1 Хүснэгт, хаягийн арифметик

Санах ойд дараалллан байрласан элементүүдийн цуваaг хүснэгт гэж нэрлэдэг.

Дүрэм:

төрөл нэр[хэмжээ];

Санах ойд дараалллан 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.

```
#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.

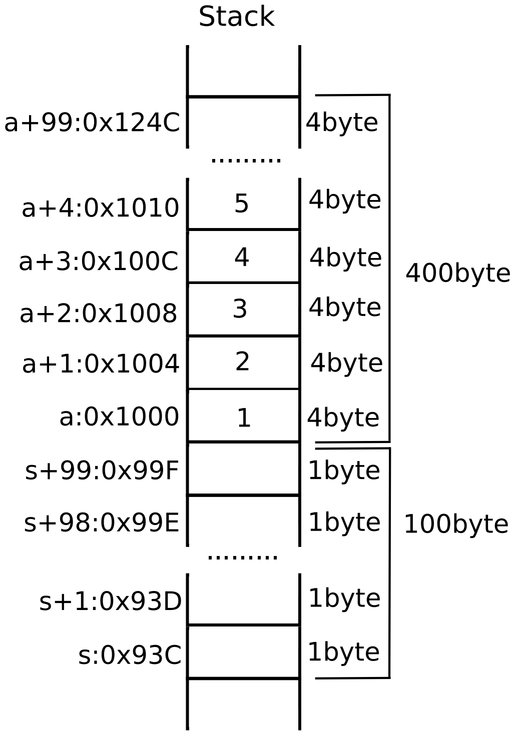
Хүснэгт файлийн хүрээнд зарлагдвал хэмжээ нь заавал тогтмол байна.

```
#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 1: хүснэгт

Зураг 1-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 заах байна.

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;
}
```

1.1 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:
   +---+---+---+---+---+---+
a: | h | e | l | l | o |\0 |
   +---+---+---+---+---+---+
   +-----+     +---+---+---+---+---+---+
p: |  *======> | w | o | r | l | d |\0 |
   +-----+     +---+---+---+---+---+---+
```

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.

1.2 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.

1.2.1 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.

1.3 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.	1. Хаяган хувьсагч хаа нэгэн газар байгаа ямар нэгэн элементийг (тодорхой төрлийн) заана.
2. Санах ой автоматаар stack дээр хуваарилагддаг, ба ахин хэмжээ, хаяг нь өөрчлөгдөхгүй.	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. Хаягийн арифметик ашиглан хүснэгтийн элементүүдийн хаягийг дараах байдлаар авч гараас утга уншиж болно. Тэгвэл хаягийн арифметик ашиглан хүснэгтийн утгуудыг хэвлэ.

```
int main()
{
  int i, a[100];
  for (i = 0; i < 5; i++)
      scanf("%d", a + i);     // &a[i]

  // xevlex uildel xiine uu.
}
```

2. Хүснэгтийн эхлэл болон, төгсгөл хаягийг дамжуулахад хэмжээ нь хэд байгааг олох *size* функцийг хэрэгжүүл.

```
int size(int *begin, int *end);

int main()
{
  int i, n;
  scanf("%d", &n);
  int a[n];
  printf("%d\n", size(a, a + n)); // n-ii utgiig xewlene.
  return 0;
}
```

3. Хүснэгтийн эхлэл болон, төгсгөл хаягийг дамжуулахад хүснэгтээс х тоог хайж, байрлалын хаягийг буцаах *find()* функцийг хэрэгжүүл. Олдохгүй бол *NULL* хаягийг буцаана.

```
int *find(int *begin, int *end, int x);

int main()
{
  int a[5] = {3, 7, 1, 2, 4};
  int *p, x;
  printf("xaix too: ");
  scanf("%d", &x);
  p = find(a, a + 5, x);
  if (p == NULL)
     printf("%d too oldsongui\n", x);
  else
     printf("%d too %d bairlald oldloo\n", x, p - a);
  return 0;
}
```

4. Дээрх find функцийг ашиглан хүснэгтээс х тоо олдох бүрд у утгаар солих програм бич.

5. Хоёр хүснэгтэн хаяг дамжуулахад завсар байгаа утгуудыг эсрэгээр нь хадгалах функцийг бич.

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

int main()
{
  int a[5] = {3, 7, 1, 2, 4};
  reverse(a, a + 3);
  int i;
  for (i = 0; i < 5; i++)
     printf("%d ", a[i]);     // 1 7 3 2 4
  return 0;
}
```