгаа боловч тэнд өөр өөрөөр очиж байна.

1.4 Функцийн параметрт хүснэгтийг дамжуулах

```
void f(char a[])
{ ... }
```

Энд a нь хүснэгт биш хаяган хувьсагч юм.
Угаасаа эдгээр функцүүдэд хүснэгт дамжуулахад хаяг л хүлээж авна.

```
void f(char *a)
{ ... }
```

```
int a[100];
f(a);
```

Энд хүснэгтийг хаягаар нь өөрчлөх учраас функцэд хийсэн өөрчлөлт дамжуулсан хүснэгтэд нөлөөлнө.

1.4.1 Хоёр хэмжээст хүснэгтийг функцэд дамжуулах

Доор *NROWS* мөртэй, *NCOLUMNS* баганатай хоёр хэмжээст хүснэгтийг *void f(int (*)[])* функцэд дамжуулж байна.

```
int array[NROWS][NCOLUMNS];
f(array);
```

Функцийн тодорхойлолт болон зарлалт доорхтой тохирох ёстой:

```
void f(int (*)[]); // zarlalt

void f(int a[][NCOLUMNS]) // todorhoilolt
{ ... }

// esvel
void f(int (*ap)[NCOLUMNS])
{ ... }
```

Функц тухайн хүснэгтийн хаягийг л авч байгаа болохоос тухайн хүснэгттэй ижил хэмжээний хүснэгт үүсгэхгүй. Тийм учраас мөрийн хэмжээг мэдэх шаардлагагүй. Гэвч баганын хэмжээ чухал.

1.5 Тэгэхээр хүснэгт, хаяган хувьсагчийн ялгаа юу вэ?

Хүснэгт	**Хаяган хувьсагч**
1. Хүснэгт бол ганцхан, өмнөөс хуваарилагдсан, дараалсан элементүүдийн (бүгд нэг төрөл) хэсэг, хэмжээ ба байрлал нь тогтмол. 2. Санах ой автоматаар stack дээр хуваарилагддаг, ба ахин хэмжээ, хаяг нь өөрчлөгдөхгүй.	1. Хаяган хувьсагч хаа нэгэн газар байгаа ямар нэгэн элементийг (тодорхой төрлийн) заана. 2. Ахин өөр хаяг зааж болдог (malloc ашиглан санах ой хуваарилаад хэмжээг нь өөрчилж болдог). 3. Хаяган хувьсагч хүснэгтрүү зааж болдог ба динамикаар хүснэгт үүсгэн (malloc-оор) хүснэгтийг дүрсэлж чаддаг. Хаяган хувьсагч нь илүү ерөнхий төрөл.
malloc ашиглан хуваарилсан блок санах ой яг хүснэгт шиг л хэрэглэгдэж болдог учраас хүснэгт, хаяган хувьсагч хоёр адил мэт санагддаг. Гэхдээ sizeof оператор энэ хоёр тохиолдолд өөр үр дүн буцаахыг анхаарна уу.	

2 Дасгалууд

2.1 Ангид

1. Доорх хувьсагчуудын санах ойн хаягийг хэвлэн харуул.

```
int a[100];
int b = 4;
double c = 5;
int *p = &b;
```

2. Хоёр тооны утгыг сольдог доорх функцийг бич.

```
void swap(int *a, int *b);
```

3. Хүснэгтээс хамгийн их элементийн хаягийг олох програмын алдааг ол.

```
#include <stdio.h>

int main() {
        int a[100];
        int *p = a;
        int i, n;
        scanf("%d", &n);
        for (i = 0; i < n; i++)
                scanf("%d", a + i);

        int *max;
        *max = p[0];
        for (i = 0; i < n; i++)
                if (max > *(p + i))
                        max = p + i;

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

4. Хүснэгтийн эхлэл болон төгсгөлийн хаяг өгөгдөхөд агуулсан элементүүдийг эрэмбэлэх доорх функцийг хэрэгжүүл.

```
void sort(int *begin, int *end);
```

2.2 Гэрт

N ширхэг үг өгөгдөхөд тэдгээрийг хоёр хэмжээс *char* төрлийн хүснэгтэд мөр бүрд нь нэг үгийг хадгалах замаар хадгалан доорх функцүүдийг хэрэгжүүл. Үг бүрийн урт 10-аас хэтрэхгүй. Жишээ:

```
scanf("%d", &n);
char a[n][10];
int i;
for (i = 0; i < n; i++)
    scanf("%s", a[i]);
```

1. Хоёр хэмжээст хүснэгтэд хадгалагдсан үгүүдийг хэвлэх функц.

```
void print(char a[][10], int size);
```

2. Хоёр хэмжээст хүснэгтэд хадгалагдсан үгүүдийг эрэмбэлэх функц, *strcmp* функц ашиглан эрэмбэлнэ.

```
void sort(char a[][10], int size);
```

```
Оролт:
3
robot
car
apple
Гаралт:
apple
car
robot
```

3. Параметрээр дамжуулсан үгийг хоёр хэмжээст хүснэгтээс хайн олдвол байрлалыг, үгүй бол -1 утгыг буцаах функц.

```
int search(char a[][10], char *word);
```