ЛЕКЦИЯ 6
ЖОЛДАР ЖӘНЕ ЖОЛДАРДЫҢ ТҮРЛЕРІ

Жоспары:
Жолдар және жолдардың түрлері.
Символық жолдарды өңдеу функциялары.
Жолды символдық массив түрінде анықтау.
String класы. AnsiString типі.

С тілінде сөз тіркестерін өңдеу кезінде қолданылатын тип char . Сөз тіркестері char типті бір өлшемді жиым ретінде қарастырылады, яғни сөз тіркесі – нөлдік байтпен аяқталатын char типті бір өлшемді жиым. Нөлдік байт – барлық биттері де нөлге тең байт, ол ´ \0 ´ символдық константасымен анықталады (тіркес соңы белгісі немесе нөл-терминатор). Сондықтан егер тіркесте k символ болса, онда жиымды сипаттауда k+1 элемент көрсетілуі тиіс.

Мысалы, **char a[7]** деген сипаттау тіркестің 6 символдан тұратынын, ал соңғы байт нөлдік екенін білдіреді. Си тіліндегі тіркестік (жолдық) константа – қос тырнақшаға алынған символдар жиыны. Мысалы, “Берілген тапсырма” тіркесі, оның соңына нөлдік байтты компилятор автоматты түрде өзі жазады.

Айнымалы мәні болатын сөз тіркесін сипаттау кезінде бірден көрсетуге болады, мысалы,
char S1[10]=”123456789”, S2[]=”Болат”;

Соңғы сөз ұзындығы тіркестің символдары санымен анықталады.

Символдар тіркесін пернелерден енгізу үшін екі стандартты функция – **scanf()**немесе **gets()**қолданылады, ал олардың прототиптері **stdio.h** тақырыптық файлында болады.

Символдық таңбаларды енгізу/шығару

Символдарды біртіндеп енгізу/шығару үшін **printf()** және **scanf()** функцияларының **%c** форматы қолданылады.

getch() – параметрсіз функция, басылған перненің кодын (int) береді, экранға ешқандай символ шығармайды.

getchar() – параметрсіз функция. Пернеден символдарды бір-бірлеп енгізеді. Сөз тіркесі <**Enter**> пернесі басылғанша енгізіле береді, оған дейін оны өзгертуге де болады.

putch(c) – бір символды (c – символдық айнымалы немесе константа), яғни бір таңбаны ғана экранға шығарады.

putchar(c) – бұл да тек бір таңбаны экранға шығарады.

Бұлар conio.h тақырып файлы бойынша жұмыс істейді.

Мысалы, латын алфавиті әріптерін экранға шығару программасы төмендегідей болады:

```
#include <conio.h>
#include <stdio.h>
void main()
{
 char z;
 clrscr();
 for(z='A';z<='Z';z++)
    putchar(z);
 getch();
}
```

Нәтижесі:
ABCDEFGHIJKLMNOPQRSTUVWXYZ

Ал енді осы символдарды ASCII-кодтарымен бірге шығаратын мына программаны көрейік.

```
/* латын алфавиті*/
#include <conio.h>
#include <stdio.h>
void main()
{
 char z; clrscr();
 for(z='A';z<='Z';z++)
 {   if (z=='K' || z=='U') printf("\n");
   printf(" %c-%d ",z,z);  }
 getch(); }
```

Программа жұмысы нәтижесі:
A-65 B-66 C-67 D-68 E-69 F-70 G-71 H-72 I-73 J-74

K-75 L-76 M-77 N-78 O-79 P-80 Q-81 R-82 S-83 T-84
U-85 V-86 W-87 X-88 Y-89 Z-90

Келесі программа 0 мен 9 арасындағы цифрлық символдарды және олардың ASCII кодтарын басып шығарады:

```
#include <conio.h>
#include <stdio.h>
void main() {
 char z;  clrscr();
 for(z='0';z<='9';z++)
  {   if (z=='0' || z=='5') printf("\n");
  }
 getch();
 }
```

Жұмыс нәтижесі:

0 – 48 1 – 49 2 – 50 3 – 51 4 – 52
5 - 53 6 – 54 7 – 55 8 – 56 9 – 57

Символдық тіркестер

Символдық жолдарды немесе тіркестерді бірнеше тәсілмен өңдеуге болады, олардың негізгілері:

1. Тіркестік константаларды қолдану;
2. Char типті жиымды қолдану;
3. Char типіне сілтейтін нұсқауыштарды пайдалану;
4. Символдық тіркестерден тұратын жиымдарды қолдану.

Сөз тіркестері немесе тіркестік (жолдық) константа қостырнақшаға алынып жазылады. Тырнақшаға алынатын символдар тізбегінің ең соңына автоматты түрде ‘\0’ символы жазылады. Компилятор жолдық символдарды компьютер жадына жазғанда, олардың көлемін анықтау үшін сол символдар санын есептейді. Символдық константа осы сөз тіркесі жазылған жады аймағына сілтейтін нұсқауыш болып табылады. Символдық тіркестер жиымын (массивін) беру кезінде компилятор компьютер жадының қажетті көлемін анықтау үшін жиымды сипаттағанда, оны тіркестік константа арқылы инициалдауға болады. Мысалы:

char c[]= “Атырау, Алтай - жеріміз”;

Әдеттегі жиым қолданылатын жағдайлар сияқты бұл жиым аты **c** осы жиымның 1-элементіне сілтейтін нұсқауыш болып табылады.

```
c ==&c[0];
*c ==’0’;
*(c+1) == c[1] == ‘n’;
```

Сөз тіркестерін анықтау үшін нұсқауыштарды мынадай түрде сипаттауға болады:

char *c1 = “\n студенттер саны”;

осы сипаттауға эквивалентті болып келесі сипаттау есептеледі:

static char c1[]=”\n студенттер саны”;

Осы қарастырылған екі сипаттау да **c1** тіркесінің нұсқауыш екенін білдіреді. Компьютер жадының қажетті көлемін айқын көрсетуге де болады. Мысалы, сыртқы сипаттауда келесі жолдың мынадай түрде жазылғаны көрсетілген.

char c[25]=”Білім - өмір шырағы”;

Элементтердің саны жолдың ұзындығынан бір символ артық болуы керек, яғни оның ең соңында ‘\0’ символы болуы тиіс.

Статикалық немесе сыртқы жиымдағы бұрынғы қарастырылған әдеттегі жиымдар оларды қолдану кезінде автоматты түрде 0-мен инициалданған болатын. Ал сөз тіркестерін пайдалану кезінде де статистикалық немесе сыртқы жиымдар солар тәрізді 0 символымен инициалданады.

Келесі мысалды қарастыралық:

```
#include <stdio.h>
#include <string.h>
main ()
{
 char msg[30];
 strcpy(msg, "Сәлем, Азат!");
 puts(msg);
}
```

Мұндағы msg сөзінен соң тұрған [30] саны компиляторға 29 символ үшін, яғни char типіндегі 29 айнымалыдан тұратын жиым үшін жады бөлуді қамтамасыз етеді (30-орын нөлдік символмен – \0 толтырылады). msg айнымалысының символдық мәні жоқ; ол тек char типіндегі 29 айнымалының алғашқысының адресін (компьютер жадындағы белгілі бір орын адресі) сақтайды.

Компилятор strcpy(msg, "Сәлем, Азат!") операторын кездестіргенде, екі түрлі әрекет орындайды:

- "Сәлем, Азат!" тіркесі соңына (\0) символын (ASCII коды 0) қосады.
- strcpy функциясын орындап, msg айнымалысы нұсқап тұрған жады аймағына сол сөз тіркесі символдарын біртіндеп көшіреді. Ол тіркесті көшіруді сөз соңындағы нөлдік символдан кейін барып аяқтайды.

puts(msg) функциясын орындағанда, оған msg мәні, яғни тіркес құрамындағы бірінші символ адресі беріледі. Одан кейін puts сол символдың нөлдік символ емес екенін анықтап, ары қарай адреске бірді қосып, келесі символы оқиды, т.с.с. тіркес соңына дейін жетеді. Нөлдік символға жеткен соң, puts жұмысты аяқтайды;

Осындай тәсіл тіркес ұзындығына шек қоймай, нөлдік символға дейінгі символдарды біртіндеп оқуды жүзеге асырады.

Символға нұсқауышты пайдалану

Екінші тәсіл – символдарға нұсқауыш жасау. Программаны келесі түрге келтірейік:

```
#include <stdio.h>
#include <string.h>
main()
{
        char *msg;
        msg = "Сәлем, Азат!";
        puts(msg);
}
```

msg алдындағы жұлдызша (*) компиляторға оның символға нұсқауыш екенін білдіреді, яғни msg белгілі бір символ адресін сақтай алатын айнымалы. Бірақ мұнда компилятор символдар үшін ешқандай орын бөлмейді және msg да ешқандай мәнге ие болмайды.

Компилятор strcpy(msg, "Сәлем, Азат!") операторын кездестіргенде, ол тағы екі түрлі әрекет орындайды:

- объектілік код файлы ішіндегі бір орынға соңына (\0) символы қосылған "Сәлем, Азат!" тіркесін (ASCII коды 0) жазып қояды.
- сол тіркестің алғашқы символы адресін msg айнымалысына меншіктейді.

strcpy функциясын орындалып, puts(msg) командасы бұрынғыша нөлдік символға дейінгі мәліметті көшіреді.

Енді символдық тіркестерден тұратын жиымдарды қарастыралық. Бұл жиымдардың әрбір жолы символдық жиым болып табылады. Мысалы, статикалық жиымның сипатталуы келесідей түрде жазылуы мүмкін:

```
static char
    *m[4]={"регистр","жады","курсор","элемент"};
```

бұл жиым символдық тіркестерге сілтейтін 4 нұсқауыш болып табылады. Сонымен, символдық тіркестер жиымдар болып табылатын болса, онда осы жиымдарға сілтейтін 4 нұсқауыш қарастырылады. 1-жолға сілтейтін 1-нұсқауыш болып **m[0]** есептеледі, **m[1]–** 2-жолға сілтейтін 2-нұсқауыш болып табылады. Сонымен, әрбір нұсқауыш соған сәйкес жолдың немесе қатардың ең бірінші символына сілтейді.

```
*m[0]=='р'; *m[1]=='ж'; *m[2]=='к'; *m[3]=='э';
```

Тіркестерден құрылған жиымдарды сипаттағанда, символдық тіркестер көлемін көрсетуге де болады және бұл сипаттауда тіркестердің ұзындығын келесідей түрде көрсетуге болады:

```
static char m[10];
```

Символдар тіркестерін енгізу/шығару үшін **printf()** және **scanf()** функцияларының %s форматы қолданылады.

Келесі мысалда сөз тіркесінің ұзындығы екі тәсілмен анықталады.

```
#include <conio.h>
#include <stdio.h>
#include <string.h>
void main()
{
 char st[80];
 int i;
 clrscr();
```

```
puts("Сөз тіркесін енгізіп, Enter басыңыз:");
gets(st);
i=0;
while (st[i++])
     ;
printf("Енгізілген тіркес ұзындығы: %i\n",i-1);

puts("Сөз тіркесін енгізіп, Enter басыңыз:");
gets(st);
printf("Енгізілген тіркес ұзындығы: %i\n",
      strlen(st));
getch();
}
```

Ендігі мысалда енгізілген сөздің палиндром (алды-артынан оқығанда, мәні бірдей) екенін анықтайық.

```
#include <conio.h>
#include <stdio.h>
#include <string.h>
void main()
{
 char str[80];
 int k,s=0;
 clrscr();
 puts("Бір сөз (палиндром) енгізіңіз:");
 gets(str);
 k=strlen(str);
 for (int i=0; i<k/2; i++)
   if (str[i] == str[k-i-1])  s+=1;
 if (s==k/2)
 puts("Сөз - палиндром.");
 else puts("Сөз - палиндром емес.");
 getch();
 }
```

Сөз тіркестерін енгізу функциялары scanf(), gets(str).

Scanf() функциясы тіркестік айнымалылар мәнін %s форматымен енгізеді, бірақ ол тіркесті тек бірінші босорын таңбасына дейін ғана енгізе алады.

Ал **gets(str)** функциясы арасында босорыны бар тіркестерді енгізеді, енгізу ENTER пернесімен аяқталады.

Екі функция да тіркес соңына нөлдік байт қосып жазады. Оның үстіне тіркес –символдық жиым болып, ал жиым аты – оның компьютер жадындағы алғашқы адресіне сілтеме болғандықтан, тіркестік айнымалы атының алдына «&» символы жазылмайды.

Сөз тіркестерін шығару функциялары cprintf(),puts(),cputs()

printf() – экранға формат арқылы сөз тіркесін шығарады;

cprintf() – экранға **printf()** сияқты формат арқылы сөз тіркесін шығарады, тек олардың түстерін **textcolor()** және **textbackground()** функциялары арқылы өзгертуге мүмкіндік береді;

puts(str) – экранға сөз тіркесін шығарып, курсорды бірден келесі жолдың басына алып барады, мұндағы str – тіркестік константа немесе тіркестік айнымалы. Бұлар stdio.h тақырып файлы бойынша жұмыс істейді.

Екі функция да символдық жиымды нөлдік байтқа дейін шығарады. printf() функциясы символ тіркесі шығарылған соң, курсорды келесі жолға көшірмейді, ол үшін арнайы формат (\n) жазылуы тиіс. Ал puts( ) функциясы символдар шығарылған соң, автоматты түрде курсорды келесі жол басына көшіреді.

```
//puts функциясын пайдалану мысалы
#include <stdio.h>
#include <conio.h>
main()
{ char str1[]= “abc”;
  char str2[]= “def\nghi\n”;
```

```
char str3[]= “jkl”;
  puts(str1);
  puts(str2);
  puts(str3);
}
```

Нәтижесі:

```
abc
def
ghi
jkl
```

cputs(str) – экранға сөз тіркестерін шығарып, олардың түстерін **textcolor()** және **textbackground()** функциялары арқылы өзгертуге мүмкіндік береді, **conio.h** тақырып файлы бойынша жұмыс істейді.

Сөз тіркестерімен орындалатын басқа операциялар да стандартты функциялар арқылы атқарылады. Ол функциялар жұмыс істеуі үшін **string.h** тақырыптық файлы қажет.

Жалпы сөз тіркестеріне қолдану үшін **stdlib.h** немесе **string.h** тақырыптық файлдары қолданылады.

Сөз тіркестерімен жұмыс істейтін функциялар

1) **strlen(str)** функциясы str сөз тіркесіндегі символдар санын (соңғы нөлді есепке алмайды), яғни жолдың ұзындығын анықтайды, оның типі int, тақырыптық файлы <string.h>.

Мысалы. Бірнеше сөз тіркестерінің ұзындығын анықтайтын программа құру керек.

```
// strlen(str) функциясын пайдалану
#include <conio.h>
#include <stdio.h>
#include <string.h>
main ()
{
static char t[]=”Студенттер жайлы хабарлама.”;
clrscr();
printf(“%d\n”,strlen(t));
printf(“%d\n”,strlen(”Студенттер жайлы
                хабарлама.”));
printf(”%d\n”,strlen(”аль-Фараби ат.КазҰУ”));
printf(“%d\n”,strlen(“”));
getch();
}
```

2) strcat(stroka1,stroka2) функциясы қатарларды біріктіру үшін қолданылады. Ол **stroka1** және **stroka2** тіркестерін біріктіріп, нәтижені **stroka1** айнымалысына меншіктейді, **stroka2** тіркесінің мәні өзгермейді

Мысалы:

```
// strcat(str1, str2) функциясын пайдалану
#include <conio.h>
#include <stdio.h>
#include <string.h>
main ()
{
char str1[50]="Си тілін оқимыз, ";
char str2[]="жақында емтихан тапсырамыз.";
clrscr();
printf("%s\n",strcat(str1,str2));
puts(str1);   // қатарды экранға шығару
puts(strcat("Егер жақсы оқысақ, ",str2));
getch();
}
```

3) **strcmp(stroka1,stroka2)** функциясы екі сөз тіркесін салыстыру үшін қолданылады. Егер олар бірдей болса, функцияның мәні 0-ге тең болады, әйтпесе ол екі тіркестің айырмасын береді. Егер

stroka1<stroka2 болса, нәтиже 0-ден кіші, ал **stroka1> stroka2** болса, нәтиже 0-ден артық болады. Көбінесе бұл тәсіл екі тіркестің бірдей еместігін анықтау үшін ғана қолданылады.

Мысалы:

```
main ()
{
  printf("%d\n",strcmp("Сәлем","Сәлем"));
  printf("%d\n",strcmp("Azat","Izat"));
  printf("%d\n",strcmp("Абайда","Абайла"));
 getch();
}
```

Мұның нәтижесі:

0
-8
-7

Алғашқы екі сөз бірдей, нәтижесі – 0, келесі екі сөздің алғашқы әрпі әр түрлі, олардың ASCII-кодтарының айырмасы – -8 (А - 65, I - 73), ал 3-жолы -7 (д – 164, л – 171, олардың кодтарының айырмасы 164-171=-7).

```
// strcmp(str1, str2) функциясын пайдалану
#include <conio.h>
#include <stdio.h>
#include <string.h>
#define NAME "Ритчи"
main ()
{
 char f[20];
 puts("Си тілінің авторы кім:");
 gets(f);
 while(strcmp(f,NAME)!=0)
  {puts("басқа кім болуы мүмкін:");
   gets(f);
  }
puts("Жауап дұрыс!");
getch();
}
```

Нәтижесі:

Си тілінің авторы кім:
Керниган
басқа кім болуы мүмкін:
Ритчи
Жауап дұрыс!

4)strcpy(str1,str2) функциясы сөз тіркесінің көшірмесін алу үшін қолданылады, мұнда str2 айнымалысындағы сөз тіркесі str1 айнымалысына көшіріледі. Мысалы:

```
// strcpy(str1, str2) функциясын пайдалану
#include <conio.h>
#include <stdio.h>
#include <string.h>
main ()
{
 char str1[21];
 strcpy(str1,"Хал қалай, Азат?");
 puts(str1);
 strcpy(str1,"Тамаша!");
 puts(str1);
 getch();
}
```

Нәтижесі:

Хал қалай, Азат?
Тамаша!
2-мысал:

```
// strcpy(str1, str2) функциясын пайдалану
```

```
#include <conio.h>
#include <stdio.h>
#include <string.h>
#define stroka "көшіру функциясы"
main ()
{
 char *ptr=stroka;
 char res[25];
 clrscr();
 puts(ptr);
 puts(res);
 strcpy(res,ptr);
 puts(ptr);
 puts(res);
 getch();
}
```

Нәтижесі:

көшіру функциясы

көшіру функциясы

көшіру функциясы

Мұнда **ptr** айнымалысы **көшіру функциясы** сөзін береді, **res** айнымалысы бос жол береді, ал келесі жолы екеуі де **көшіру функциясы** сөзін береді.

5) **strstr(str1,str2)** функциясы 2-ші көрсетілген жолды 1-ші жолдың ішінен іздейді.

6) **strset(str,ch)** функциясы берілген қатардағы барлық символдарды көрсетілген символға (char ch) ауыстырады.

7) **strtod(str1,str2)** функциясы берілген қатарды **double** типті санға ауыстырады.

8) **strchr(str,c)** функциясы берілген қатардағы коды көрсетілген символдың позициясын анықтайды.

9) **strrev(str)** функциясы берілген қатардың барлық символдарын керісінше бейнелейді.

10) **strpbrk(str1,str2)** функциясы 2-ші қатардың кез келген символын 1-ші қатардан іздейді.

Бақылау сұрақтары

1. Тіркестік айнымалылардың сипатталу тәсілдері қандай?
2. Тіркестік айнымалы қандай идентификатормен (атаумен) және қалай анықталады?
3. Бір тіркестік айнымалыға немесе тұрақтыға қанша символ жазуға болады?
4. Тіркестік айнымалының ұзындығы қалай анықталады?
5. Тіркестік өрнектер дегеніміз не?
6. Тіркестік айнымалылар мен тұрақтыларға қандай амалдар қолданылады?
7. Тіркестің ішкі символдарын қалай бөліп алуға болады?
8. Си тілінде сөз тіркестерін өңдейтін қандай функциялар бар? Оларды қалай пайдаланады және олар қалай жазылады?

БӨЖ тақырыбы: «Жолдар және жолдардың түрлері»

Әдебиеттер:

1. Программирование на языке С++: Практический курс. Учебное пособие. Огнева М.В., Кудрина Е.В. 2018
2. Программалау. С++ тілін пайдалану қағидалары мен тәжірибесі. 2-бөлім: оқулық/ Б. Страуструп; Ауд.: Б.Бөрібаев, С. Адилғажинова.- Алматы, 2014.- 764 б.
3. Программалау, Астана: Фолиант, М.Мұқашева, 2013
4. Программирование на С++ : учебное пособие / Э.К. Нуркеева, А.С. Бижанова.- Алматы: КазАТК, 2012.- 82с.
5. Язык программирования С++. Лекции и упражнения. Прата, Стивен М.: Вильямс 2015, стр 445
6. С/С++ жоғары деңгейлі тілде программалау: оқулық, Павловская, Т.А.,Алматы: Дәуір, 2012. - 504б.

7. Программирование на C++: Учеб.пособие/ В.П.Аверкин, А.И.Бобровский и др. Под ред.проф. Хомоненко А.Д.. - СПб.: КОРОНА принт; М:Альтекс-А, 2003.- 508 с.