Саптар

Жалпы маалымат.

С++ тилинде сап катары символдордун удаалаштыгын алышат жана ар бир саптын аягына компилятор тарабынан нөлдүк символ (\0) коюлат, бул символду 0 цифрасы менен алмаштырбоо керек. Мисалы “Си тили” деген сапты карасак 8 элементтен турган символдук массив катары берилет, эгерде массивтин ысымы а болсо, анда а[0]=’С’, а[1]=’и’, а[2]=’ ‘, а[3]=’т’,
а[4]=’и’, а[5]=’л’, а[6]=’и’, а[7]=’\0’

Эгерде сап туруктуу катары берилсе б.а. тырмакчанын ичинде жасылса, мисалы “Информатика” анда саптын аягына компилятор нөлдүк символду (‘\0’) автоматтык түрдө кошот. Ал эми калган учурда, сапты айкын түрдө берүүдө программист саптын аягына нөлдүк символду (‘\0’) коюуга тийиш.

Мисалы:

char sap[2]={‘С’, ‘и’, ‘\0’}.

Сапты символдордун массиви же символдордун удаалаштыгы катары карашат. Сапты символдордун массиви катары кароодо, массивтерди маанилештирүүнү колдонсо болот.

Мисалы:

char str[10]={‘Т’, ‘u’, ‘r’, ‘b’, ‘o’, ‘ ‘, ‘С’,‘\0’}

Ал эми символдордун удаалаштыгы катары берилгенде,

Мисалы:

char str[10]={“Turbo C”}

саптын аягына 8-символ катары нөлдүк символ (‘\0’) автоматтык түрдө коюлат.

Саптарды киргизүүдө Си тилинде эки функция каралган scanf жана gets. Эки функциянын айырмасы: scanf – функциясы ачык (пробел) символу кезиккенге чейин же Enter клавишасын басканга чейин сапты окуп киргизет, ал эми gets – функциясы сапты <Enter> клавишасын басканга чейин окуп киргизет, ал эми ачык (пробел) символун саптын маниси катары эсептейт.

Сапты чыгарууну карайлы. Мейли char a[10] берилсе printf(“%s”, a) функциясынын иштөөсү төмөндөгүдөй, бул функция аткарылганда, а көрсөткөн биринчи элементтин адреси берилет. Эгерде биринчи элемент нөлдүк символ (‘\0’) болсо, чыгаруу аяктайт, ал эми нөлдүк символ болбосо, ал элемент экранга чыгарылат да учурдагы адреске бир кошулуп кийинки элементин маниси текшерилет, ушундай иш аракеттер сап экранга толук чыгарылып бүтмөйүнчө улана берет. Саптарды киргизүү мисалдары:

```
# include <stdio.h> #include
<conio.h >
    main( )
    {
    char sap[20]; clrscr();
    printf(“\n Ысымыңыз ким ?”); scanf(“%s”, sap);
```

printf(“Саламатсыңбы, %s \n”, sap);
}
Жыйынтык:
ЫсымыңызкимNуртегинСаламатсыңбы, Нуртегин

Жогорку программада sap символдордун массиви ошол эле учурда sap массивдин адреси болгондуктан scanf – киргизүү функциясында адрести аныктоочу & амалы, sap ысымынын алдына коюлбайт.

Программанын иштеши: Экранга, ысымыңыз ким? деген суроо чыгат. Бул сурооого “Акмат“ деп жооп берсеңиз, анда программа, экранга “Саламатсыңбы Акмат“ деген сүйлөмдү чыгарат. Эгерде сурооого “Акмат Асанов” деп жооп берилсе, анда экранга “Саламатсыңбы Акмат” деген эле кабар чыгарылат. Себеби Акматтан кийинки коюлган ачык (‘ ‘) белги scanf – функциясы үчүн киргизүүчү саптын аягын билдирет. Мындай кемчиликти жоюуш үчүн: Эки сапты киргизүү менен же gets сапты киргизүү функциясын колдонуу менен жойсо болот. Бул учурлар үчүн мисал карайлы:

Мисал 1.

```
# include <stdio.h>
#include <conio.h> main (
)
{char sap1 [15], sap2 [20]; clrscr();
printf(“\n Ысымыңызданафамилияңыздыкиргиз:”); scanf(“%s%s”, sap1, sap2);
printf(“Саламатсыңбы, %s %s\n”, sap1,sap2);
}
```

Жыйынтык:

Ысымыңызданафамилияңыздыкиргиз:

уртегин Максатбеков

Саламатсыңбы, Нуртегин Максатбеков

Бул программада эки саптык өзгөрмө sap1, sap2 колдонулду. Мисал 2.

```
# include <stdio.h> #include
<conio.h>
main( )
{char sap[20];
clrscr();
printf(“\n ЫсымыңызкимN:”); gets(sap);
printf(“Саламатсыңбы, %s\n”, sap)}
```

Жыйынтык:

ЫсымыңызкимN: Балтатыр Саламатсыңбы, Балтатыр

Экинчи программанын иштешинде gets – функциясы, колдонуучу киргизүү баскычын (“Enter”) басканга чейинки терген символдорунун баардыгын окуйт, бирок киргизилген саптын аягына ‘\0’ нөлдүк символун жайгаштырат. Эгерде сапты киргизүү

мезгилинде саптын узундугу, берилген сандан ашып кетсе, ашыкча символдор сапка берилген эстин областынын башка бөлүгүнө жазылат. Сап – символдордун массиви болгондуктан, массивдин ысымы массивтин биринчи элементинин көрсөткүчү болот. Ошондуктан программада саптарды же символдордун массиви же символго болгон көрсөткүч катары баяндаса болот. Мисалы char m[20], *m1 бул жерде “char m[20]” m массиви 20 символдон турат жана бул сап үчүн учурдагы эстен орун бөлүнөт. Ал эми “char *m1” болсо сапты символго болгон көрсөткүч катары баяндайт. Бул учурда эстен айкын түрдө орун бөлүнүп, m1 – символго болгон көрсөткүч болгондуктан m1 же символдордун массиви же баштапкы адреси m1 болгон сап. Бир эле программада сапты массив катары да көрсөткүч катары да караса болот. Мындай колдонуу сапты функциянын аргументи катары берген учурда өзгөчө маанилүү болуп эсептелет.

Мисалы: #include
<stdio.h> #include
<conio.h >

```
main( )
{char sapmass[10], *p; clrscr();
  sapmass[0]=’C’;
  sapmass[1]=’a’;
  sapmass[2]=’л’;
  sapmass[3]=’a’;
  sapmass[4]=’м’;
  sapmass[5]=’\0’; p=
  sapmass; puts(p);
  printf(“\n p=%s\n”, p);
  puts(sapmass);
  while(*p!=NULL)  putchar(*p++);
}
```

Жыйынтык:

Салам p=
Салам
Салам
Салам

Бул программада 3 – сапта символдордун удаалаштыгын массив жана көрсөткүч түрүндө баяндап жатат, ал эми 4 – саптан
9 – сапка чейин символдордун массиви айкын түрдө маанилештирилген 10- сапта p символдук массивти менчиктеп алат. 11 – саптагы puts(p) функциясы көрсөткүчтүн мааниси боюнча символдук массивти экранга чыгарат б.а. “Салам” деген сөз чыгат. 12 – сапта printf функциясы сапты толугу менен чыгарат. 13 – саптагы puts функциясы sapmass сабын экранга чыгарат.

14 – саптагы while цикл оператору p көрсөткүчү кармаган адресте жайгашкан символ NULL турактуусунан айырмалуу болсо putchar функциясы *p++ адрести бирге

чоңойтуу менен улам кийинки адресте жаткан символдорду чыгара берет. Чыгаруу процессинде ал адрестеги символдорду жоюп салат.

Саптардын өзгөчөлүктөрү.

Саптарды туруктуу жана өзгөрмөнүн мааниси катары кароого болот.

Саптарды туруктуу катары караган учурда аларды берүүнүн эки жолу бар.

1. Программанын текстинде сап туруктуусу кош тырмакчага алынып жазылат. Бул учурда компилятор сап туруктуусунун аягына “\0” нөлдүк символун автоматтык түрдө кошот.

2. define директивасынын жардамы менен *мисалы:*

#define sap “Си тили”

… puts(sap);

Бул программанын фрагментинде sap туруктуу катары каралып анын мааниси “Си тили” экранга puts –функциясы менен чыгарылат.

Саптарды өзгөрмөнүн мааниси катары караган учурда, сапты аныктоонун да эки жолу болот.

1. Символдук массивти колдонуу.

2. Символго болгон көрсөткүчтү колдонуу.

Символдук массивти колдонуу менен сапты берүүгө *мисал:*

```
#include <stdio.h> #include
<conio.h > #define n 10
    main( )
    {char sap[n];
    clrscr(); sap[0]=’C’;
    sap[1]=’и’;
    sap[2]=’ ’;
    sap[3]=’т’;
    sap[4]=’и’;
    sap[5]=’л’;
    sap[6]=’и’;
    sap[7]=’\0’; puts(sap);
    }
```

Жыйынтык:

Си тили

Символго болгон көрсөткүчтү колдонуу менен сапты берүүгө мисал:

```
#include <stdio.h> #include
<conio.h >
    main( )
    {char *sp; clrscr();
```

```
sp=”Си тили”;
puts(sp);
printf(“sp=%s\n”,  sp);
}
```

Жыйынтык:

Си тили

Саптар менен иштөөдө байкалбаган каталар кетип калышы мүмкүн. Мисалы:

```
#include <stdio.h> main( )
{char *sap; char
sp[10];
printf(“\n Сапты киргиз:”);
scanf(“%s”, sap);
sp=”Киргизилген сап”;
printf(“%s%s\n”, sp,sap);
}
```

Жыйынтык:

Сапты киргиз: тттт

Null pointer. Assigment

Жогорку программадагы 6 саптагы scanf (“%s”,sap); функциясынан ката кеткен. Кетирилген катанын мааниси төмөндөгүдөй sap үчүн эстен орун бөлүнбөйт, киргизилген сап кандайдыр бир башка адреске жазылат. Мындай ката программанын иштешин токтотпойт.

Экиничи ката sp=”Киргизилген сап” менчиктөө операторун колдонгондон кеткен. Компилятор “Киргизилген сап” деген саптык турактуулуктун адресин sp - өзгөрмөсүнүн маанисине алмаштырууга аракет катары эсептейт. Мындай иш аракет туура эмес, себеби массивтин ысымы турактуулук (константа) катары каралат б.а. турактуууга өзгөрмөнү (9=a) менчиктөө менен тең күчтө болуп калат, мындай менчиктөө туура эмес.

Жогорку ката жазылган программанын туура жазылышы.

```
# include <stdio.h> #include
<conio.h >
    main ( )
    { char sap [10]; char
    *sp; clrscr();
    printf (“\n Сапты киргиз:”); scanf
    (“%s”, sap); sp=”Киргизилген сап:”;
    printf (“%s%s\n”, sp, sap);
```

```
}
```

Жыйынтык:

Сапты киргиз: Манас Киргизилген

сап: Манас

Саптарды иштетүүчү программаларга токтололу.

1-Мисал: strlen камтылган программаны колдонбой киргизилген саптын узундугун аныктоочу программа.

```
# include <stdio.h> #include
<conio.h >
    main ( )
    { char  *sap; int i
    ; clrscr();
    printf ("\nСапты киргиз \n"); gets (sap);
    for (i=0; sap[i] !=0; i++);
    printf ("Киргизилген саптын узундугу=%d", i);
    }
```

Жыйынтык:

Сапты киргиз: Манас эпосу Киргизилген

саптын узундугу=11

2-Мисал: символдук массивди колдонуу менен бир сапты экинчи сапка менчиктөө.

```
# include <stdio.h> #include
<conio.h >
    main ( )
    { char  sap1[10], sap2[10]; int i ;
    clrscr();
    printf("\n Сапты киргиз sap1="); gets (sap1);
    i=0;
    do
    {sap2[i]=sap1[i];} while
    (sap1[i++]!=0);
    printf("sap2=");
    puts (sap2);
    }
```

Жыйынтык:

Сапты киргиз sap1=Тагай бий

sap2= Тагай бий

3-Мисал: сапты киргизүүнүн жана чыгаруунун ар түрдүү жолдору.

```
# include <stdio.h> #include
```

```
<conio.h >
    main ( )
    { char  smas [10], *sap; clrscr();
    printf (“\n Сапты киргиз smas=”); gets (smas);
    sap=smas;  /* sap көрсөткүчү smas сабынан башталыш адресин менчиктейт */
    printf (“smas=”);

    puts (smas); printf
    (“sap=”); puts (sap);
    printf (“sap=”);
    while (*sap!=NULL) putchar (*sap++);
    }
```

Жыйынтык:

```
Сапты киргиз smas=Барсбек каган
smas=Барсбек каган sap=
Барсбек каган sap= Барсбек
каган
```

3-мисалда сапты киргизүүнүн эки жолу жана чыгаруунун үч жолу берилген, киргизүү gets(smas) функциясы жана sap=smas менчиктөө оператору менен берилсе. Чыгаруу puts(smas), puts(sap) жана адрес аркылуу чыгаруу putchar (*sap) функциялары менен берилген.

Саптар менен иштөөчү стандарттык функциялар.

Саптар менен иштөөчү стандарттык функциялар Турбо-Си системасында string.h файлында баяндалган функцияларды баяндоонун жалпы түрү.

<жыйынтыктын тиби> <функциянын ысымы> (<формалдуу параметрлер);

Саптарды бириктирүүчү функциянын эки түрү бар.

1. char *strcat(sap1, sap2); бул функция sap1 сабынын аягына sap2 сабынын көчүрмөсүн бириктирет.

Мисалы sap1=”эсен” sap2=”аман” анда strcat(sap1, sap2) функциясынын жыйынтыгы «эсенаман» деген маанини берет.

2. char *strncat(sap1, sap2, k); sap2 сабынан k сандагы символдун күчөрмөсүн бириктирип келип чыккан саптын аягына “\0” нөлдүк символду коёт.

Мисалы sap1= “Асан”, sap2=”баймат”

strncat (sap1, sap2, 3) функциясынын жыйынтыгы “Асанбай” болот

```
#include <stdio.h> #include
<conio.h > #include <string.h>
    main ( )
```

```
{ char *sap1, *sap2, *ch1, *ch2; clrscr();
sap1=”Эсен”;
sap2=”аман”; strcat
(sap1, sap2); puts (sap1);
ch1=”Асан”;
ch2=”баймат”;
strncat (ch1, ch2, 3); puts
(ch1);
}
```

Жыйынтык:

Эсенаман

Асанбай

Саптарды салыштыруу функциясы

int strcmp (sap1, sap2)

sap1 менен sap2 саптары салыштырылат. Салыштыруу, саптарды түзгөн символдордун коддору менен жүргүзүлөт. Салыштырууда салыштырылып жаткан символдордун коддору кайсынысы чоң болсо ошол сапты чоң деп эсептейт. Эгерде ал символдордун коддору барабар болсо кийинки символдор салыштырылат. Мындай иш аракеттер символдордун коддорунун кайсынысы чоң же кичине болгонго чейин жүргүзүлөт.

Си тилинде саптарды салыштыруу үчүн салыштыруу функциясынын үч түрү каралган. Бул функциялардын жыйынтыктарынын бардыгы бүтүн типте болушат.

1. int strcpm (sap1, sap2); sap1 менен sap2 салыштырылат, эгерде sap1<sap2 болсо функциянын жыйынтыгы терс болот, sap1==sap2 болсо strcpm функциясынын мааниси 0 болот, ал эми sap1>sap2 болсо функциянын мааниси оң болот.

2. int stricmp (sap1, sap2) бул функция саптагы баш (чоң) тамгалар менен кичине тамгаларды айырмалабай sap1 жана sap2 салыштырылат.

3. int strncpm (sap1, sap2, k) бул функция саптагы баш (чоң) тамгалар менен кичине тамгаларды айырмалабай, бирок k сандан ашпаган гана символдорду салыштырат.

1-Мисал:

```
# include <stdio.h> #
include <string.h> #
include <conio.h >
main ( )
{
clrscr();
printf (“%d \n”, strcmp (“A”, “A”)); /* 0 саны чыгарылат*/ printf (“%d \n”, strcmp
(“A”, “B”)); /* -1 саны чыгарылат*/ printf (“%d \n”, strcmp (“C”, “B”)); /* 1 саны
чыгарылат*/ printf (“%d \n”, strcmp (“D”, “A”)); /* 3 саны чыгарылат*/
```

```
}
```

Жыйынтык:

```
0
-1
1
3
```

2-Мисал:

```
# include <stdio.h> #
include <string.h> #
include <conio.h >
  clrscr(); main (
  )
  { char sap1[20], sap2[20];
   printf ("\n sap1 жана sap2 саптарын 20дан ашпаган узундукта киргиз \n");
   printf ("\n sap1 киргиз:"); scanf
   ("%s", sap1);
   printf ("\n sap2 киргиз"); scanf
   ("%s", sap2);
   if (strcmp (sap1, sap2)<0) printf ("\n sap1<sap2"); if (strcmp (sap1,
   sap2)>0) printf ("\n sap1>sap2"); if (strcmp (sap1, sap2)==0)printf
   ("\n sap1=sap2");
  }
```

Жыйынтык:

```
sap1 жана sap2 саптарын 20дан ашпаган узундукта киргиз
sap1 киргиз: AAA sap2
киргиз: DDD sap1<sap2
```

Саптарды көчүрүү.

Си тилинде саптарды көчүрүү үчүн strcpy-функциясынын эки түрү каралган.

1-түрүнүн жалпы берилиши char * strcpy(sap1, sap2); sap2 сабынын баарысын sap1 сабына көчүрөт.

2-түрүнүн жалпы берилиши char * strcpy(sap1, sap2, k); sap2 сабынан k сандагы символду sap1 сабына көчүрөт.

1-түрдөгү көчүрүү функциясына мисал.

```
# include <stdio.h> #
include <string.h> #
include <conio.h> main ( )
  {char s1[10], s2[10]; clrscr ( );
```

```
    printf (“s2 сабын киргиз:”); scanf
    (“%s”, s2);
    strcpy(s1, s2);
    printf (“s1=%s\n”, s1);
  }
```

Жыйынтык:

s2 сабын киргиз: Атилла

s1= Атилла

Көчүрүү функциясынын 2-түрүнүн саптын ортосундагы символдорду көчүрүүгө мисал карайлы.

```
# include <stdio.h> #
include <string.h> #
include <conio.h>
    main ( )
    {char s1[6], s2[10]; clrscr (
    );
    printf (“s2 сабын киргиз:”); scanf (“%s”,
    s2);
    strncpy(s1, &s2[2], 6);
    printf (“%s\n”, s1);
    }
```

Жыйынтык:

s2 сабын киргиз: informatika format

Саптын узундугу.

Саптын узундугун strlen-кызматчы сөзү менен берилген функция аркылуу табылат. Бул функциянын жалпы берилиши: int strlen (sap)

sap канча символдордон турса ошол символдордун санын чыгарып берет.

Мисалы: strlen(“Си программалоо тили”) жыйынтыгы 20санын берет, себеби көрсөтүлгөн сап 20 символдон турат. Сапты аяктоочу нөлдук символ саптын узундугунун санына кирбейт.

Мисалы: Клавиатура аркылуу киргизилген саптын узундугун берүүчү программа.

```
# include <stdio.h> #
include <string.h> #
include <conio.h>
    main ( )
    {char s[80];
    clrscr ( );
    printf (“Сапты киргиз:”); gets (s);
    printf (“Берилген сап: %s \n %d                символдон турат \n”, s,
```

```
strlen(s));
    }
```

Жыйынтык:

Сапты киргиз: Информатика Берилген

сап: Информатика 11 символдон турат

Саптан символдорду издөө.

Символдорду издөө Си тилинде strchr-кызматчы сөзү менен берилген функция аркылуу жүргүзүлөт. Бул функциянын эки түрү бар.

1-түрүнүн жалпы берилиши char *strchr (<берилген сап>, <изделүүчү символ>). Бул функция берилген сапта <изделүүчү символ> биринчи кезиккен орундан баштап саптын аягына чейин чыгарып берет. Эгерде <изделүүчү символ> кезикпесе анда функция NULL деген маанини чыгарат.

Мисалы: char *strchr (sap, ‘C’). мисалдагы функция ‘C’ тамгасы сапта биринчи кезиккен орундан баштап аягына чейин символдордун удаалаштыгын чыгарып берет.

2-түрүнүн жалпы берилиши char *strrchr (<берилген сап>, <изделүүчү символ>). Бул функция берилген сапта <изделүүчү символ> акыркы кезиккен орундан баштап саптын аягына чейин чыгарып берет.

Мисалы char *strrchr (sap, ‘C’) бул функция берилген сапта ‘C’ тамгасы акыркы кезиккенинен баштап саптын аягына чейинки бөлүгүн чыгарат.

```
# include <stdio.h> #
include <string.h>
 # include <conio.h > main ( )
      {char sap[10], *p, *q; clrscr();
        printf (“Сапты киргиз: ”); gets (sap);
        p=strchr(sap, ‘m’);
        q= strrchr(sap, ‘m’);
        printf (“Бул сап:  %s \n”, p); printf (“%s
        \n”, q);
      }
```

Жыйынтык:

Сапты киргиз: zxcvmasmkkk

Бул сап: masmkkk mkkk

Бир саптагы символдорду экинчи бир саптан издөөчү функция strpbrk-кызматчы сөзү менен берилет. Жалпы берилиши char *strpbrk(sap1, sap2) бул функция sap2-биринчи символу sap1

сабында кезиккен ордунан баштап sap1 сабынын аягына чейин чыгарып берет. Эгерде sap1 ден sap2нин бир дагы символу кезикпесе функция NULL деген маанини чыгарып берет.

Мисалы: strpbrk #

```
include <stdio.h>
# include <string.h> #
include <conio.h>
main ( )
{char sap1[10], sap2[10], *p; clrscr();
   printf (“sap1 ди киргизгиле: ”); gets (sap1);
   printf (“sap2 ни киргизгиле: ”); gets (sap2);
   p=strpbrk(sap1, sap2); printf (“%s
   \n”, p);
}
```

Жыйынтыгы:

```
sap1 ди киргизгиле: zxcvb sap2 ни
киргизгиле: rtyxkum xcvb
```

Түшүндүрмө: sap1дин мааниси катары “zxcvb” берсек ал эми sap2 мааниси катары “rtyxkum” киргизсек жообу: “xcvb”-сабы болот. Издеп табылган саптардын узундуктары. int strspn (sap1, sap2)-функциясы менен табылат. Бул функцияга мисал карайлы.

```
# include <stdio.h> #
include <string.h> #
include <conio.h> main ( )
  {char *sap1=”d8656bn8”;
  char *sap2=”2td86bnc”; int t;
  clrscr();
  t=strspn (sap1, sap2);
  printf (“sap2де кезиккен sap1деги символдордун
удаалаштыгынын узундугу =%d\n”,t)
   }
```

Жыйынтыгы:

sap2де кезиккен sap1деги символдордун удаалаштыгынын узундугу =3 Түшүндүрмө: sap2деги ”2td86bnc” саптан d86 удаалаштыгы sap1деги “d8656bn8” маанинин биринчи 3 символу болгондуктан узундугу 3кө барабар.