سایت مرجع دانشجوی پیام نور

- ✓ نمونه سوالات پیام نور : بیش از ۱۱۰ هزار نمونه سوال همراه با پاسخنامه تستی و تشریحی
- ✓ کتاب ، جزوه و خلاصه دروس
- ✓ برنامه امتحانات
- ✓ منابع و لیست دروس هر ترم
- ✓ دانلود کاملا رایگان بیش از ۱۴۰ هزار فایل مختص دانشجویان پیام نور

دانشگاه پیام نور- گروه کامپیوتر

اصول کامپیوتر ۲

رشته علوم رایانه

براساس کتاب اصول کامپیوتر ۲

تالیف دکتر داود کریم زادگان مقدم

تهیه کننده: مهدی یوسفخانی

تابستان ۸۵

فصل اول

مفاهیم اولیه

الگوریتم:

الگوریتم مجموعه محدود و پایانپذیر از دستورالعملها است.

شرایط الگوریتم:

- ورودی
- خروجی
- قطعیت
- محدودیت
- کارایی

۱- ورودی:

یک الگوریتم می تواند هیچ یا چندین کمیت ورودی داشته باشد که از محیط خارج تامین می شود.

۲- خروجی:

الگوریتم بایستی حداقل یک کمیت بعنوان خروجی ایجاد کند.

۳- قطعیت:

هر دستورالعمل باید واضح و بدون ابهام باشد.

۴- محدودیت:

اگر ما دستورالعملهای یک الگوریتم را دنبال کنیم برای تمام حالات باید پس از طی مراحل محدودی الگوریتم خاتمه یابد.

۵- کارایی:

تنها قطعیت کافی نیست بلکه هر دستورالعمل نیز باید انجام پذیر باشد.

پایان پذیری تفاوت میان یک برنامه و یک الگوریتم است.

پیچیدگی فضای لازم

میزان حافظه یا پیچیدگی فضای یک برنامه
مقدار حافظه مورد نیاز برای اجرای کامل
یک برنامه است.

فضای مورد نیاز یک برنامه:

- نیازمندیهای فضای ثابت
- نیازمندیهای فضای متغیر

نیازمندیهای فضای ثابت:

این مطلب به فضای مورد نیازی که به تعداد و اندازه ورودی و خروجی بستگی ندارد اشاره دارد.

نیازمندیهای فضای متغیر:

این مورد شامل فضای مورد نیاز متغیرهای ساخت یافته استکه اندازه آن بستگی به نمونه ای از مساله ای که حل می شود دارد.

پیچیدگی زمانی:

میزان یا پیچیدگی زمانی یک برنامه مقدار زمانی است که کامپیوتر برای اجرای کامل برنامه لازم دارد.

چرخه زندگی یک سیستم:

- نیازمندیها
- تحلیل
- طراحی
- کدنویسی
- بازبینی

جنبه های مهم بازبینی:

- اثبات درستی
- آزمایش درستی
- اشکال زدایی

بعضی از الگوهای توسعه نرم افزاری:

- روش آبشاری
- برنامه نویسی اکتشافی

۱- روش آبشاری:

این روش فرایند نرم افزار را متشکل از چند مرحله می داند. پس از تعریف هر مرحله توسعه نرم افزار به سمت مرحله بعدی پیش می رود.

۲- برنامه نویسی اکتشافی:

در این روش در حداقل زمان ممکن سیستمی ایجاد می شود و سپس اصلاحات لازم در آن بوجود می آید تا به درستی عمل کند.

طراحی شی گرا:

طراحی شی گرا بر مبنای پنهان سازی اطلاعات و نهادها است که دارای یک حالت اختصاصی و اعمالی بر روی آن حالت است.

ویژگیهای طراحی شی گرا:

۱- ناحیه مشترک داده ها حذف می شود.

۲- اشیا نهادهای مستقلی هستند که قابل تغییر می باشند.

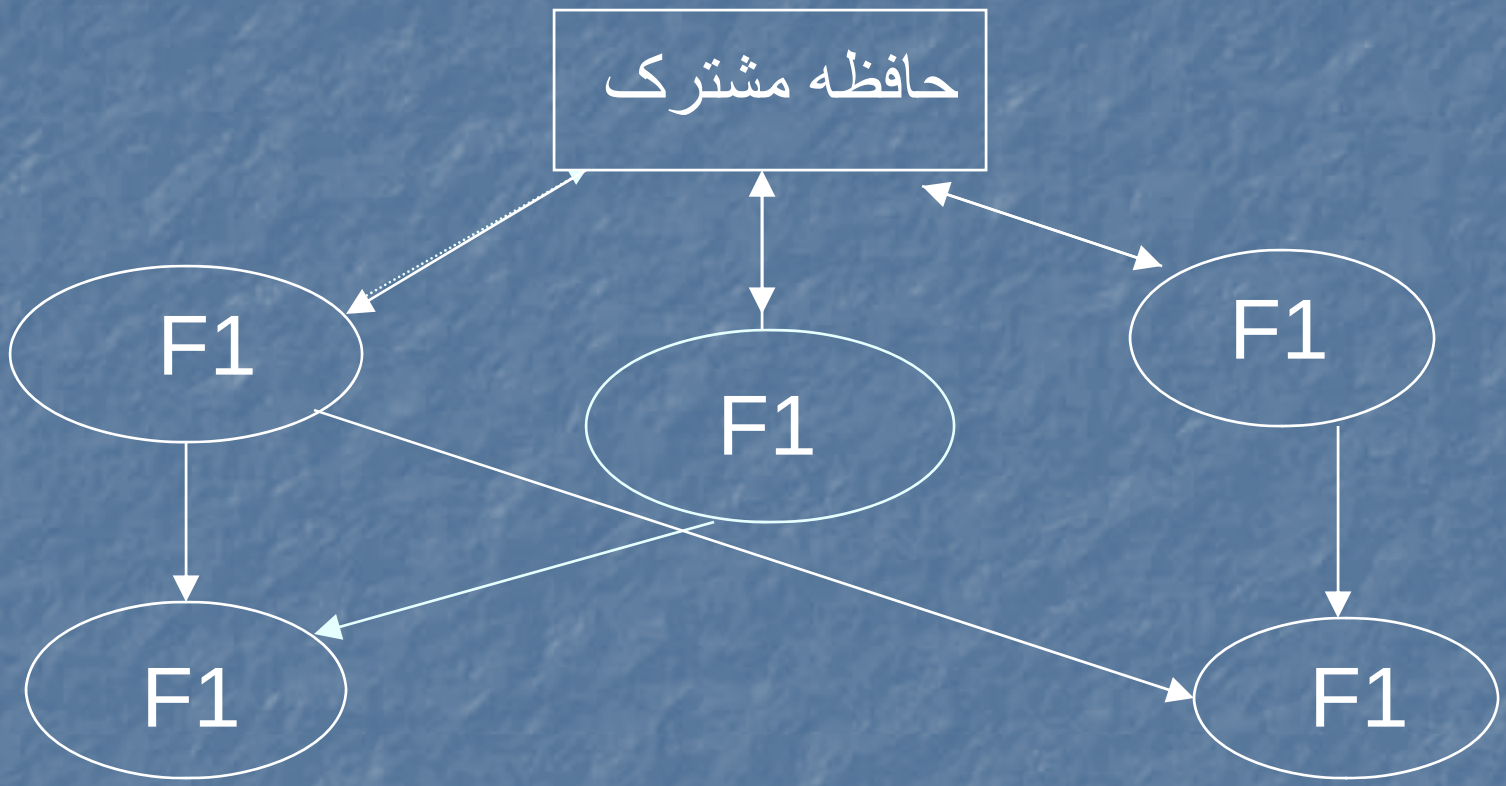
۳- اشیا ممکن است توزیع شده باشند و بصورت موازی یا ترتیبی اجرا شوند.

طراحی تابعی:

روشی برای طراحی نرم افزار است که در آن طراحی به مجموعه ای از واحدهای متاثر به هم تجزیه می شود که هر کدام وظیفه خاصی دارند.

استراتژی طراحی تابعی بر تجزیه سیستم به مجموعه
ای از توابع متاثر به هم تکیه دارد که حالت
متمرکز سیستم بین توابع مشترک است.

*



دید تابعی به طراحی

فصل دوم

زبان برنامه نویسی C

ویژگیهای زبان c:

- در دسترس است.
- ساخت یافته سطح بالا و انعطاف پذیر است.
- پیمانه ای است.

انواع کاراکتر در c:

- حروف بزرگ و کوچک
- ارقام دهدهی
- جای خالی
- کاراکترهای مخصوص
- کاراکترهای فرمت دادن

- حروف:

زبان c برخلاف پاسکال بین حروف بزرگ و کوچک فرق می گذارد.

- ارقام دهدهی:

شامل ۰ تا ۹

- کاراکترهای مخصوص:

& , . ; : ^ % $ # @ ! + - = / \
* ? ( ) < > { } [ ]

کاراکترهای فرمت دادن:

- کاراکتر Horizontal Tab یا \t
- کاراکتر Vertical Tab یا \v
- کاراکتر خط جدید یا \n

- کاراکتر برگشت به عقب یا \b
- کاراکتر تغذیه فرم یا \f
- کاراکتر ابتدای سطر یا \r
- کاراکتر تهی یا \o

شناسه:

یک شناسه c دنباله ای از حروف ارقام یا علامت زیر خط که با هر ترتیبی میتوانند قرار گیرند اما اولین کاراکتر باید یک حرف باشد.

متغیر:

متغیرها در زبان c شناسه هایی هستند که در میان یک بخش از برنامه برای نسبت دادن نوع تعیین شده ای از اطلاعات مورد استفاده قرار می گیرند.

بعضی از شناسه های زبان c کلمات رزرو شده هستند یعنی مفهوم آن از قبل در زبان تعریف شده و نمی توانند بعنوان شناسه تعریف شوند.

متداولترین کلمات کلیدی:

main	int	float	char	if	else	goto	for
do	double	while	default	signed	return	register	enum
static	continue	short	case	auto	struct	const	sizeof
break	long	switch	void	typedef	extern	unsigned	union

ساختار برنامه:

```
Main( )
{
  variables declaration;
  program statements;
}
```

تابع اصلی

شروع تابع اصلی

تعریف متغیرها

دستورات برنامه

پایان تابع اصلی

دستورالعملهای اجرایی:

در هر برنامه دستورالعملهای اجرایی باید بعد از تعریف متغیرها درج شوند. دستوری قابلیت اجرا دارد که در پایان آن (;) نوشته شود.

برنامه ای که مساحت مستطیلی به طول ۵ و عرض ۳ را محاسبه و چاپ کند:

```
#include<stdio.h>
Main()
{
  int length , width , S ;
  length = 5;
  width = 3;
  S = length*width;
  printf (“area =  %d”, S);
}
```

عبارت:

مجموعه ای معنی دار از داده ها است که با استفاده از نشانه ها یا عملگرهای محاسباتی، قیاسی و منطقی با یکدیگر ترکیب شده اند.

انواع عبارت:

- عبارت محاسباتی
- عبارت قیاسی
- عبارت منطقی

دستور:

دستور حکمی است که سبب می شود کامپیوتر عملی انجام دهد و به دو گروه تقسیم می شود:

- دستورهای ساده
- دستورهای ساخت یافته

دستور های ساده:

- دستور انتساب
- خواندن و نوشتن
- فراخوانی یک تابع
- انتقال کنترل به نقطه ای از برنامه

دستورهای ساخت یافته:

- دستور مرکب

```
{
  scanf(“%d%d”,&a,&b);
  s=a*b;
  p=2*(a+b);
  printf(“%d%d”,s,p)
}
```

- دستور حلقه تکرار

```
for(i=0;i<10;i++)
  sum=sum+i;
```

- دستور شرطی

```
if(a>b)
  c=a+b;
else
  c=a-b;
```

عملگرها:

نشانه هایی هستند که در عبارتها بکار می روند و به کمک آنها می توان اعمالی را روی داده ها انجام داد و عبارتند از:

- عملگرهای محاسباتی
- عملگرهای انتساب

- عملگرهای رابطه ای
- عملگرهای منطقی
- عملگرهای حافظه
- عملگرهای یکانی
- عملگر شرطی
- عملگر کاما

عملگرهای محاسباتی:

نام عملگر	نشانه	فرم
جمع	+	a+b
تفریق	-	a-b
منهای یکانی	-	-a
جمع یکانی	+	+a
ضرب	*	a*b
تقسیم	/	a/b
باقیمانده تقسیم	%	a%b
یک واحد افزایش	++	a++,++a
یک واحد کاهش	--	a--,--a

قواعد اولویت عملگرها و پرانتزها

w = x * y + w

تقدم عملگرهای محاسباتی:

()

++ --

- (یکانی)

*** / %**

+ -

عملگرهای محاسباتی انتساب:

عملگر	نام عملگر	دستورانتساب	معادل
+=	انتساب جمع	a+=b;	a=a=b;
-=	انتساب تفریق	a-=b;	a=a-b;
*=	انتساب ضرب	a*=b;	a=a*b;
/=	انتساب تقسیم	a/=b;	a=a/b;
%=	انتساب باقیمانده تقسیم	a%=b;	a=a%b;

عملگرهای رابطه ای:

نام عملگر	نشانه	فرم
بزرگتر از	>	a>b
کوچکتر از	<	a<b
مساوی یا بزرگتراز	>=	a>=b
مساوی یا کوچکتر از	=<	a<=b
مساوی	==	a==b
مخالف	!=	a!=b

فصل سوم

انواع داده ها

انواع داده:

- نوع صحیح
- نوع اعشاری
- نوع کاراکتر

- مقادیر ثابت
- مقادیر متغیر

انواع داده‌های اصلی

int

float

double

char

void

boolean ?!!

int

اعداد صحیح با دامنه محدود
برای کامپیوترهای شخصی دو بایت
-32767
+2762

float

اعداد حقیقی با دامنهٔ محدود

نمایش معمولی

نمایش علمی

12.3E- 4 = 12.00003

double

اعداد حقیقی با دقتی بیشتر از float

Char

کاراکترها نمادها یا حروف

‘a’

‘A’

‘+’

‘~’

بسته به محل استفاده عدد یا کاراکتر است.

void

دادهٔ تهی

دارای کاربردهای مختلف

مثال: توابع فاقد خروجی

انواع دادهای دیگر

با ترکیب کلمات زیر با برخی از انواع داده‌های اصلی:

signed ، **unsigned** (با علامت ، بدون علامت)

long ، **short**

مانند:

unsigned int

long int

unsinged long int

متغیرها

قوانین نامگذاری متغیرها:

- حروف ’a‘ تا ’z‘ ، ’A‘ تا ’Z‘ ، ارقام و ’_‘
- اولین کاراکتر رقم نباشد.
- کلمات کلیدی نمی‌توانند نام متغیر باشند.

متغیرها

اسامی مجاز:

count

c124

avg_grade

اسامی غیرمجاز:

1test

bin#tree

for

تعریف متغیر

نوع داده نام متغیر ;

```
int x ;
float m, n ;
char ch1, ch2, ch3 ;
long int count ;
```

مقدار دهی اولیه به متغیرها

```
int x = 5, y ;
char ch1 = ‘a’, ch2 = ‘A’, ch ;
```

ثابتها

تعریف ثابت:

#define نام ثابت مقدار ثابت

یا

const نوع داده نام ثابت = مقدار

مثال

```
#define   M    100
#define   P    3.14

const   int     n = 100 ;
const   char  c = ‘a’ ;
```

دستورات متداول پیش پردازنده:

- دستور #include
- دستور #define

مثال

```
#define   M    100
#define   P    3.14

const   int     n = 100 ;
const   char  c = ‘a’ ;
```

فصل چهارم

توابع ورودی و خروجی

توابع متداول ورودی و خروجی:

- scanf, printf
- getchar, putchar
- gets, puts

تابع خروجی printf

printf (“ عبارت 1 ” , عبارت 2)

عبارت 2 : اطلاعاتی که باید به خروجی منتقل شوند.
(اختیاری است)

printf (“ عبارت 1 ” , عبارت 2)

عبارت 1 می‌تواند شامل:

- اطلاعاتی که باید عینا در خروجی چاپ شوند
- کاراکترهای تعیین کنندهٔ فرمت خروجی
- کاراکترهای کنترلی

کاراکترهای تعیین کنندهٔ فرمت خروجی

- مشخص کنندهٔ نوع اطلاعات ذکر شده در عبارت 2
- با علامت % شروع می‌شوند. مانند:

%c (برای کاراکتر)

%d (برای عدد صحیح)

%f (برای عدد اعشاری)

کاراکترهای کنترلی

- تعیین شکل اطلاعات خروجی
- با علامت \ شروع می‌شوند. مانند:

\n انتقال به سطر جدید

\f انتقال به صفحهٔ جدید

مثال

```
printf (“this is a test.”);
```

خروجی

this is a test.

مثال

```
int i =10 ;
char ch = ‘a’ ;
printf (“%d , %c” , i , ch);
```

خروجی

10 , a

تابع ورودی scanf

scanf (“ عبارت 1 ” , عبارت 2)

عبارت 2 : آدرس متغیرهایی که باید خوانده شوند

عبارت 1 : نوع متغیرها و نحوهٔ خوانده شدن آنها

عبارت 1 شامل:

1. کاراکترهای فرمت. مشخص کنندهٔ نوع اطلاعات. مانند:

%c (کاراکتر)

%d (عدد صحیح)

2. کاراکتر فضای خالی

تاثیر: در نظر نگرفتن (رد کردن) فضای خالی در اطلاعات ورودی

3. کاراکترهای دیگر

تاثیر: خواندن و صرفنظر کردن از کاراکتر فوق

مثال

```
int i , j ;

char ch ;

scanf (“%d %d %c” , &i , &j ,
   &ch) ;
```

فصل پنجم

ساختارهای کنترلی و شرطی

دستورات و ساختارهای کنترلی:

این دستورات این امکان را فراهم می کنند که یک قطعه از برنامه چندین بار تا موقعی که شرط ویژه ای برقرار است اجرا گردد.

دستور ساختار حلقه تکرار:

- **for**
- **while**
- **do-while**

دستور شرطی:

- **if**
- **switch**

دستورات کنترلی:

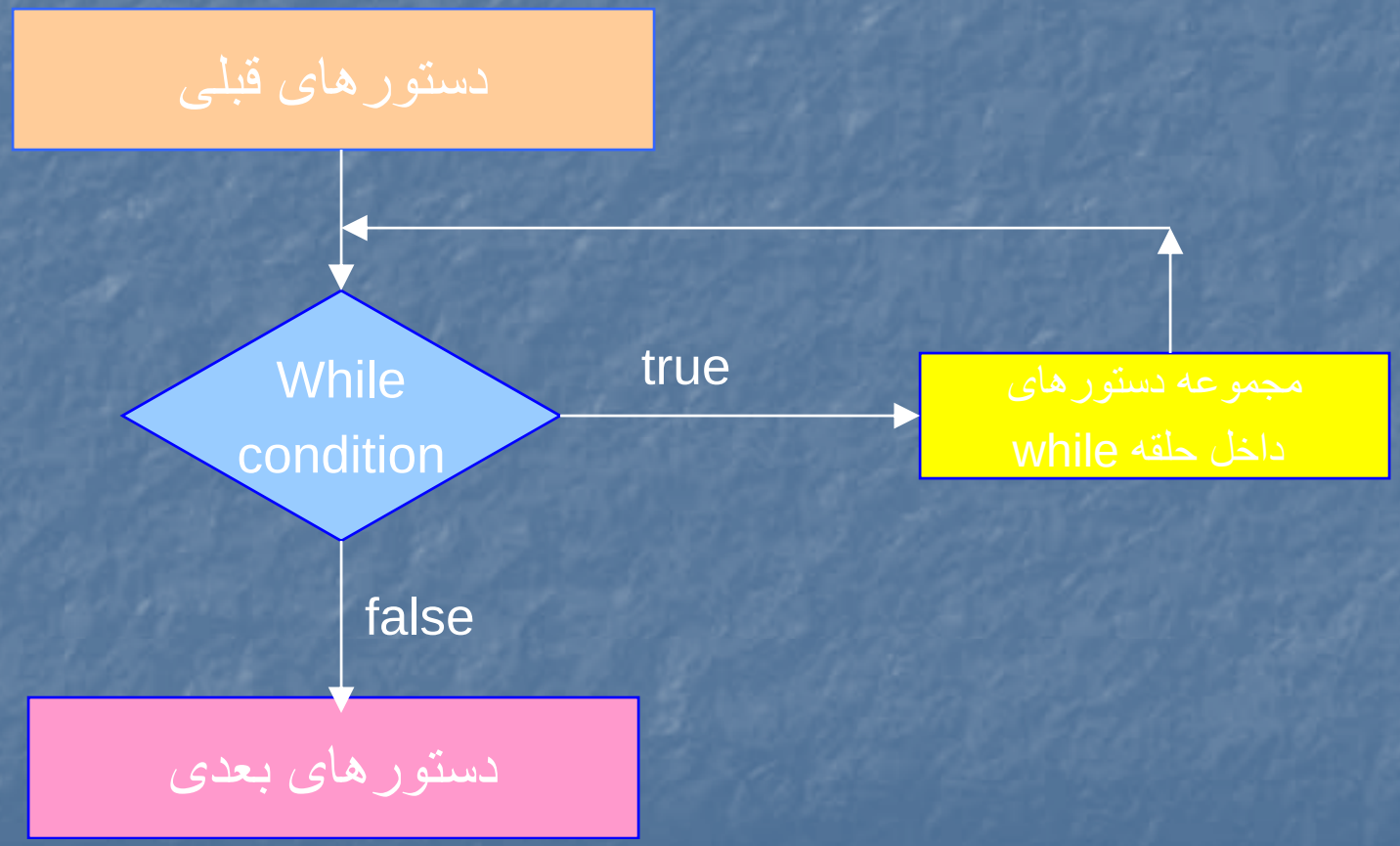
- goto
- continue
- break
- exit

دستور کنترلی while

این دستور برای انجام عملیات تکراری بکار می رود.
با استفاده از آن یک حلقه تا موقعی که شرط معینی برقرار باشد اجرا می گردد.

نمودار دستور while

*



مثال: برنامه ای که اعداد صحیح صفر تا ۱۰ را روی خطوط متوالی چاپ می کند:

```
#include<stdio.h>
main()
   {
     int number=0;
     while(number<=10)
     printf(%d\n”,number++);
   }
```

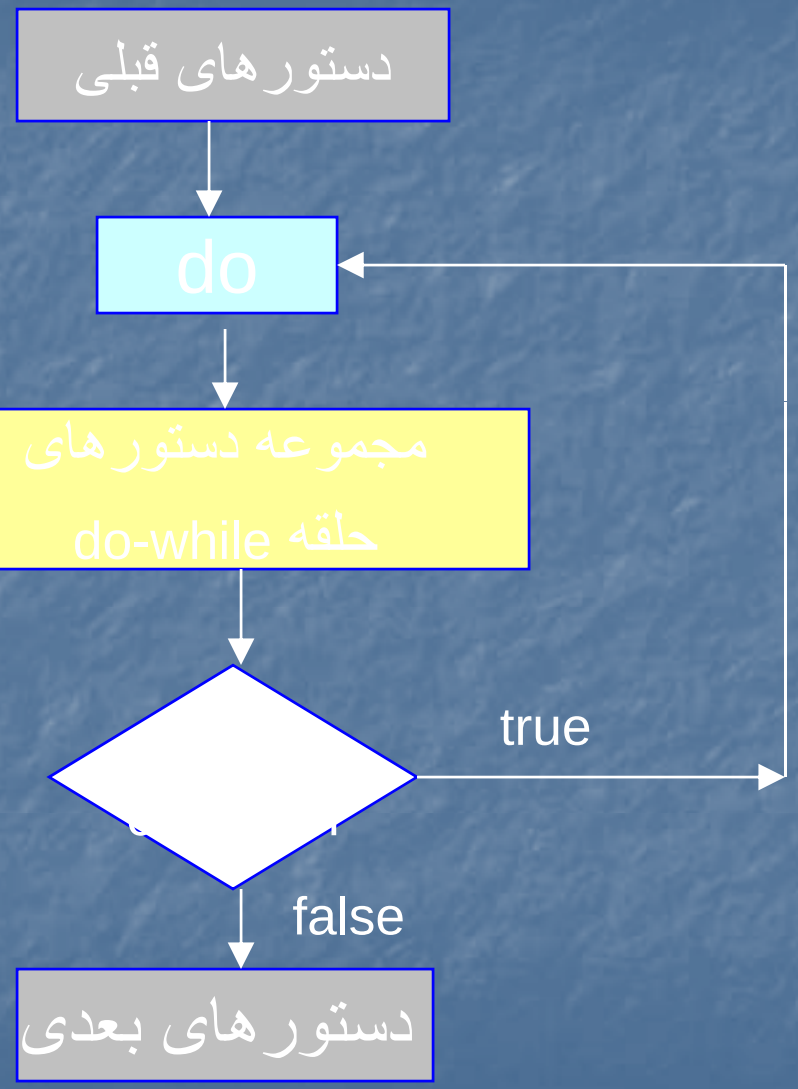
مثال:

```
int  sum = 0 , i = 0 ;
while  ( i != -1 )
{
      sum = sum + i ;
      scanf (“ %d” , &i) ;
}
```

دستور کنترلی do while:

در دستور while آزمایش شرط برای ادامه حلقه در آغاز هر تکرار حلقه انجام می شود اما با این دستور این آزمایش در پایان حلقه انجام می شود.

نمودار دستور do while



*

مثال: برنامه ای که اعداد صحیح صفر تا ۱۰ را روی خطوط متوالی چاپ می کند:

```
#include<stdio.h>
main()
 {
   int number=0;
   do
   printf("%d\n",number++);
  while(number<=10);
 }
```

مثال:

```
do {
    sum = sum + n ;
    scanf (“%d” , &n) ;
} while (n != -1)
```

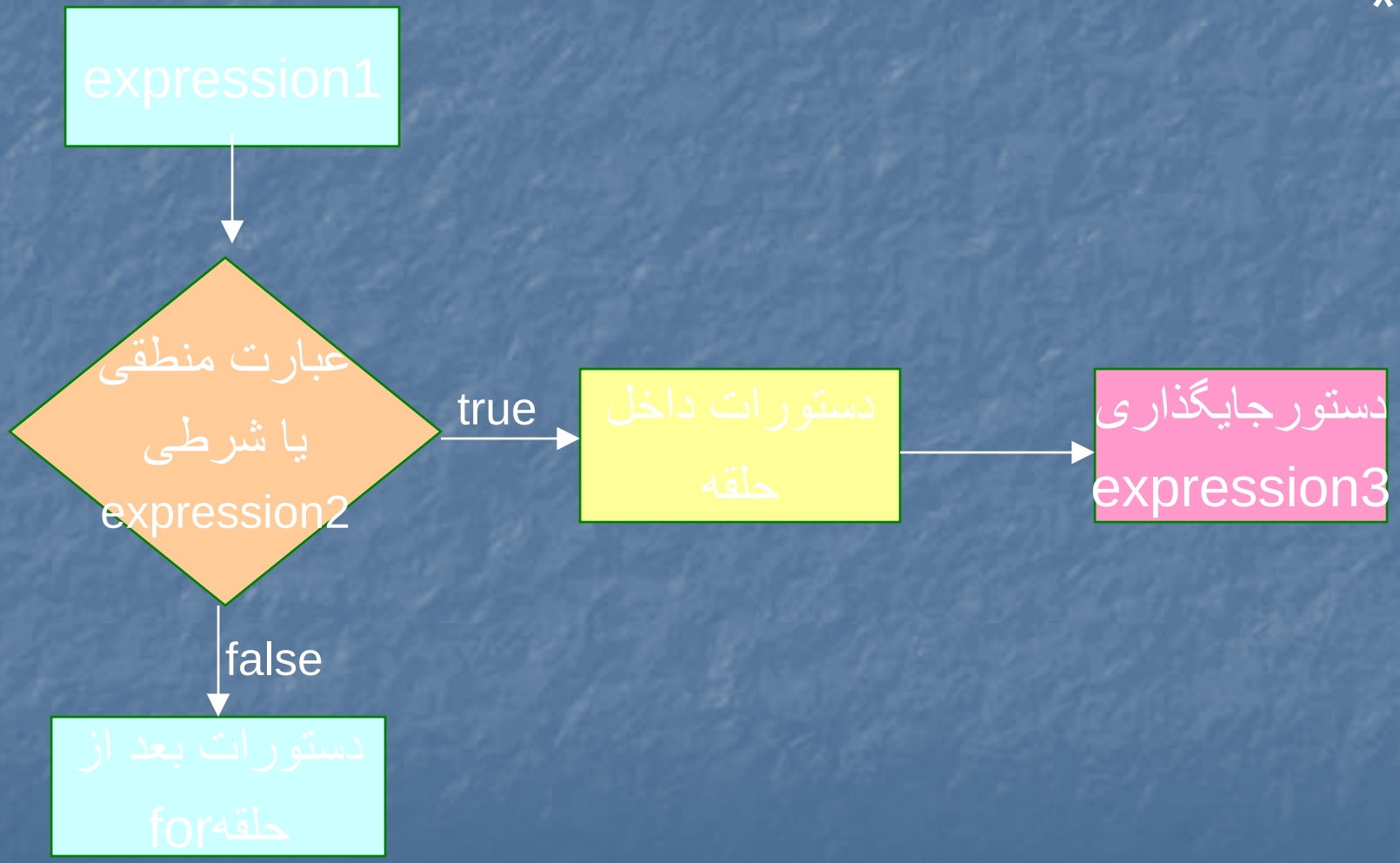
دستور کنترلی for:

این دستور شبیه دستور while بوده و دارای فرم کلی زیر است:

```
for(expression1; expression2; expression3)
   statement;
```

نمودار دستور for

*



مثال: برنامه ای که اعدادصحیح صفر تا ۱۰ را روی خطوط متوالی چاپ کند:

```
#include<stdio.h>
main()
 {
  int number;
  for ( number=0; number<=10;
 ++number)
    printf (“%d\n”, number);
 }
```

مثال:

```
for ( i = 0 , sum = 0 ; i < 20 ; i ++ ,
  j -- )
{
      sum = sum + i + j ;
      printf (“ %d” , sum) ;
}
```

مثال:

```
for  ( i = 10 , j = 0 ;
       (i > 0) && (j <=20) ;
       i -- , j + + )
            printf (“ %d , “ , i + j) ;
```

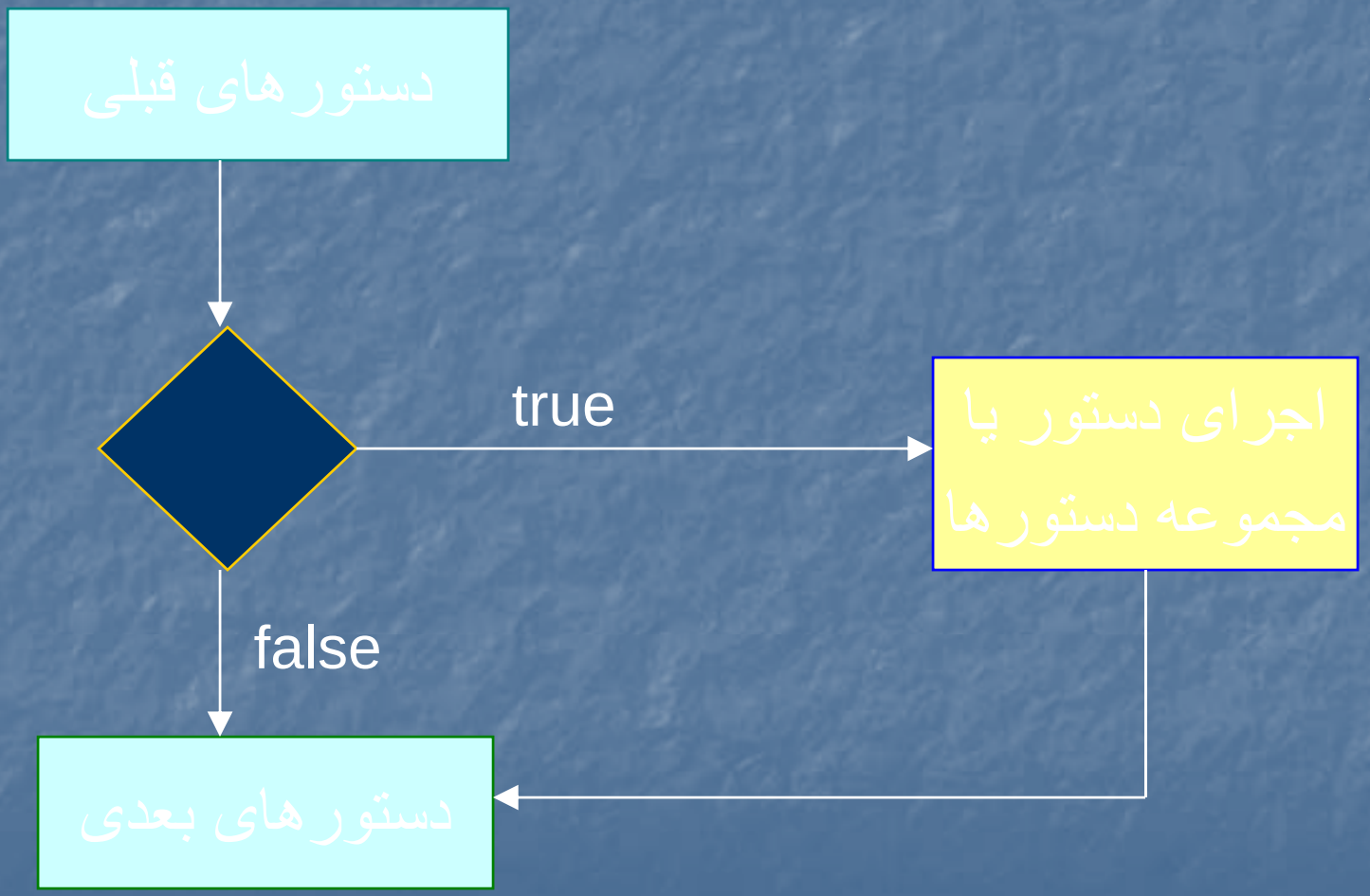
تبدیل حلقهٔ while به for (روش دیگر)

for (; while شرط حلقهٔ ; while بدنهٔ حلقهٔ)
;

دستورهای شرطی if و if else:

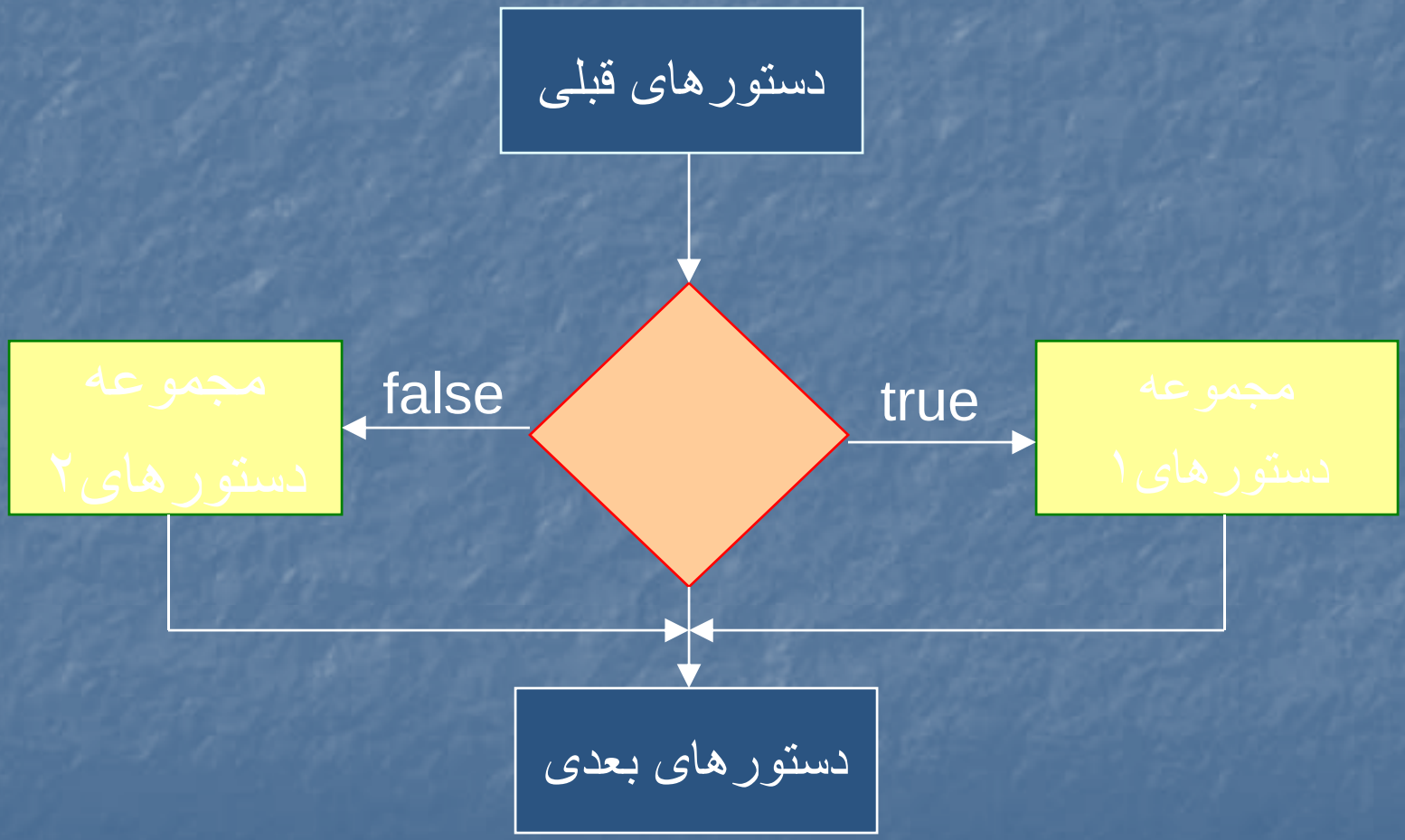
این دستورها موجب می شوند تا در صورت
وجود شرایطی یک مجموعه از دستورها
ودر صورت عدم وجود آن مجموعه دیگری
از دستورها اجرا شوند.

نمودار دستور if



*

نمودار دستور if else



*

مثال: برنامه ای که متنی را خوانده تعداد محلهای خالی ارقام، حروف ، پایان خط و جمع سایر نشانه های موجود در آن متن را شمرده چاپ کند:

```
#include<stdio.h>
main()
 {
   int c, blank_cnt, digit_cnt, letter_cnt,
  n1_cnt, other_cnt;
```

```
    blank_cnt=digit_cnt=letter_cnt=n1_c
    nt=other_cnt=0;
while((c=getchar())!=EOF)
if (c==´´)
   ++blank_cnt;
elfe if(´0´<=c&&c<=´9´)
    ++digit_cnt;
elfe if(´a´<=c&&c<=´z´||
   ´A´<=c&&c<=´Z´)
    ++letter_cnt;
```

```
else if(c==´\n´)
   ++n1_cnt;
else
    ++other_cnt;
printf("\n%12s%12s%12s%12s%12s%12
   s","blanks","digits","letters","lines","other
   s","totals");
printf("\n\n%12d%12d%12d%12d%12d
    %12d
          \n\n, blank_cnt, digit_cnt, letter_cnt,
   n1_cnt,
          other_cnt, blank_cnt+ digit_cnt+
   letter_cnt+ n1_cnt+ other_cnt);
}
```

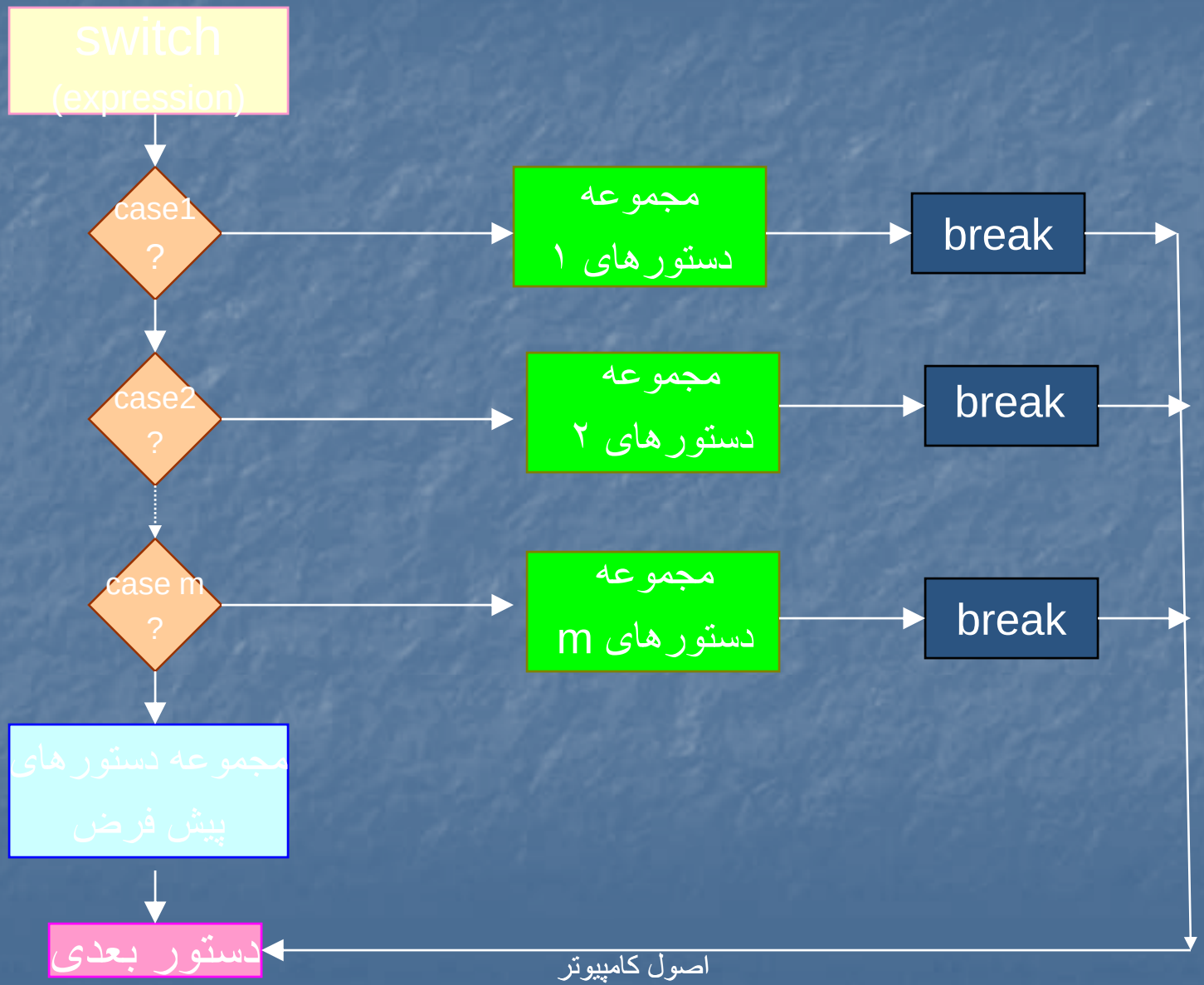
مثال:

```
if  (ch == '+')
        r = x + y ;
else  if (ch == '-')
              r = x - y ;
         else if (ch == '*')
                    r = x * y ;
```

دستور شرطی switch:

این ساختار یا دستور موجب می گردد که گروه مشخصی از دستورها بین چندین گروه از دستورها انتخاب گردد.

نمودار دستور switch



*

مثال: برنامه زیر متغیر کاراکتری color را از صفحه کلید دریافت کرده و رنگی که با آن حرف آغاز می شود را در خروجی چاپ می کند:

```
#include<stdio.h>
main ( )
{
   char  color ;
   color = getchar( ) ;
```

```
switch (color)
   {
      case ´Y´:   printf (“Yellow”);
                     break ;
      case ´B´:   printf (“Black”);
                     break ;
      case ´W´:  printf (“White”);
                     break ;
```

```
    case ´R´:  printf (“Red”);
                    break ;
    case ´G´:  printf (“Green”);
                    break ;
    default  :   printf(“error”);
   }
}
```

مثال:

```
char  ch;
switch  (ch)  {
  case ‘+’ :
          r = x + y ;
          break ;
```

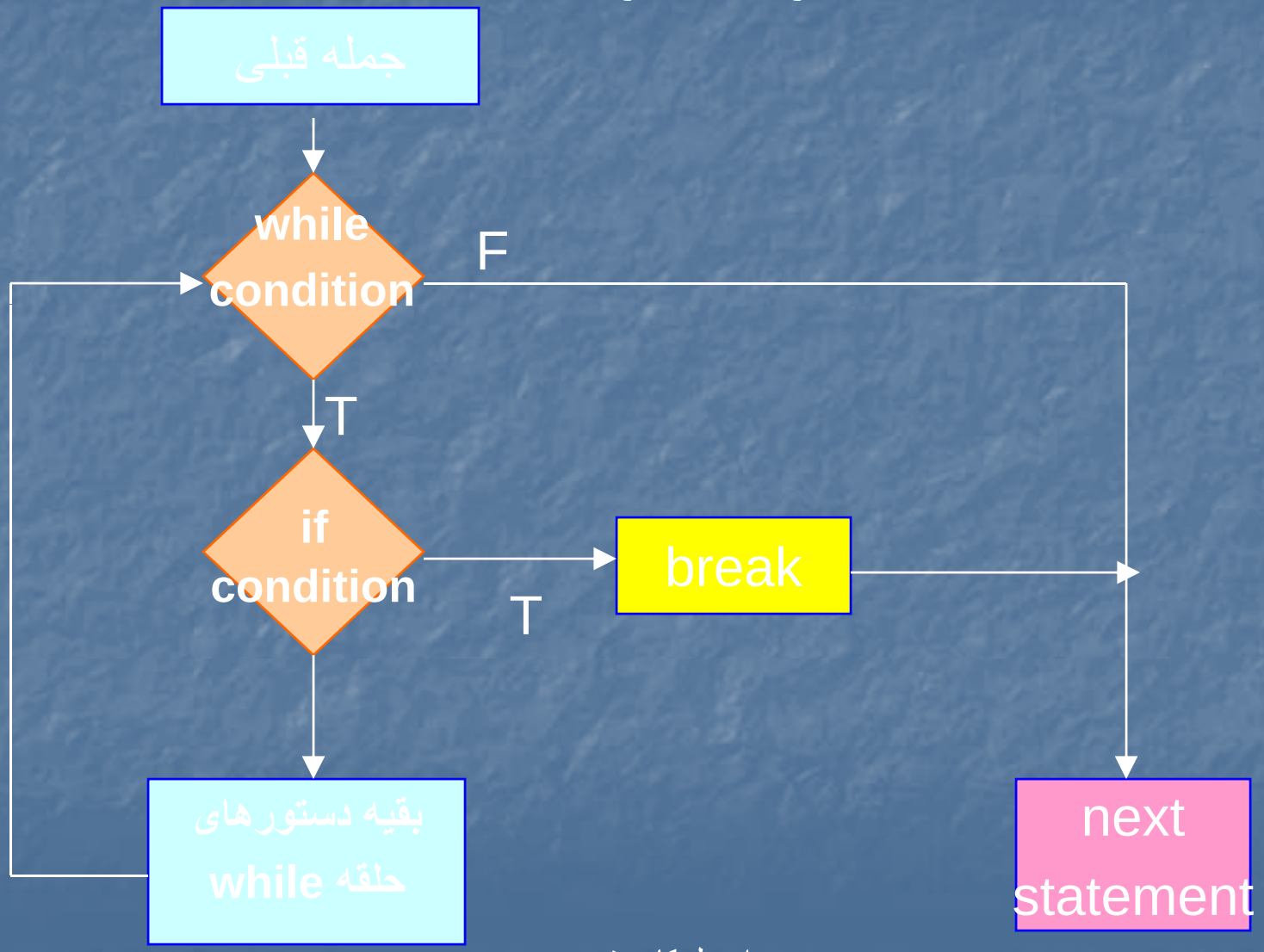
```
case '-' :
        r = x - y ;
        break ;
case '*' :
        r = x * y ;
        break ;
```

```
    case '/' :
            r = x / y ;
            break ;
    default :
            r = 0 ;
            printf ("Invalid operator.") ;
}
```

دستور break:

این دستور مسیر معمولی و ثابت کنترل را تغییر داده و یا قطع می کند. این دستور برای خارج شدن از یک دستور switch و یا پایان دادن حلقه ها استفاده می شود.

نمودار دستور break

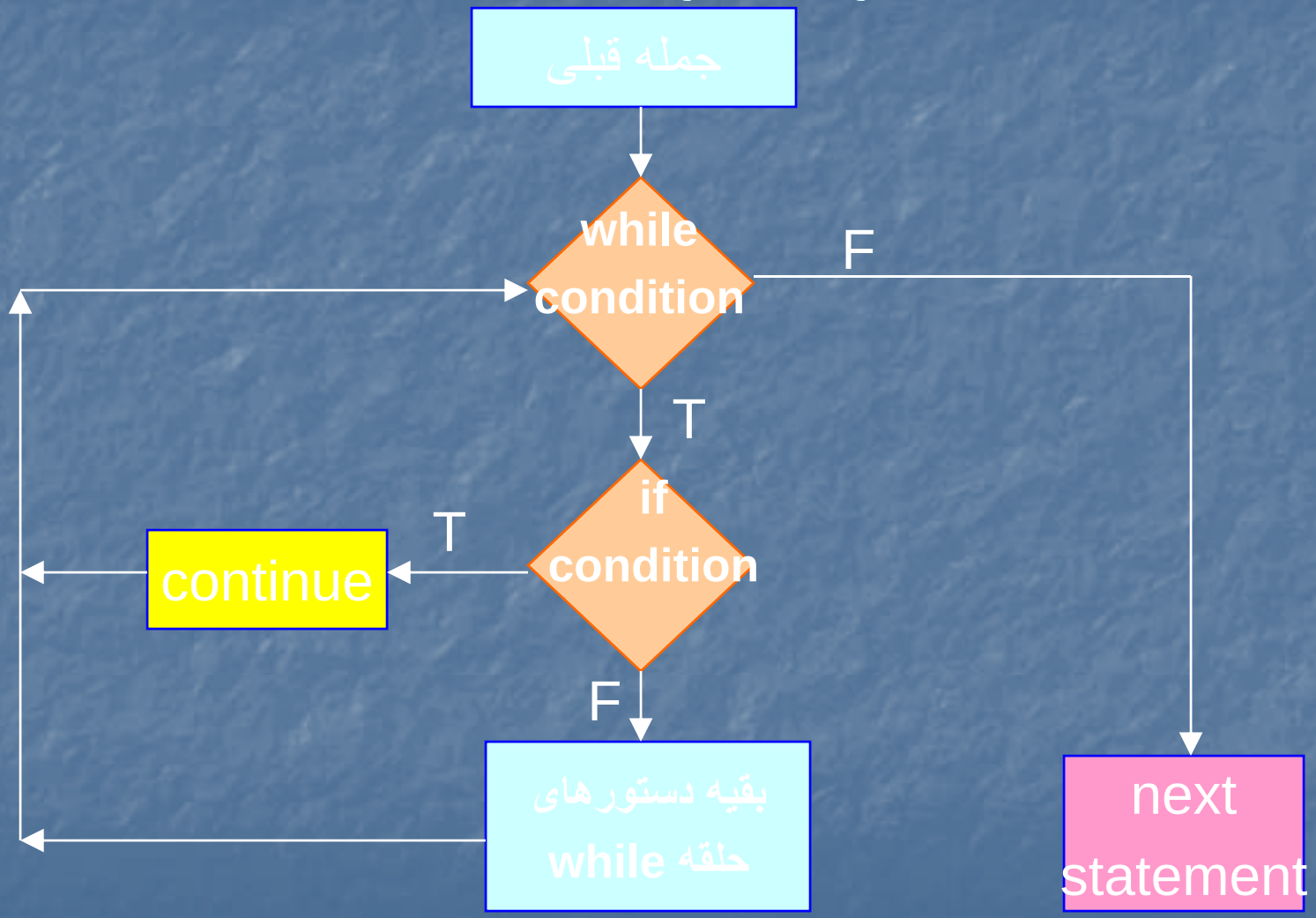


*

دستور continue:

این دستور به منظور یک عبور جانبی از کنار سایر مراحل جاری یک حلقه بکار برده می شود در واقع باقیمانده تکرار جاری یک حلقه نادیده گرفته شده و بلافاصله تکرار بعدی حلقه آغاز می شود.

نمودار دستور continue



*

مثال: برنامه زیر n عدد از ورودی خوانده و مجموع اعداد مثبت را محاسبه می کند:

```
#include<stdio.h>
main ( )
  {
   int  n,i,x,sum= 0 ;
   printf(“how many numbers?”);
   scanf(“%d”,&n);
```

```
for (i=1 ; i<n ; ++ i)
   {
     scanf("%%d",&x);
     if(x<=0)
       continue ;
     sum +=x ;
     }
  printf("\n sum = %d\n",sum);
 }
```

دستور goto:

این دستور ترتیب طبیعی برنامه را تغییر داده و کنترل را به قسمت دیگری از برنامه منتقل می کند و در واقع یک انتقال کنترل بدون شرط است.

مثال: برنامه زیر عددی صحیح را از ورودی خوانده در صورتی که زوج باشد پیغام مناسب نمایش می دهد در غیر این صورت از برنامه خارج می شود:

```
#include<stdio.h>
main ( )
    {
      int  n ;
      scanf(“%d”,&n) ;
```

```
    if(n%2 == 0)
      goto even ;
    else
          goto  odd;
      even :  printf("even  number.\n")
  ;
      exit(1) ;
      odd  :  exit (0);
}
```

دستور exit:

این دستور یک تابع است و اجرای برنامه را بطور کامل قطع کرده و موجب خاتمه پذیرفتن کل برنامه می گردد.

کاربرد متداول آن زمانی است که یک شرط خاص برای اجرای برنامه برقرار نباشد و اجرای برنامه قطع شود.

```
{
  ....
   if  x<2
       exit (0);
}
```

نکات:

- بخش **default** اختیاری است.
- مقادیر موجود در **case** ها نباید مساوی باشند.

- در switch فقط تساوی را می‌توان چک کرد.
- در صورت عدم استفاده از break دستورات case بعدی و تا آخر اجرا خواهد شد.

فصل ششم

برنامه سازی پیمانه ای

در زبانc زیربرنامه تابع نامیده میشود. تابع یک قطعه برنامه کامل است که کار معینی را انجام میدهد و موجب جلوگیری از برنامه نویسی تکراری در بین برنامه ها میشود.

توابع به دو گروه دسته بندی میشوند:

۱-توابع کتابخانه ای: توابعی که از پیش تعریف شده اند.

۲-توابع فرعی: توابعی که توسط برنامه نویس تعریف میشوند.

مزایای طراحی به صورت پیمانه ای:

- ساده شدن کنترل و خطا یابی
- امکان انجام تغییرات در برنامه و اصلاح آن
- امکان استفاده مجدد از تابع
- امکان همکاری برنامه نویسان متعدد در نوشتن یک برنامه

هر تابع شامل عناصر زیر است:

- عنوان تابع: نام و نوع و آرگومانهای تابع
- بدنه تابع:
  - اعلان متغیرهای محلی: متغیری که مخصوص خود تابع باشد.
  - سایر دستورهای تابع: شامل متن تابع و دستورهای اجرایی.

عناصر یک تابع:

(اسامی پارامترها) <نام تابع> <نوع تابع>
تعریف پارامترها
{
بدنه تابع
}

دستور return:

برای خروج از تابع استفاده میشود. اگر نیاز باشد تابع مقداری را برگرداند دستور return دارای آرگومان خواهد بود در غیر اینصورت نقش آن فقط خروج از تابع خواهد بود.

مثال: تابع زیر ماکزیمم دو عدد صحیح را بدست آورده در خروجی چاپ می کند:

```
Maximum(int  x , int y)
 {  int   z ;
    z = (x>=y)?x:y ;
    printf("%d" ,z) ;
    return:
  }
```

فراخوانی تابع:

در زبان c فراخوانی یک تابع با نوشتن نام تابع و پارامترهای آن به صورت یک تک دستور و یا در یک دستور جایگذاری انجام می شود.

محاسبه فاکتوریل عدد صحیح n:

```
#include<stdio.h>
Main()
  {
      int n;
      scanf(“%d”,&n);
      printf(“n=%d fact(n)=%d”, n,
   fact(n) );
   }
```

```
int  fact(int  n)
  {
        int f = 1 , i ;
        for(I =2 ; i<=n;++i)
            f=f * i;
         return(f);
   }
```

توابع به دو صورت از نظر نحوه انتقال آرگومانها فراخوانی میشوند:

۱- فراخوانی با مقدار یا فراخوانی توسط ارزش
۲- فراخوانی توسط آدرس یا فراخوانی توسط ارجاع

- فراخوانی با مقدار:

در این روش خود متغیرها به تابع فراخوانده شده انتقال نمی یابند بلکه مقدار آن انتقال می یابد و هر تغییری که توسط تابع فرعی روی آن مقدار انجام شود در تابع فراخواننده آن منعکس نمیشود.

مثال:

```
#include<stdio.h>
Main()
  {
     int n;
     long int factorial;
     printf(“n=“);
     scanf(“%d”,&n);
     factorial=fact(n);
     printf(“\n fact(n)=%d”,factorial);
   }
```

```
int fact (int n)
 {
  long int f=1;
  while(n>1)
     f*=n--;
  return(f);
  }
```

- فراخوانی توسط ارجاع:

در این روش آدرس آرگومان به درون تابع فراخوانده شده کپی میشود و انجام هرگونه تغییرات بر آنها بر آرگومانهای متناظر آن در تابع فراخواننده نیز همان اثر را خواهد داشت.

انتقال آرایه به تابع:

سه روش برای تعریف پارامتری که اشاره گر آرایه را دریافت می کند وجود دارد:

- روش اول: پارامتر مورد نظر به صورت آرایه تعریف می گردد.

```
#include<stdio.h>
void display(int num[10]);
main()
{
 int a[10] , i;
 for(i=0 ; i<10 ; ++i)
  a[i]=i;
 display(a);
}
void noor(num)
{  int i;
  for (i=0 ; i<10 ; ++i)
    printf ("\n%d", num[i]);
}
```

- روش دوم: آنرا بصورت آرایه ای معرفی می کنیم که اندازه آن مشخص نشده باشد.

```
void noor (num)
int num[];
{
   for (i=0 ; i<10 ; ++i)
     printf("\n%d" , num[i]);
}
```

- روش سوم: آنرا بعنوان یک اشاره گر int تعریف کنیم.

```
void noor (num)
int *num;
{
   int i ;
   for (i=0 ; i<10 ; ++i)
     printf("%d" , num[i]);
}
```

تابع بازگشتی:

بازگشتی بودن فرآیندی است که در آن یک تابع بطور مکرر خودش را فراخوانی می کند تا آنکه به شرط خاصی برسد.

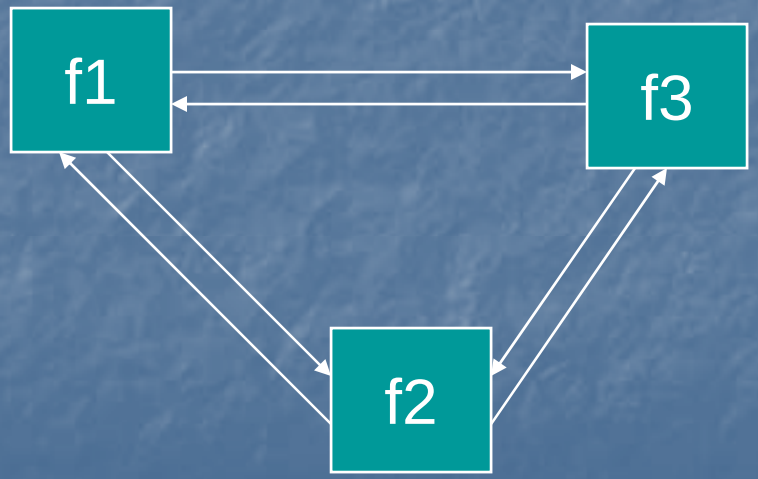
- یک تابع را بازگشتی نامند اگر بصورت مستقیم یا غیرمستقیم تابع خود را فراخوانی کند.

مثال: تابع زیر مجموع اعداد طبیعی 1 تا n را محاسبه می کند

```
int sum(int n)
{
   if(n<=1)
     return(n);
   else
     return (n+sum(n-1));
}
```

استفاده از چند تابع:

در زبان c می توان در هر برنامه به هر تعداد که نیاز باشد تابع تعریف کرد و هر تابع می تواند تابع دیگری را فراخوانی کند.



قلمرو متغیر ها:

متغیر ها برحسب اینکه در چه قسمتی از برنامه شناخته شده اند به دو دسته تقسیم میشوند:

۱- متغیر های محلی

۲- متغیر های عمومی

متغیر های محلی یا خصوصی:

- اینگونه متغیر ها فقط درون همان تابع شناخته شده اند.
- فقط در موقعی که اجرای برنامه وارد بلوک آنها میشود وجود دارند و بعد از آن از بین می روند.

```
func1()
{
  int x;
  scanf("%d",&x);
  if(x<=1)
   {
     char str[20];
     printf("Enter City: ");
     gets(str);
     puts(str);
   }
}
```

متغیرهای عمومی یا سراسری:

- در تمام طول برنامه شناخته شده اند و می توانند بوسیله هر قسمت از برنامه بکار روند.
- در تمامی وقت اجرای برنامه مقادیر خود را حفظ می کنند.

کلاسهای حافظه:

سطح ذخیره سازی اطلاعات و یا کلاس حافظه قلمرو متغیر و نیز زمان حیات آنرا در یک برنامه مشخص می کند.

در زبان c چهار کلاس حافظه وجود دارد:

- اتوماتیک (automatic)
- استاتیک (static)
- ثبات (register)
- خارجی (external)

کلاس حافظه اتوماتیک:

این متغیرها همیشه در درون یک تابع توصیف می شوند و نسبت به آن تابع متغیر محلی هستند. بنابراین متغیرهایی که در توابع گوناگون از این نوع حافظه تعریف می شوند از یکدیگر مستقل خواهند بود حتی اگر اسامی یکسان داشته باشند.

متغیرهای اتوماتیک با خروج از تابع مقادیر خود را از دست می دهند پس باید هنگام ورود مجدد به برنامه دوباره مقدار دهی شوند.

کلاس حافظه خارجی:

اگر متغیرهایی را در واحد اصلی تعریف کنیم و بخواهیم از آنها در واحد فرعی استفاده کنیم باید آنها را در این واحدها با کلمه کلیدی extern معرفی کنیم.

File1

```
int x , y;
char ch;
main( )
    { ....
      ....
    }
func1( )
    {
      x =123;
    }
```

File2

```
extern int x ,y;
extern char ch;
func2( )
        {
         x= y /10 ;
        }
func3( )
        {
         y =10 ;
        }
```

چون متغیر های خارجی بصورت عمومی تعریف میشوند قلمرو آنها از محل تعریفشان تا انتهای برنامه می باشد و در بقیه برنامه و همه توابع بعد از آن شناخته شده اند.

کلاس حافظه استاتیک:

در یک برنامه تک فایل هر کدام از متغیرهای استاتیک در همان فایلی که مربوط به آن هستند تعریف می گردد.

متغیر های استاتیک نسبت به تابع خود محلی هستند اما برخلاف متغیر های اتوماتیک در تمام طول برنامه مقدار قبلی خود را حفظ می کنند.

```
float  a,b,c;
main ( )
   {
     static float a ;
     void dummy(void) ;
   }
void site (void) ;
   {
      satatic int  a ;
      int  b ;
   }
```

می توان متغیرهای اتوماتیک و ایستا را همنام با متغیرهای خارجی تعریف کرد در این موارد هرگونه تغییر روی این متغیرها هیچ نقشی در مورد متغیرهای عمومی همنام با آنها نخواهد داشت.

هنگام توصیف متغیرهای محلی استاتیک می توان آنها را مقداردهی اولیه کرد اما این مقدار باید بشکل ثابتها بیان گردد نه عبارات.

کلاس حافظه ثبات:

این نوع کلاس حافظه فقط در مورد متغیرهایی از نوع int و char قابل اعمال است.

استفاده از این کلاس حافظه سرعت هرنوع عملیات روی این متغیرها را افزایش می دهد.

```
int  noor( m,n )
int  m ;
register int  n ;
  {
     register int  k ;
     k =1 ;
     for (; n>0 ; n--)
        k = k + m ;
     return  k ;
  }
```

ماکرو:

یک ماکرو شناسه ای است که معادل با یک عبارت یک دستور و یا گروهی از دستورات تعریف شده باشد.

در صورتی که ماکرو شامل گروهی از دستورات باشد مشابه تابع عمل می کند.

```
#include<stdio.h>
#define  area    length*width
main( )
{
  int  length , width ;
  scanf(“%d%d” , length,width) ;
  printf(“area =  %d” , area) ;
}
```

- تعریف ماکروها در ابتدای برنامه و پیش از تعریف اولین تابع است.

- حوزه تعریف ماکرو از محل تعریف شدن تا انتهای آن فایل است.

- ماکرویی که در یک فایل تعریف شده در فایل دیگری قابل شناسایی نیست.

یک ماکرو برخلاف یک تابع:

- سرعت اجرای برنامه را افزایش و در عوض اندازه برنامه اجرایی را بزرگ می کنند.
- شناسه یک ماکرو آدرس پذیر نیست.
- یک ماکرو نمی تواند بصورت بازگشتی خود را فراخوانی کند.

فصل هفتم

آرایه

آرایه:

- مجموعه عناصری است دارای ویژگیها و صفات یکسان هستند.
- یک فضای پیوسته ازحافظه اصلی کامپیوتر که می تواند چندین مقدار را در خود جای دهد.

تعریف آرایه:

در زبان c آرایه ها بصورت متغیرهای معمولی تعریف می شوند با این تفاوت که نام آرایه باید با یک مشخصه اندازه همراه باشد:

```
type array-name[array-size];
```

- آرایه ها از نظر نوع اختصاص حافظه فقط نمی توانند به صورت register تعریف شوند.

- برای آرایه هایی که بیرون از تابع تعریف می شوند سطح ذخیره سازی خارجی بعنوان پیش فرض در نظر گرفته می شود.

- آرایه ها از نوع کلاس اتوماتیک برخلاف متغیرهای اتوماتیک نمی توانند هنگام تعریف آنها مقدار اولیه بپذیرند.

- هرگاه نام آرایه بعنوان آرگومان یک تابع ظاهر شود بعنوان آدرس اولین عنصر آرایه تعبیر می گردد.

رشته ها:

در زبان c رشته ها بعنوان آرایه ای از کاراکترها تعریف میشوند بطوریکه هر کاراکتر رشته درون یک عنصر از آرایه ذخیره می گردد.

- یک ثابت رشته ای آرایه ای را معرفی می کند که اندیس پایین آن صفر و اندیس بالای آن تعداد کاراکترهایی است که در رشته وجود دارد.

- هر رشته به کاراکتر null که پایان رشته را نشان می دهد خاتمه می یابد.

مثال: برنامه زیر ۱۵ رشته را می خواند و آنها را در سطرهای متوالی چاپ می کند:

```
# include<stdio.h>
main ( )
   {
     char name[12] ;
```

```
int  i ;
     for (i=1 , i < = 15 ; ++i)
        {
          scanf(“%s”, name);
          printf(“\n%s” , name) ;
        }
   }
```

مرتب سازی:

چنانچه مجموعه ای از داده ها یا اطلاعات بر اساس یک ویژگی یا نظم خاص سازمان دهی شوند این عمل را مرتب سازی گویند.

این عمل به منظور سرعت بخشیدن به عمل جستجو می باشد.

جستجو:

- هرگاه داخل مجموعه ای از عناصر دنبال عنصر خاصی بگردیم این عمل را جستجو نامند.
- جستجو در زمینه هایی چون بانکهای اطلاعاتی کاربرد زیادی دارد.

برخی روشهای مرتب سازی و جستجو:

- مرتب سازی حبابی
- مرتب سازی انتخابی
- جستجو به روش خطی
- جستجو به روش دودویی

مرتب سازی حبابی:

این روش از نظر منطق ساده ترین و از نظر کارایی پایین ترین روش مرتب سازی اطلاعات است.

```
void BubbleSort (int A[ ] , int n)
{
  int i , j , temp;
  for (i=1 ; i<n ; ++i)
   for (j=0 ; j<n-I ; ++j)
```

```
If (A[j] > A[j+1])
  {
    temp = A[j];
    A[j] = a[j+1];
    A[j+1] = temp;
   }
}
```

مرتب سازی انتخابی:

برتری این روش نسبت به روش حبابی آن است که تعداد تعویض عناصر کمتر شده و سرعت بیشتر می شود.

```
void SelectionSort (int a[ ] , int  n)
{
   int  i , max , temp ;
   for (i=1 ; i<n ;  + + i)
     {
```

```
max = 0 ;
for (j=1 ; j<= n-i+1 ; + + j)
  if(A[j] > A[max] )
    max = j ;
  if(max !=n-i)
    {
      temp = A[max] ;
      A[max] = A[n-i] ;
      A[n-i] = temp ;
    } } }
```

جستجو به روش خطی:

این روش ساده ترین راه برای جستجو در یک آرایه یا جدول نامرتب است. برای اینکار عنصر مورد جستجو را بطور متوالی با عناصر جدول مقایسه می کنیم.

مثال: تابع زیر عنصر x را در آرایه nعنصری A جستجو می کند:

```
int LinearSearch (int A[ ] , int  n , int
x)
   {
    int  i ;
    for (i = 0 ; i<n ; + + i)
       if (x = = A[i] )
          return (i+1) ;
    return (0) ;
   }
```

جستجو به روش دودویی:

این روش در صورتی امکان پذیر است که عناصر جدول مورد نظر مرتب شده باشند در این صورت از روش خطی سریعتر می باشد.

مثال: تابع زیر در یک آرایه مرتب شده nعنصری عنصرx را جستجو می کند:

```
int  BinarySearch (int A[ ] , int  n , int
   x)
{
    int  middle , L , H ;
     L = 0 ;
     H = n-1 ;
     while ( L <= H)
         {
          middle = (L+H)/2 ;
```

```
        if (x = = A[middle] )
              return (middle +1) ;
         if ( x >A[middle] )
             L= middle =1 ;
          else
              H = middle -1 ;
        }
    return(0)
}
```

توابع کتابخانه ای رشته ها:

نام تابع	عمل تابع
strcpy(s1,s2)	رشته s2 را روی رشته s1 کپی می کند.
strcat(s1,s2)	رشته s2 را به دنبال رشته s1 ملحق می کند.
strlen(s)	طول رشته s را برمی گرداند.
strcmp(s1,s2)	رشته s1 را با رشته s2 مقایسه می کند.

مثال: برنامه ای که دو رشته را از ورودی خوانده و آن دو را با هم مقایسه می کند:

```
# include<string.h>
# include<stdio.h>
main ( )
  {
    char  s1[80] , s2[80] , s3[80] ;
    gets (s1) ;
    gets (s2) ;
```

```
        printf ("\n Lengths :  %d%d\n"
,strlen (s1) , strlen (s2)) ;
   if  (!strcmp (s1 , s2))
        printf ("\n The strings are
equal\n") ;
        strcpy (s1 , s3)
        staract (s1 , s2) ;
        printf ("\n%s" , s3) ;
        printf ("\n%s" , s1) ;
}
```

فصل هشتم

اشاره گر

- اشاره گر متغیری است که آدرس متغیر دیگری را در خود نگه می دارد یعنی به آدرس متغیر دیگری اشاره می کند.

- اشاره گر روش غیرمستقیم دسترسی به داده ها است.

موارد استفاده از اشاره گرها:

- انتقال آدرس متغیرها به تابع فرعی
- برگرداندن چندین مقدار از تابع فرعی
- دستیابی به عناصر آرایه ها
- تشکیل ساختارهای پیچیده
- تخصیص حافظه به صورت پویا

در بحث اشاره گرها عملگر * به دو مفهوم بکار می رود:

- در معرفی متغیرها به عنوان عملگر اشاره گر
- برای دستیابی به مقدار متغیری که آدرس آن در یک متغیر اشاره گر قرار دارد.

مقداردهی اولیه اشاره گرها:

به هر نوع متغیر از نوع اشاره گر می توان هنگام اعلان آن مقدار اولیه نسبت داد در اینصورت مقدار اولیه مورد نظر باید یک آدرس یا null باشد.

نحوهٔ تعریف متغیر اشاره‌گر

نام متغیر اشاره‌گر * نوع داده‌ای که به آن اشاره می‌کند

```
int   * p ;
char   * pc ;
float    * fp ;
```

عملگرهای اشاره‌گر

& : آدرس عملوند خود را مشخص می‌کند.

عملوند آن نام یک متغیر است. & i

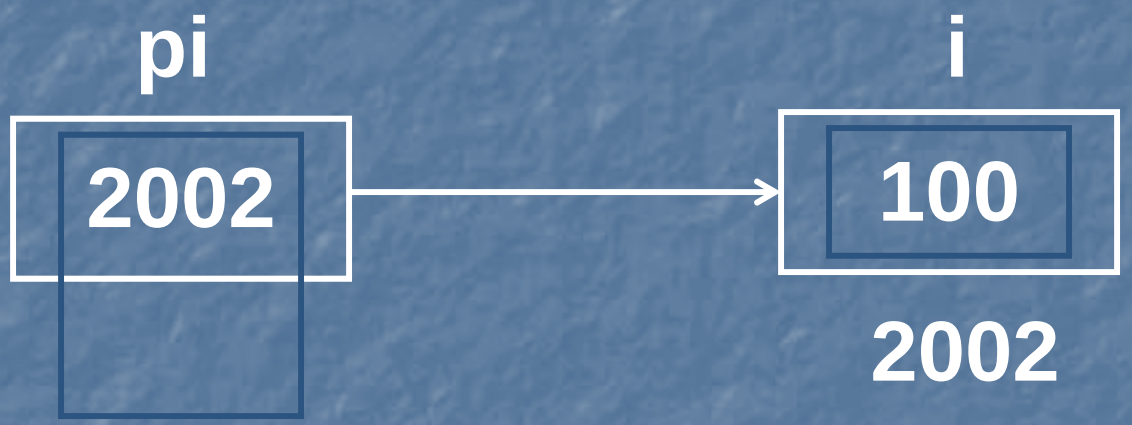
* : محتوای عملوند خود را مشخص می‌کند.

عملوند آن نام یک متغیر اشاره‌گر است. * p

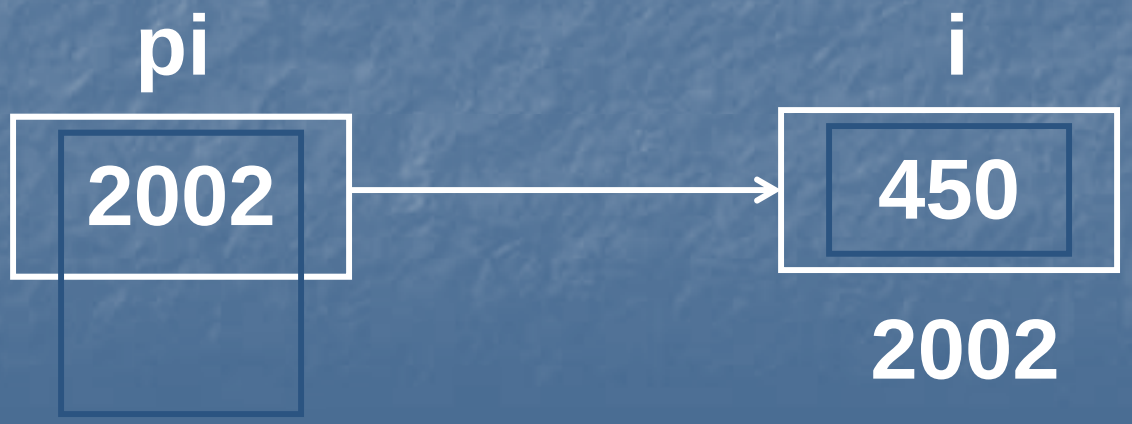
مثال

```
int  i = 100 ;
int  * pi ;
```

pi = & i ;



* pi = 450 ;



اشاره گر تهی:

- زبان c مفهوم اشاره گر null را پشتیبانی می کند و آن اشاره گری است که به هیچ مقدار معتبر اشاره نمی کند.
- هر اشاره گری که مقدار صحیح صفر به آن نسبت داده شده باشد.

مثال

```
int  * p1 , * p2 , i ;
p1 = p2 ;
p1 = p1 + 2 ;
p1 = p1 + i * 2 ;
p1 == p2
p2 <= p1
```

مثال: در برنامه زیر حلقه while تا موقعی که p یک اشاره گر NULL نباشد عمل تکرار را ادامه می دهد:

```
char *p ;
....
....
while (p)
  {
   ....
    ....
   }
```

عملیات روی اشاره گرها:

- عمل انتساب
- عمل محاسباتی
- مقایسه

- انتساب: به یک اشاره گر می توان آدرس یک متغیر یا مقدار صفر نسبت داد.

روش غلط

```
void  swap (int  a , int  b)
{
  int  temp ;
  temp = a ;
  a = b ;
  b = temp ;
}
```

روش صحیح

```
void  swap (int  *a , int  *b)
{
  int  temp ;
  temp = *a ;
  *a = *b ;
  *b = temp ;
}
```

```
void  main ()
{
   int x = 10 , y = 20 ;
   swap ( &x , &y ) ;
   printf (“x = %d , y = %d” , x , y) ;
}
```

خروجی:

x = 20 , y = 10

- محاسباتی: می توان یک مقدار صحیح را به اشاره گر اضافه کرد.

```
#include<stdio.h>
main ( )
{
   int   *px , *py ;
   static int A[6] = {1,2,3,4,5,6};
```

```
px = &A[0] ;
   py = &A[5] ;
   printf(“px=%x   py=%x” , px , py) ;
   printf(“\n py – px =%x” , py – px) ;
}
```

- مقایسه: اشاره گرهایی را که به داده های از یک نوع اشاره دارند می توان با هم مقایسه کرد.

```
px < py
Px >= py
Px = = py
Px != py
Px = = null
```

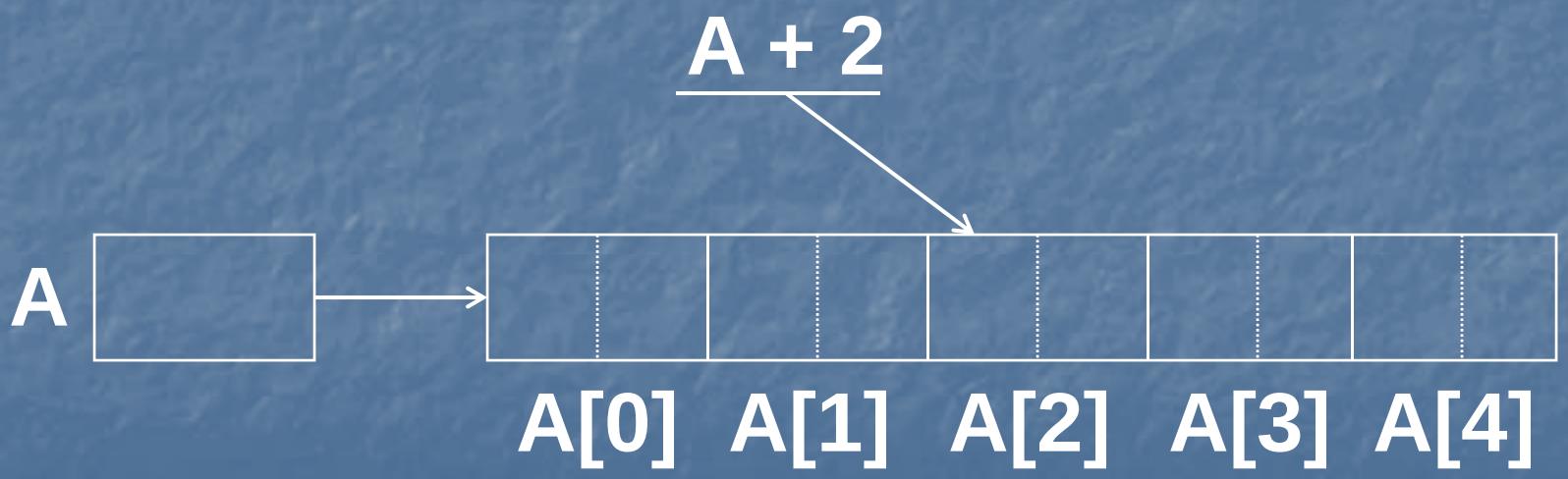
و یا:

```
if  (px<py)
   printf (“px  points to lower memory than
   py”);
else
   printf (“px  points to upper memory than
   py”);
```

اشاره‌گر و آرایه

نام آرایه اشاره‌گر به اولین عنصر آن است.

```
int A [5] ;
```



بنابراین A [2]

معادل * (A + 2)

```
A [2] = 100 ;
* (A + 2) = 100 ;
```

رشته و اشاره‌گر

رشته نیز یک آرایه است

و بنابراین

نام رشته اشاره‌گر به اولین عنصر آن است.

```
scanf (“%s” , s) ;
```

آرایه و رشته بعنوان پارامتر تابع

```
void  f (int  *a ; char *s) {
   ...
   a [2] = 100 ;
   strcpy (s , “Ali”) ;
}
```

```
void  main () {
  int  b [10] ;
  char  str [20] ;
  ...
  f (b , str) ;
}
```

تابع تخصیص حافظهٔ پویا

void * malloc (اندازهٔ حافظهٔ مورد نیاز به بایت)

```
int *p ;
p = (int *) malloc ( sizeof (int) ) ;
```

تعریف یک آرایه بصورت پویا

```
int  *A ;
A = (int *) malloc ( sizeof (int) * 100 )
  ;
```

با دستورات فوق آرایه‌ای با ظرفیت 100 عنصر با نام A و بصورت پویا ایجاد می‌شود.

تابع آزادسازی حافظهٔ پویا

free (اشاره‌گری که قبلا به آن حافظه اختصاص داده شده)

```
int  *A ;
A = (int *) malloc ( sizeof (int) * 100 ) ;
... // کار با حافظهٔ پویا
free (A) ;
```

روش غلط

```
int  n ;
float  A [n] ;
...
scanf (“ %d” , &n) ;
```

روش صحیح

```
int  n ;
float  *A ;
...
scanf (“%d” , &n) ;
A = (float *) malloc (sizeof (float) * n)
   ;
...  //  کار با آرایهٔ A
free (A) ;
```

فصل نهم

نوع داده کاربر

نوع داده هایی که کاربر با توجه به نیاز خود ایجاد می کند عبارتند از:

- ساختار
- Typedef
- اجتماع
- شمارشی

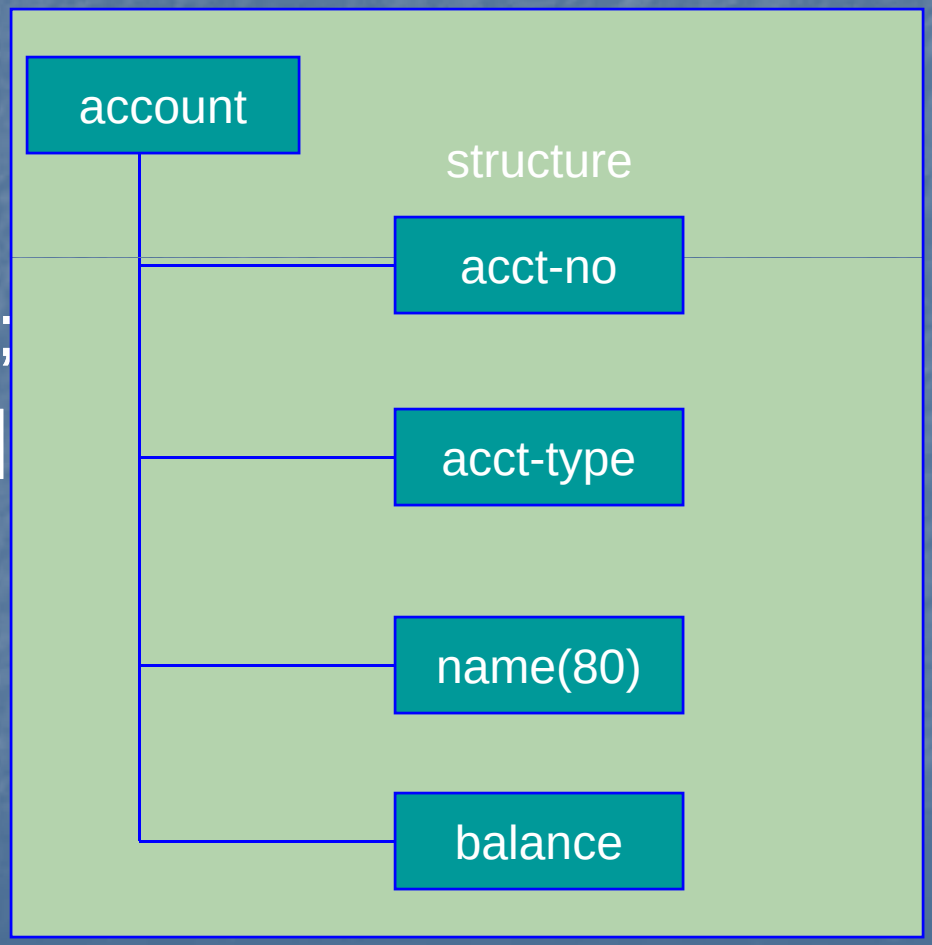
ساختار:

یک ساختار مجموعه ای از متغیرها است که تحت یک نام به آنها مراجعه می شود و هر عنصر از آن از نوع داده متفاوت می باشد

```
struct structure_name
{
  type variable1_name;
  ....
  ....
  type variablem_name;
  };
```

نمونه ای از تعریف ساختار همراه با نمای ظاهری:

```
struct account{
        int acct-no ;
        char acct-type;
        char name[80]
        float balance ;
          }
```



- به اعضا یک ساختار نمی توان کلاس حافظه اختصاص داد.
- هنگام تعریف یک ساختار نمی توان به اعضای آن مقدار اولیه نسبت داد.

قطعه برنامه زیر نحوه اختصاص مقادیر اولیه به اعضای یک متغیر ساختار را نشان می دهد:

```
struct  data {
                int month ;
                int  day ;
                int year ;
              } ;
struct  account {
                      int acc-no ;
```

```
        char  acc-type ;
        char  name[80] ;
        float  balance ;
        struct  data  lastpayment ;
    } ;
Static  struct  account  customer = {12746 ,'R' ,
  "payman noor" , 2986.50 , 5 , 24 , 75} ;
```

پردازش یک ساختار:

- اعضای یک ساختار معمولا مستقل از هم پردازش می شوند
- به هر عضو ساختار با استفاده از عملگر `.` رجوع می شود.

Variable.member

داده تعریف شده توسط کاربر:

در زبان c کاربران می توانند نام جدیدی برای نوع داده تعریف کنند این کار به کمک کلمه کلیدی typedef انجام می شود.

Typedef type name;

اجتماع:

- Union متغیری است که امکان ذخیره کردن انواع مختلف داده در مکان مشترکی از حافظه را فراهم می کند.

```
union tag {
        member 1;
        member 2;
        ...
        ...
        member m;
          };
```

نوع شمارشی:

یکی از انواع داده اسکالر است و اعضا آن ثابتهایی هستند که بعنوان شناسه نوشته می شوند.

```
enum tag {member1,...,memberm};
```

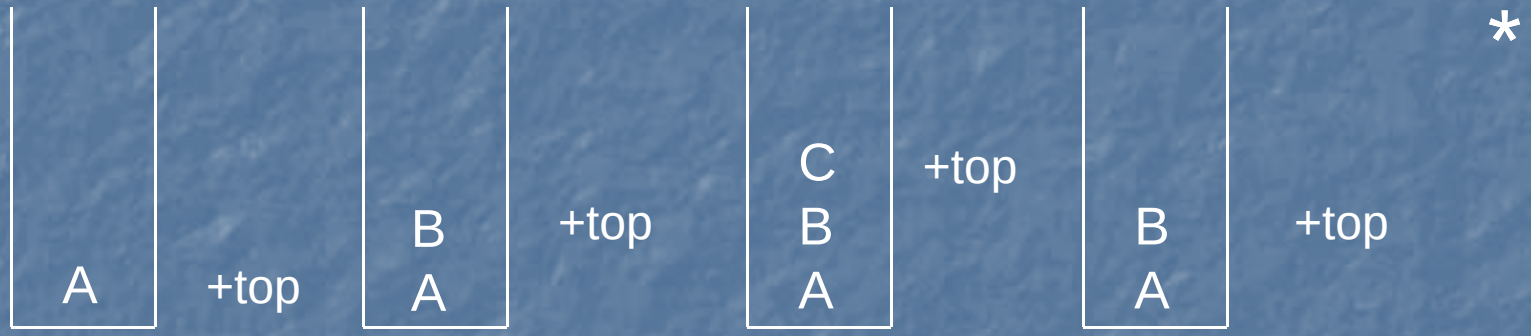
فصل دهم

پشته و صف

پشته:

پشته لیست مرتب شده ای است که هم جایگذاری و هم حذف از یک سمت آن که top نامیده می شود صورت می گیرد.

حذف و جایگذاری عناصر در یک پشته



از آنجا که آخرین عنصر وارده به پشته اولین عنصر حذف شده از آن می باشد پشته را بعنوان یک لیست LIFO شناسند.

برخلاف آرایه پشته در طول اجرای برنامه بطور پویا افزایش می یابد و محدودیتی نیز در تعداد عناصر پشته وجود ندارد.

پشته در زبان c بصورت ساختمانی با دو عضو تعریف می شود یکی آرایه ای از عناصر پشته دیگری متغیری که نشان دهنده موقعیت بالای پشته در آرایه است.

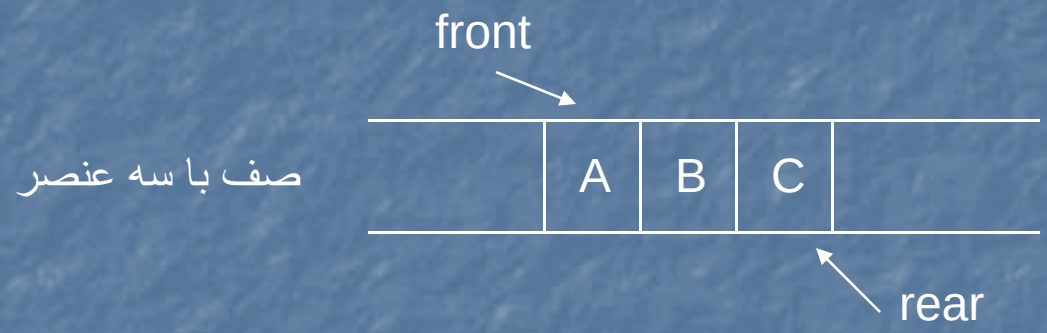
صف:

صف یک لیست مرتب شده است که تمام جایگذاریها در آن از یک سمت و تمام حذفهای آن از سمت دیگر انجام می گیرد.

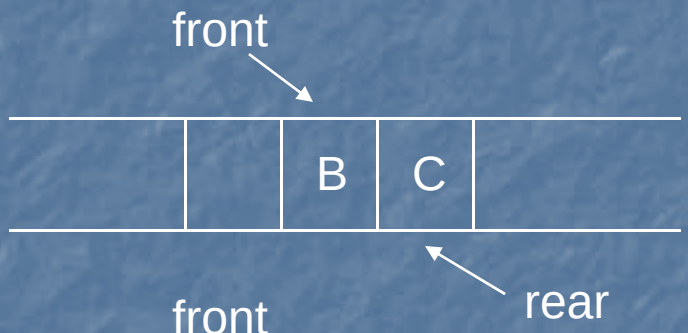
از آنجا که اولین عنصر وارده به یک صف اولین عنصری است که خارج می شود صف بعنوان لیست FIFO در نظر گرفته می شود.

برای نمایش یک صف علاوه بر یک آرایه یک بعدی به دو متغیر front و rear جهت نشان دادن ابتدا و انتهای عناصر صف نیاز داریم.

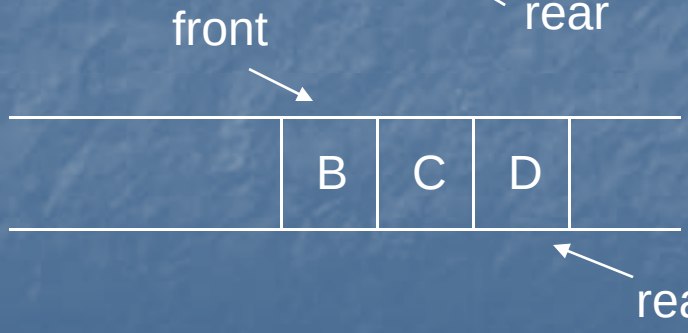
نمایش صف



صف با سه عنصر



حذف عنصر از صف



اضافه کردن عنصر به صف

*

صف اولویت:

صف اولویت ساختمان داده ای است که در آن ترتیب طبیعی عناصر نتایج اعمال ابتدایی آن را تعیین می کند و بر دو نوع است:

- صعودی
- نزولی

- صف اولویت صعودی:

صفی است که درج عناصر در آن به هر صورتی ممکن است اما در موقع حذف کوچکترین عنصر حذف خواهد شد.

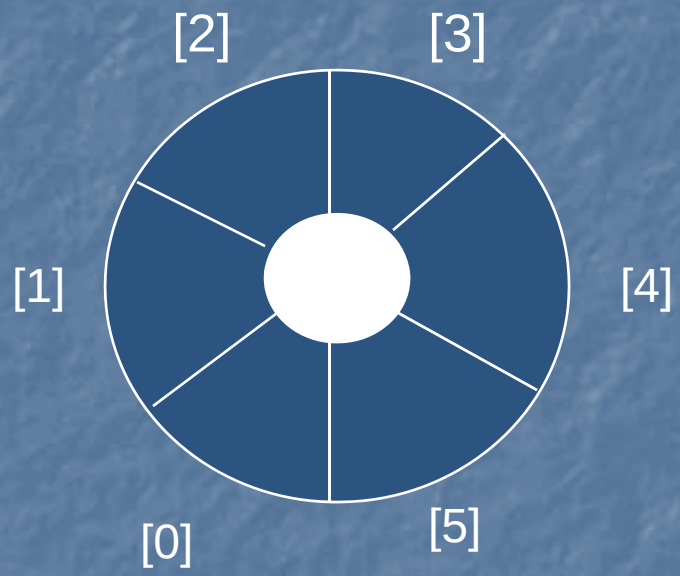
- صف اولویت نزولی:

همنند صف اولویت صعودی است با این تفاوت که عمل حذف موجب حذف بزرگترین عنصر صف می شود.

صف حلقوی:

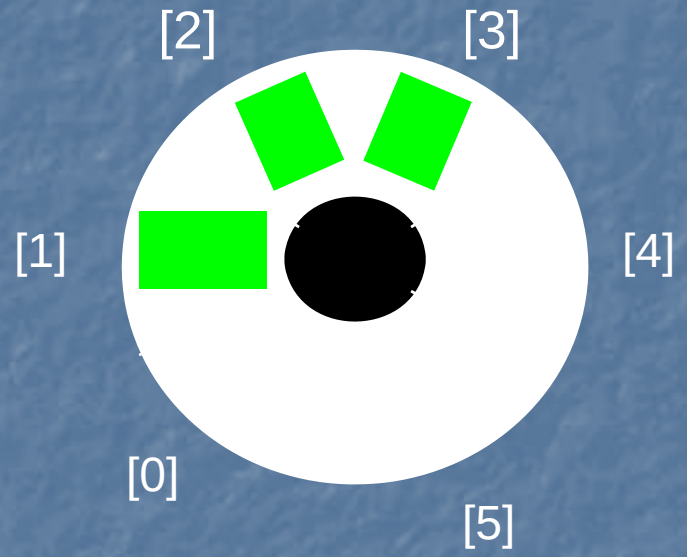
اگر آرایه را حلقوی فرض کنیم اندیس ابتدا همیشه به یک موقعیت عقب تر از اولین عنصر موجود در صف و اندیس انتها به انتهای فعلی صف اشاره می کند.

صف های حلقوی تهی و غیر تهی



front=0

rear=0



front=0

rear=3

فصل یازدهم

ساختار درختی

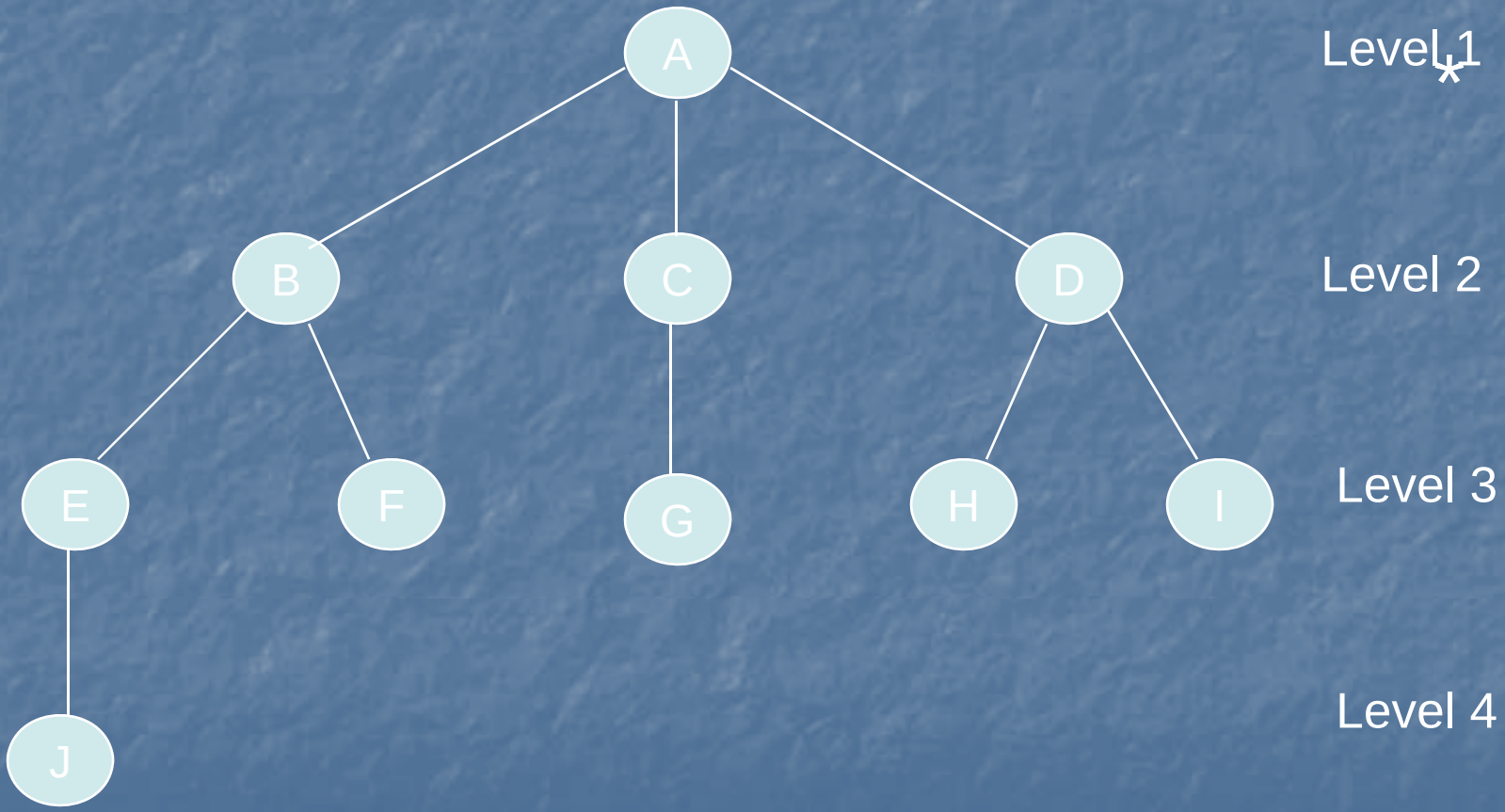
درخت:

بطور کلی درخت مجموعه محدودی از یک یا چند گره با شرایط زیر است:

۱- دارای گره خاصی به نام ریشه است.

۲- بقیه گره ها به n مجموعه مجزا که هریک زیردرخت ریشه نام دارند تقسیم شده اند.

نمونه ای از یک درخت



- به عنصر حاوی اطلاعات گره گفته می شود.
- تعداد زیردرختهای یک گره درجه آن گره نامیده می شود.
- گره هایی که درجه صفر دارند برگ نامیده می شوند.

- درجه یک درخت حداکثر درجه گره های آن درخت است.
- فرزندان یک گره گره های همزاد نام دارند.
- ارتفاع درخت به بیشترین سطح گره های آن درخت گفته می شود.

درخت دودویی:

اگر هر گره در یک درخت دارای دو انشعاب باشد به آن درخت درخت دودویی گفته می شود.

یک درخت دودویی یا تهی است یا حاوی مجموعه ای محدود از گره ها.

تفاوت بین یک درخت دودویی و یک درخت عادی:

- در هیچ درخت عادی صفر گره وجود ندارد.
- در یک درخت دودویی ترتیب فرزندان دارای اهمیت است.

خواص درختان دودویی:

۱- حداکثر تعداد گره ها در سطح iام یک درخت دودویی 2^i-1 است.

۲- حداکثر تعداد گره ها در یک درخت دودویی به عمق k برابر 2^k است.

۳- برای هر درخت دودویی غیر تهی اگر n0 تعداد گره های پایانی و n2 تعداد گره های درجه ۲ باشد داریم:

$$n0=n2+1$$

پیمایش درخت دودویی:

پیمایش درخت یا دستیابی به هر گره درخت فقط برای یکبار بر سه نوع است:

- Inorder
- Preorder
- Postorder

Inorder:

حرکت به سمت پایین به طرف چپ تا اخرین گره سپس گره بازیابی شده و به سمت راست حرکت را ادامه می دهیم.

```
void  inorder (treepointer  ptr}
  {
     if (ptr)
      {
```

```
inorder (ptr->leftchild);
        printf (“%d”, ptr->data);
        inorder (ptr->rightchild);
    }
 }
```

Preorder:

ابتدا گره و بعد انشعابات چپ را بازیابی کرده تا رسیدن به گره تهی ادامه می دهیم سپس به نزدیکترین گره والدی که دارای یک فرزند راست باشد مراجعه و همین کار را ادامه می دهیم.

```
void preorder (treepointer  ptr)
{
   if  (ptr)
    {
       printf (“%d”, ptr->data);
       preorder (ptr->leftchild);
       preorder (ptr->rightchild);
    }
}
```

:Postorder

این پیمایش دو فرزند یک گره را قبل از بازیابی آن گره ملاقات می کند یعنی فرزندان یک گره قبل از خود آن گره بازیابی می گردد.

```
void  postorder (treepointer ptr)
{
   if (ptr)
     {
```

```
        postorder (ptr->leftchild);
        postorder (ptr->rightchild);
        printf”%d”, ptr->data);
    }
}
```

درختان دودویی مساوی:

درختان دودویی را در صورتی مساوی نامیم که ساختاری نظیر هم داشته و اطلاعات موجود در گره های نظیرشان با هم برابر باشند.

```
int  equal (treepointer frist , treepointer second)
{
    return ((!frist && !second) | | (frist && second
  &&
                  (frist->data = = second->data)&&
                  equal (frist->leftchild , second-
  >leftchild) &&
                  equal (frist->rightchild , second-
  >rightchild))
}
```

درختان جستجوی دودویی:

یک درخت جستجو یک درخت دودویی است که ممکن است تهی باشد اگر درخت تهی نباشد دارای این خصوصیات است:

۱- هر عنصر دارای یک کلید منحصر به فرد است.

۲- کلیدهای واقع در زیردرخت غیرتهی چپ باید کمتر از مقدار کلید واقع در ریشه زیردرخت راست باشد.

۳- زیردرختان چپ و راست نیز خود درختان جستجوی دودویی می باشند.

جستجوی یک درخت دودویی:

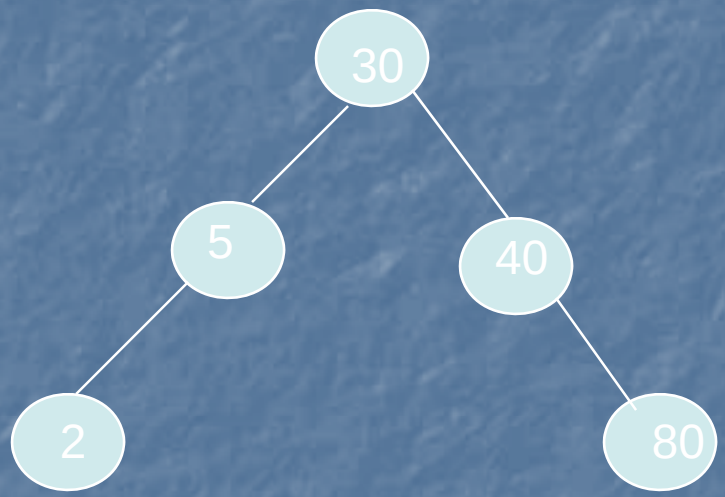
```
treepointer  search (treepointer  root , int  key)
{
        if (!root)
           return NULL ;
        if (key = = root->data)
```

```
        return  root ;
        if (key < root->data)
           return  search (root->leftchild , key) ;
        return  search (root-> rightchild , key);
}
```

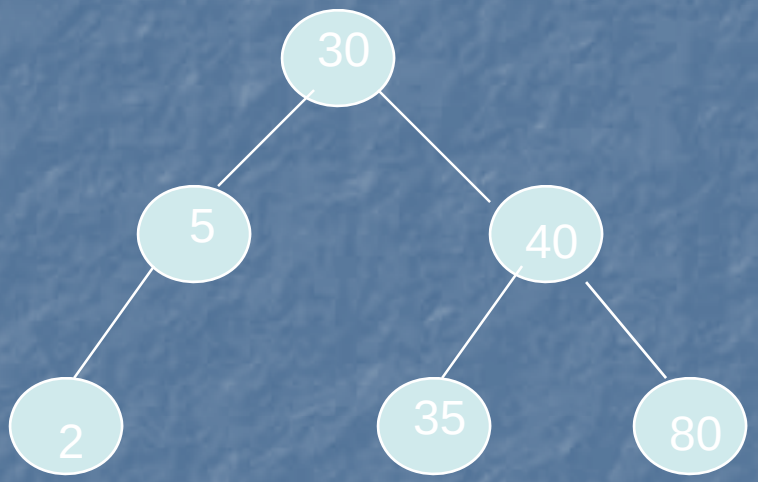
درج عنصر در درخت جستجوی دودویی:

برای درج عنصر باید درخت را جستجو نمود اگر جستجو ناموفق باشد می توان عنصر را در محلی که جستجو خاتمه پیدا کرده درج کرد.

درج عنصری به داخل درخت دودویی



جایگذاری ۸۰



جایگذاری ۳۵

*

- **درخت انتخابی:**

یک درخت دودویی است که هر گره آن کوچکتر از دو فرزند خود می باشد.

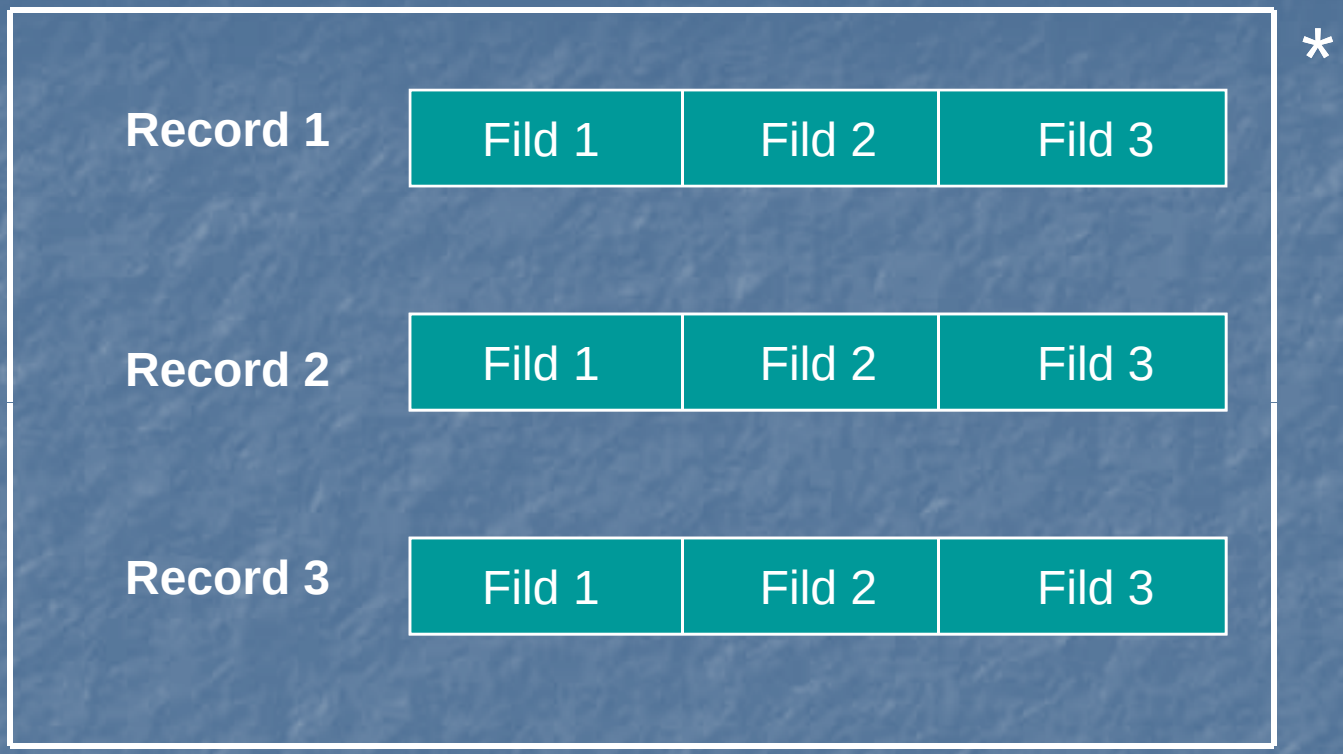
- **جنگل:**

اگر ریشه یک درخت را حذف کنیم آنگاه دارای یک جنگل خواهیم بود.

فصل دوازدهم

فایل

- هر فایل شامل مجموعه ای از داده های مرتبط به هم است.
- داده های مربوط به هر یک از اجزای فایل یک رکورد نام دارد.
- هر رکورد مجموعه ای از چند فیلد است.



فایل حاوی چند رکورد و فیلد

داده ها ممکن است به چهار روش در فایل ذخیره و بازیابی شوند:

۱- داده ها کاراکتر به کاراکتر در فایل نوشته شده و سپس کاراکتر به کاراکتر از فایل خوانده شوند.

۲- داده ها به صورت رشته ای از کاراکترها در فایل نوشته و به همان صورت رشته ای از کاراکترها نیز مورد دستیابی قرار گیرند.

۳- داده ها در حین نوشتن بر روی فایل با فرمت خاصی نوشته و با همان فرمت نیز خوانده شوند.

۴- داده ها به شکل رکورد بر روی فایل نوشته و بصورت رکورد نیز خوانده شوند.

انواع فایل:

داده ها در فایل به دو صورت ذخیره می شوند:

- text
- binary

تفاوت این دو در سه مورد است:

- تعیین انتهای فایل
- تعیین انتهای خط
- نحوه ذخیره شدن اعداد بر روی دیسک

در فایل text اعداد بصورت رشته ای از کاراکترها ذخیره می شوند ولی در فایل باینری اعداد به همان صورتی که در حافظه قرار می گیرند روی دیسک ذخیره می شوند.

دو روش از نظر نحوه ذخیره و بازیابی داده ها در فایل وجود دارد:

- سازمان فایل ترتیبی
- سازمان فایل تصادفی

- در سازمان فایل ترتیبی رکوردها به همان ترتیبی که از ورودی خوانده می شوند در فایل ذخیره و به همان ترتیب هم مورد دسترسی قرار می گیرند.

- در سازمان فایل تصادفی به هر رکرد یک شماره اختصاص می یابد بنابر این می توان به رکورد دلخواه دسترسی پیدا کرد بدون اینکه رکوردهای قبل آن خوانده شوند.

بازکردن و بستن فایل:

هر قبل از اینکه مورد استفاده قرار گیرد باید باز گردد.
مواردی که در حین باز کردن فایل مشخص می شوند عبارتند از:

- نام فایل
- نوع فایل ازنظر ذخیره اطلاعات
- نوع فایل ازنظر ورودی-خروجی

انواع فایل:

- فایل خروجی:

فایلی که طوری باز شود که فقط عمل نوشتن اطلاعات بر روی آن مجاز است.

- فایل ورودی:

فایلی که طوری باز شود که فقط عمل خواندن اطلاعات از آن امکان پذیر باشد.

- فایل ورودی و خروجی:

فایلی که طوری باز شود که عمل خواندن و نوشتن اطلاعات بر آن مجاز باشد.

تابع fopen برای باز کردن فایل مورد استفاده قرار گرفته و دارای الگوی زیر است:

```
FILE *fopen (char *filename, *mode)
```

مقادیر معتبر mode در تابع fopen()

mode	مفهوم
r (rt)	فایلی از نوع text را بعنوان ورودی باز می کند.
w (wt)	فایلی از نوع text را بعنوان خروجی باز می کند.
a (at)	فایل را طوری باز می کند که بتوان اطلاعاتی را به انتهای آن اضافه نمود.
rb	فایلی از نوع باینری را بعنوان ورودی باز می کند.
wb	فایلی از نوع باینری را بعنوان خروجی باز می کند.
ab	فایل موجود از نوع باینری را طوری باز می کند که بتوان اطلاعاتی را به انتهای آن اضافه نمود.

توابع putc و getc:

برای نوشتن و خواندن یک کاراکتر در فایلی که قبلا باز شده است استفاده می شوند.

```
int putc (int ch , FILE*fp)

int getc (FILE*fp)
```

توابع putw وgetw :

این دو تابع مشابه getc و putc هستند ولی برای خواندن و نوشتن مقادیر صحیح از یک فایل به فایل دیگر بکار می روند.

```
putw(50 , fp);
```

توابع fputs و fgets:

برای نوشتن رشته ها در فایل از تابع fputs و برای خواندن رشته ها از فایل از تابع fgets استفاده می گردد.

```
int fputs (const char *str , FILE *fp)
char *fgets (char *str , int length , FILE *fp)
```

فایلهای ورودی و خروجی:

می توان یک فایل را به عنوان وسیله ورودی و خروجی مورد استفاده قرار داد برای این منظور در تابع fopen بجای mode:

- r+t یا r+ برای باز کردن فایل text

- w+t یا w+ برای ایجاد یک فایل text

- a+t یا a+ برای ایجاد یک فایل text و یا باز کردن فایل text موجود

- r+b برای باز کردن فایل باینری موجود

تابع remove:

برای حذف فایلهای غیر ضروری می توان از این تابع استفاده کرد:

```
int remove (char *filename)
```

توابع fprintf وfscanf:

هرگاه لازم باشد داده ها با فرمت خاصی در فایل نوشته یا از آن خوانده شوند بکار می روند:

int fprintf (FILE *fp , “*control_string,...”, char arg,...)

int fscanf (FILE *fp , “*control_string,...”, char arg,...)

توابع fwrite و fread:

برای ورودی خروجی رکورد و سایر ورودی خروجی ها استفاده می شود:

int fread (void *buffer, int num_byte, int count,FILE*fp)

int fwrite (void *buffer, int num_byte, int count,FILE*fp)

مثال

```
struct  student  stud ;
fread ( &stud ,
        sizeof (struct  student) ,
        1 , fp ) ;
```

مثال

```
int  a [20] ;
fread (a , sizeof (int) , 20 , fp )
  ;
```

دریافت 20 عدد از فایل و قرار دادن آن در آرایه

نکته:

fprintf (FILE *fp , " عبارت 1 " , عبارت 2
)

fscanf (FILE *fp , " عبارت 1 " , عبارت 2
)

دقیقا همانند printf و scanf

با تفاوت

مثال

```
FILE  *fp1 , *fp2 ;
fp1 = fopen (“c:\test.txt” , “rt”) ;
fp2 = fopen (“c:\ali.dat” , “wb”) ;
...
fscanf (fp1 , “%d %f %s” , &i , &f ,
   str) ;
fprintf (fp2 , “%f, %s” , f , str) ;
```

تابع fseek:

برای خاندن و نوشتن داده ها بصورت تصادفی بکار می رود:

```
int fseek( FILE *fp,long int num_bytes,int origin);
```

دستگاه های ورودی و خروجی استاندارد:

اشاره گر فایل	نام دستگاه (فایل)
stdin	دستگاه ورودی استاندارد (صفحه کلید)
stdout	دستگاه خروجی استاندارد (صفحه نمایش)
stderr	دستگاه استاندارد جهت ثبت پیامهای خطا (صفحه نمایش)
stdprn	دستگاه استاندارد چاپ (چاپگر موازی)
stdaux	پورت سری

فصل سیزدهم

توابع کتابخانه ای

نام تابع	کاربرد	الگو
clrscr()	پاک کردن صفحه نمایش	void clrscr (void)
atoi()	تبدیل نوع رشته به نوع integer	int atoi (const char *s)
atof()	تبدیل نوع رشته به نوع ممیز شناور	double atof (const char *s)
atol()	تبدیل نوع رشته به نوع long	long atol (const char *s)

نام تابع	کاربرد	الگو
sqrt()	محاسبه جذر عدد مثبت	double sqrt (double x)
pow()	محاسبه توانهای یک مبنا	double pow (double x,double y)
abs()	محاسبه قدر مطلق اعداد صحیح	int abs (int x)
fabs()	محاسبه قدر مطلق اعداد اعشاری	double fabs(double x)

نام تابع	کاربرد	الگو
cabs ()	محاسبه قدرمطلق اعداد موهومی	double cabs (struct complex num)
sin ()	محاسبه سینوس زاویه	double sin(double arg)
cos ()	محاسبه کسینوس زاویه	double cos(double arg)
tan ()	محاسبه تانژانت زاویه	double tan(double arg)

نام تابع	کاربرد	الگو
asin ()	محاسبه آرک سینوس یک عدد	double asin(double arg)
acos()	محاسبه آرک کسینوس یک عدد	double acos(double arg)
atan ()	محاسبه آرک تانژانت یک عدد	double atan(double arg)
atan2 ()	محاسبه آرک تانژانت نتیجه تقسیم آرگومان اول بر دوم	double atan2(double y,double x)

نام تابع	کاربرد	الگو
sinh ()	محاسبه سینوس هیپربولیک زاویه	double sinh(double arg)
cosh ()	محاسبه کسینوس هیپربولیک عدد	double cosh(double arg)
tanh ()	محاسبه تانژانت هیپربولیک زاویه	double tanh(double arg)
ceil ()	کوچکترین عدد صحیح بزرگتر یا مساوی با یک عدد	double ceil(double x)

نام تابع	کاربرد	الگو
log ()	لگاریتم طبیعی یک عدد مثبت	double log (double x)
log10 ()	لگاریتم مبنای ۱۰ اعداد مثبت	double log10 (double x)
exp ()	محاسبه توانی از e	double exp (double arg)
ldexp ()	num*2^exp	double ldexp (double num,int exp)

نام تابع	کاربرد	الگو
floor ()	بزرگترین مقدار صحیح کوچکتر یا مساوی یک عدد	double floor (double x)
fmod ()	محاسبه باقیمانده تقسیم	double fmod (double x, double y)
modf ()	عددی را به دو قسمت صحیح و اعشاری تقسیم می کند	double modf (double x,int*y)
bypot ()	محاسبه وتر	double hypot (double x, double y)

نام تابع	کاربرد	الگو
poly ()	ارزیابی چندجمله ای	double poly (double x,int n,double C[])
tolower( )	تبدیل به حرف کوچک انگلیسی	int tolower (int ch)
toupper ()	تبدیل به حرف بزرگ انگلیسی	int toupper (int ch)
isulpha()	تشخیص حرف بودن کاراکتر	int isalpha (int ch)

نام تابع	کاربرد	الگو
isdigit ()	آیا کاراکتر رقم است	int isdigit (int ch)
islower( )	آیا کاراکتر حرف کوچک است	int islower (int ch)
ispunct( )	آیا کاراکتر یکی از کاراکترهای ویرایشی است	int ispunct (int ch)
isupper( )	آیا کاراکتر حرف بزرگ است	int isupper (int ch)

نام تابع	کاربرد	الگو
strset ()	رشته را با کاراکتر پر میکند	strset (str , 'x');
strnset ()	کاراکتری را به تعداد دفعات مشخص در رشته کپی میکند	char *strnset(char*str,char ch, signed count)
memchr ()	کاراکتری را در آرایه جستجو میکند	void *memchr(const void*buffer,int ch, unsigned count)
memcpy( )	قسمتی از آرایه را در آرایه دیگر کپی میکند	void *memcpy(void*to,const void*from, unsigned count)

نام تابع	کاربرد	الگو
memset( )	کاراکتری را در چند عنصر آرایه کپی میکند	void *memset(void*buf,int ch, unsigned count)
strcspn ()	رشته ای را در رشته دیگر جستجو میکند	int strcspn (const *str1 const *str2)
strerror ()	ارسال پیام خطا	char *strerror (char *str)
strlwr ()	حروف بزرگ یک رشته را کوچک میکند	char *strlwr (char *str)

نام تابع	کاربرد	الگو
strncat ()	الحاق قسمتی از رشته را به انتهای رشته دیگر	char * strncat (char *str1, const char *str2 , unsigned count)
strncmp ()	مقایسه تعداد مشخصی از کاراکترهای دو رشته	int strncmp (const *str , const char *str , unsigned count
strncpy ()	کپی تعداد مشخصی از کاراکترهای رشته در رشته دیگر	char *strncpy (char *str1 , const char *str2 , unsigned count)
strrchr ()	جستجوی کاراکتری در یک رشته	char *strrchr (const char *str , int ch)

نام تابع	کاربرد	الگو
strspn ()	جستجوی رشته ای در رشته دیگر و برگشت طول آن	unsigned strspn (const char *str1 , const char *str2)
strrev ()	معکوس کردن کاراکترهای یک رشته	char *strrev (char *str)
struper ()	کلیه حروف کوچک را بزرگ میکند	char *struper (char *str)

نام تابع	کاربرد	الگو
free ()	آزادسازی حافظه اخذشده از سیستم	void free (void *)
calloc ()	اخذ حافظه از سیستم	void *calloc (unsigned unm , unsigned size)
malloc ()	اخذ حافظه از سیستم	void *malloc (unsigned size)
realloc ()	تغییر میزان حافظه اختصاص یافته	*realloc (void *ptr , unsigned size)

نام تابع	کاربرد	الگو
clreol ()	اطلاعات خط جاري از محل فعلي مکان‌نما تا انتهاي خط جاري از چپ به راست پاک ميشود	void clreol()
deline()	یک خط حذف می شود	void delline()
insline()	یک خط خالي زير خطي که مکان نما روي آن قرار دارد ايجاد ميشود	void insline()
gotoxy()	در محیط متن میتوان مکان‌نما را در محل دیگری از پنجره فعال قرار داد	void gotoxy(int x , int y)

نام تابع	کاربرد	الگو
moveto()	مکان‌نما را به محلی با مختصات x و y انتقال می‌دهد	void far moveto(int x , int y)
getx() و **gety()**	مختصات جاری مکان‌نما را روی صفحه گرافیکی برمی‌گردانند	int far getx(void) int far gety(void)
wherex() و **wherey()**	مختصات جاری مکان‌نما را نسبت به پنجره موجود برمی‌گرداند	int wherex(void) int wherey(void)
gettext()	کپی کردن متن از صفحه نمایش به بافر	int gettext(int left , int top , int right , int bottom , void *buffer)

نام تابع	کاربرد	الگو
puttext()	کپي کردن متن از بافر به صفحه نمايش	int puttext(int left , int top , int right , int bottom , void *buffer)
movetext()	کپي کردن متن از قسمت صفحه تصوير به قسمت ديگر	int pettext(int left , int top , int right , int bottom , int newleft , newtop)
window()	پنجره متن با ابعاد مشخص شده را فعال می‌کند	void window(int left , int top , int right , int bottom)
textcolor()	رنگ متن را مشخص می‌کند	void textcolor(int color)

نام تابع	کاربرد	الگو
textbackground()	تعیین رنگ زمینه متن	void textbackground(int color)
textmode()	تغییر حالت صفحه تصویر	void textmode(int mode)
initgraph()	تعیین تطبیق‌دهنده حالت گرافیکی	void far initgraph(int far *driver , int far *mode , char far *path)
setbkcolor()	تعیین رنگ زمینه در حالت گرافیک	void far setbkcolor(int color)

نام تابع	کاربرد	الگو
setpalette()	تغییر جعبه رنگ	void far setpalette(int index , int color)
closegraph()	پایان دادن به حالت تصویری گرافیکی	void far closegraph()
restorecrmode()	برنامه را خاتمه و تطبیق‌دهنده گرافیکی را به حالت اول برمیگرداند	void far restorecrmode()
putpixel()	رنگ مشخص شده را به محل تعیین شده با x و y می نویسد	void far putpixel(int x , int y , int color)

نام تابع	کاربرد	الگو
line()	خطی از نقطه‌ای به نقطه دیگر رسم می‌کند	void far line(int startx , int starty , int endx , int endy)
lineto()	خطی از مکان جاری به نقطه دیگر رسم می‌کند	void far lineto(int x , int y)
circle()	دایره‌ای با مرکز و شعاع مشخص رسم می‌کند	void circle(int x , int y , int radius)
ellips()	بیضی‌ای به مرکز x و y و دو شعاع xradius و yradius با رنگ جاری رسم می‌کند	void far ellipse(int x , int y , int start , int end , int xradius , int yradius)

نام تابع	کاربرد	الگو
drawpoly()	با استفاده از رنگ جاری چندضلعی رسم میکند	void far drawpoly(int numpoints , int far *points)
floodfill()	برای پر کردن شکلی بسته با رنگ مشخص	void floodfill(int x , int y , int bordercolor)
setfillstyle()	شیوه پر کردن شکل را تغییر می‌دهد	void far setfillstyle(int pattern , int color)
outtext()	برای نمایش متن در حالت گرافیکی	void far outtext(char *str)

نام تابع	کاربرد	الگو
outtextxy()	برای نشان دادن متن در حالت گرافیکی در محل خاصی با مختصات x و y	void far outtext(int x , int y , char *str)
settextstyle ()	برای انتخاب نوع فونت، اندازه فونت و نوع نگارش افقی و عمودی	void far settextstyle(int font , int direction , int charsize)
getimage()	برای کپی کردن یک ناحیه از پنجره گرافیک با مختصات مشخص به بافر	void far getimage(int left , int top, int right , int bottom , void far *buf)

نام تابع	کاربرد	الگو
putimage()	برای نمایش دادن محتوای یک بافر از داده‌های گرافیکی	void far getimage(int left , int top, void far *buf , int op)
imagesize()	اندازه بافر برحسب بایت برای ناحیه داده شده	void far getimage(int left , int top, int right , int bottom)
arc()	رسم کمانی از start تا end را در امتداد دایره فرضی با مختصات مرکز x و y و نیز شعاع radius	void far arc(int x , int y , int star, int end , int radius)

نام تابع	کاربرد	الگو
bar()	میله‌ای مستطیل شکل رسم و با رنگ و الگوی جاری پر می‌کند	void far bar(int left , int top , int right , int bottom)
bar3d()	میله‌ای مستطیل شکل و سه بعدی به عمق با اندازه depth پیکسل رسم و با رنگ و الگوی جاری پر می‌کند.	void far bar3d(int left , int top , int right , int bottom , int depth , int topflag)
getcolor()	رنگ جاری برای رسم را برمی‌گرداند	int far getcolor(void)

پایان

سایت مرجع دانشجوی پیام نور

- ✓ نمونه سوالات پیام نور : بیش از ۱۱۰ هزار نمونه سوال همراه با پاسخنامه تستی و تشریحی
- ✓ کتاب ، جزوه و خلاصه دروس
- ✓ برنامه امتحانات
- ✓ منابع و لیست دروس هر ترم
- ✓ دانلود کاملا رایگان بیش از ۱۴۰ هزار فایل مختص دانشجویان پیام نور