اصول برنامه‌نویسی کامپیوتر

تابع:

تبدیل زیرالگوریتم و زیرفلوچارت به برنامه

محمّد فرشی

آزمایشگاه الگوریتم های ترکیباتی و بهندسی- دانشکدهٔ علوم ریاضی- دانشگاه یزد

۱۴۰۳-۱

تابع

بیان زیرالگوریتم و زیرفلوچارت به زبان C++:

- در فصل‌های قبلی از زیرالگوریتم‌ها و زیرفلوچارت‌ها برای تقسیم حل یک مسئله به حل تعدادی زیرمسئله استفاده کردیم.
- در اینجا، روش تبدیل این زیرالگوریتم و زیرفلوچارت‌ها به برنامه و همچنین روش استفاده از آنها در سایر زیرالگوریتم‌ها را بیان می‌کنیم.
- تابع در حقیقت، یک زیرالگوریتم و زیرفلوچارت تبدیل شده به برنامه است.
- با مقایسهٔ زیرالگوریتم و زیرفلوچارت با الگوریتم و فلوچارت، با دو تفاوت برخورد می‌کنیم.
- اول؛ یک الگوریتم یا فلوچارت با شروع آغاز می‌شود در حالی که در زیرالگوریتم و زیرفلوچارت، شروع با نام آن زیرالگوریتم و زیرفلوچارت و لیست پارامترهای آن است.
- دوم؛ در انتهای یک الگوریتم و فلوچارت با پایان به انتها می‌رسد اما زیرالگوریتم و زیرفلوچارت با دستور برگردان.

تابع

بیان زیرالگوریتم و زیرفلوچارت به زبان C++:

- در فصل‌های قبلی از زیرالگوریتم‌ها و زیرفلوچارت‌ها برای تقسیم حل یک مسئله به حل تعدادی زیرمسئله استفاده کردیم.
- در اینجا، روش تبدیل این زیرالگوریتم و زیرفلوچارت‌ها به برنامه و همچنین روش استفاده از آنها در سایر زیرالگوریتم‌ها را بیان می‌کنیم.
- تابع در حقیقت، یک زیرالگوریتم و زیرفلوچارت تبدیل شده به برنامه است.
- با مقایسهٔ زیرالگوریتم و زیرفلوچارت با الگوریتم و فلوچارت، با دو تفاوت برخورد می‌کنیم.
- اول؛ یک الگوریتم یا فلوچارت با شروع آغاز می‌شود در حالی که در زیرالگوریتم و زیرفلوچارت، شروع با نام آن زیرالگوریتم و زیرفلوچارت و لیست پارامترهای آن است.
- دوم؛ در انتهای یک الگوریتم و فلوچارت با پایان به انتها می‌رسد اما زیرالگوریتم و زیرفلوچارت با دستور برگردان.

تابع

بیان زیرالگوریتم و زیرفلوچارت به زبان C++:

- در فصل‌های قبلی از زیرالگوریتم‌ها و زیرفلوچارت‌ها برای تقسیم حل یک مسئله به حل تعدادی زیرمسئله استفاده کردیم.
- در اینجا، روش تبدیل این زیرالگوریتم و زیرفلوچارت‌ها به برنامه و همچنین روش استفاده از آنها در سایر زیرالگوریتم‌ها را بیان می‌کنیم.
- تابع در حقیقت، یک زیرالگوریتم و زیرفلوچارت تبدیل شده به برنامه است.
- با مقایسهٔ زیرالگوریتم و زیرفلوچارت با الگوریتم و فلوچارت، با دو تفاوت برخورد می‌کنیم.
- اول؛ یک الگوریتم یا فلوچارت با شروع آغاز می‌شود در حالی که در زیرالگوریتم و زیرفلوچارت، شروع با نام آن زیرالگوریتم و زیرفلوچارت و لیست پارامترهای آن است.
- دوم؛ در انتهای یک الگوریتم و فلوچارت با پایان به انتها می‌رسد اما زیرالگوریتم و زیرفلوچارت با دستور برگردان.

تابع

بیان زیرالگوریتم و زیرفلوچارت به زبان C++:

- در فصل‌های قبلی از زیرالگوریتم‌ها و زیرفلوچارت‌ها برای تقسیم حل یک مسئله به حل تعدادی زیرمسئله استفاده کردیم.
- در اینجا، روش تبدیل این زیرالگوریتم و زیرفلوچارت‌ها به برنامه و همچنین روش استفاده از آنها در سایر زیرالگوریتم‌ها را بیان می‌کنیم.
- تابع در حقیقت، یک زیرالگوریتم و زیرفلوچارت تبدیل شده به برنامه است.
- با مقایسهٔ زیرالگوریتم و زیرفلوچارت با الگوریتم و فلوچارت، با دو تفاوت برخورد می‌کنیم.
- اول؛ یک الگوریتم یا فلوچارت با شروع آغاز می‌شود در حالی که در زیرالگوریتم و زیرفلوچارت، شروع با نام آن زیرالگوریتم و زیرفلوچارت و لیست پارامترهای آن است.
- دوم؛ در انتهای یک الگوریتم و فلوچارت با پایان به انتها می‌رسد اما زیرالگوریتم و زیرفلوچارت با دستور برگردان.

تابع

بیان زیرالگوریتم و زیرفلوچارت به زبان C++:

- در فصل‌های قبلی از زیرالگوریتم‌ها و زیرفلوچارت‌ها برای تقسیم حل یک مسئله به حل تعدادی زیرمسئله استفاده کردیم.
- در اینجا، روش تبدیل این زیرالگوریتم و زیرفلوچارت‌ها به برنامه و همچنین روش استفاده از آنها در سایر زیرالگوریتم‌ها را بیان می‌کنیم.
- تابع در حقیقت، یک زیرالگوریتم و زیرفلوچارت تبدیل شده به برنامه است.
- با مقایسهٔ زیرالگوریتم و زیرفلوچارت با الگوریتم و فلوچارت، با دو تفاوت برخورد می‌کنیم.
- اول؛ یک الگوریتم یا فلوچارت با شروع آغاز می‌شود در حالی که در زیرالگوریتم و زیرفلوچارت، شروع با نام آن زیرالگوریتم و زیرفلوچارت و لیست پارامترهای آن است.
- دوم؛ در انتهای یک الگوریتم و فلوچارت با پایان به انتها می‌رسد اما زیرالگوریتم و زیرفلوچارت با دستور برگردان.

تابع

بیان زیرالگوریتم و زیرفلوچارت به زبان C++:

- در فصل‌های قبلی از زیرالگوریتم‌ها و زیرفلوچارت‌ها برای تقسیم حل یک مسئله به حل تعدادی زیرمسئله استفاده کردیم.
- در اینجا، روش تبدیل این زیرالگوریتم و زیرفلوچارت‌ها به برنامه و همچنین روش استفاده از آنها در سایر زیرالگوریتم‌ها را بیان می‌کنیم.
- تابع در حقیقت، یک زیرالگوریتم و زیرفلوچارت تبدیل شده به برنامه است.
- با مقایسهٔ زیرالگوریتم و زیرفلوچارت با الگوریتم و فلوچارت، با دو تفاوت برخورد می‌کنیم.
- اول؛ یک الگوریتم یا فلوچارت با شروع آغاز می‌شود در حالی که در زیرالگوریتم و زیرفلوچارت، شروع با نام آن زیرالگوریتم و زیرفلوچارت و لیست پارامترهای آن است.
- دوم؛ در انتهای یک الگوریتم و فلوچارت با پایان به انتها می‌رسد اما زیرالگوریتم و زیرفلوچارت با دستور برگردان.

تابع

بیان زیرالگوریتم در C++:

- معرفی زیرالگوریتم به صورت زیر است:

(... ,نام پارامتر دوم نوع پارامتر دوم ,نام پارامتر اول نوع پارامتر اول)نام تابع نوع مقدار بازگشتی

- مثال: int IsPrime(int n) double BMM(int m,float n, int constant)
- اگر زیرالگوریتمی مقدار بازگشتی ندارد، نوع مقدار بازگشتی را void قرار می‌دهیم. مثال:
void print(int m,int n)
- اگر زیرالگوریتم هیچ پارامتری ندارد، داخل پرانتز جلو نام آن را خالی می‌گذاریم یا void قرار می‌دهیم.
مثال: double test() یا double test(void)
- دقت کنید که در انتهای دستور و بعد از پرانتز بسته سمی کالن نمی‌آید.
- برای عبارت «برگردان» از دستور return استفاده می‌کنیم. اگر مقداری باید برگشت داده شود آن مقدار با یک فاصله بعد از دستور return قرار می‌گیرد و سپس سمی کالن قرار می‌گیرد.
مثال: return ; یا return sum;
- تابع حداکثر یک مقدار برمی‌گرداند.

تابع

بیان زیرالگوریتم در C++:

- معرفی زیرالگوریتم به صورت زیر است:

نوع مقدار بازگشتی نام تابع(نوع پارامتر اول نام پارامتر اول ,نوع پارامتر دوم نام پارامتر دوم ,...)

- مثال: int IsPrime(int n) double BMM(int m,float n, int constant)
- اگر زیرالگوریتمی مقدار بازگشتی ندارد، نوع مقدار بازگشتی را void قرار می‌دهیم. مثال:
void print(int m,int n)
- اگر زیرالگوریتم هیچ پارامتری ندارد، داخل پرانتز جلو نام آن را خالی می‌گذاریم یا void قرار می‌دهیم.
مثال: double test() یا double test(void)
- دقت کنید که در انتهای دستور و بعد از پرانتز بسته سمی کالن نمی‌آید.
- برای عبارت «برگردان» از دستور return استفاده می‌کنیم. اگر مقداری باید برگشت داده شود آن مقدار با یک فاصله بعد از دستور return قرار می‌گیرد و سپس سمی کالن قرار می‌گیرد.
مثال: return ; یا return sum;
- تابع حداکثر یک مقدار برمی‌گرداند.

تابع

بیان زیرالگوریتم در C++:

- معرفی زیرالگوریتم به صورت زیر است:

(... ,نام پارامتر دوم نوع پارامتر دوم ,نام پارامتر اول نوع پارامتر اول)نام تابع نوع مقدار بازگشتی

- مثال: int IsPrime(int n) double BMM(int m,float n, int constant)
- اگر زیرالگوریتمی مقدار بازگشتی ندارد، نوع مقدار بازگشتی را void قرار می‌دهیم. مثال:

void print (int m,int n)

- اگر زیرالگوریتم هیچ پارامتری ندارد، داخل پرانتز جلو نام آن را خالی می‌گذاریم یا void قرار می‌دهیم.
مثال: double test () یا double test(void)
- دقت کنید که در انتهای دستور و بعد از پرانتز بسته سمی کالن نمی‌آید.
- برای عبارت «برگردان» از دستور return استفاده می‌کنیم. اگر مقداری باید برگشت داده شود آن مقدار با یک فاصله بعد از دستور return قرار می‌گیرد و سپس سمی کالن قرار می‌گیرد.
مثال: return ; یا return sum;
- تابع حداکثر یک مقدار برمی‌گرداند.

تابع

بیان زیرالگوریتم در C++:

- معرفی زیرالگوریتم به صورت زیر است:

(... ,نام پارامتر دوم نوع پارامتر دوم ,نام پارامتر اول نوع پارامتر اول)نام تابع نوع مقدار بازگشتی

- مثال: int IsPrime(int n) double BMM(int m,float n, int constant)
- اگر زیرالگوریتمی مقدار بازگشتی ندارد، نوع مقدار بازگشتی را void قرار می‌دهیم. مثال:

void print (int m,int n)

- اگر زیرالگوریتم هیچ پارامتری ندارد، داخل پرانتز جلو نام آن را خالی می‌گذاریم یا void قرار می‌دهیم.

مثال: double test () یا doَuble test(void)

- دقت کنید که در انتهای دستور و بعد از پرانتز بسته سمی کالن نمی‌آید.
- برای عبارت «برگردان» از دستور return استفاده می‌کنیم. اگر مقداری باید برگشت داده شود آن مقدار با یک فاصله بعد از دستور return قرار می‌گیرد و سپس سمی کالن قرار می‌گیرد.

مثال: return ; یا return sum;

- تابع حداکثر یک مقدار برمی‌گرداند.

تابع

بیان زیرالگوریتم در C++:

- معرفی زیرالگوریتم به صورت زیر است:

(... ,نام پارامتر دوم نوع پارامتر دوم ,نام پارامتر اول نوع پارامتر اول)نام تابع نوع مقدار بازگشتی

- مثال: int IsPrime(int n) double BMM(int m,float n, int constant)
- اگر زیرالگوریتمی مقدار بازگشتی ندارد، نوع مقدار بازگشتی را void قرار می‌دهیم. مثال:

void print (int m,int n)

- اگر زیرالگوریتم هیچ پارامتری ندارد، داخل پرانتز جلو نام آن را خالی می‌گذاریم یا void قرار می‌دهیم.

مثال: double test () یا doَuble test(void)

- دقت کنید که در انتهای دستور و بعد از پرانتز بسته سمی کالن نمی‌آید.
- برای عبارت «برگردان» از دستور return استفاده می‌کنیم. اگر مقداری باید برگشت داده شود آن مقدار با یک فاصله بعد از دستور return قرار می‌گیرد و سپس سمی کالن قرار می‌گیرد.

مثال: return ; یا return sum;

- تابع حداکثر یک مقدار برمی‌گرداند.

تابع

بیان زیرالگوریتم در C++:

- معرفی زیرالگوریتم به صورت زیر است:

(... ,نام پارامتر دوم نوع پارامتر دوم ,نام پارامتر اول نوع پارامتر اول)نام تابع نوع مقدار بازگشتی

- مثال: int IsPrime(int n) double BMM(int m,float n, int constant)
- اگر زیرالگوریتمی مقدار بازگشتی ندارد، نوع مقدار بازگشتی را void قرار می‌دهیم. مثال:

void print (int m,int n)

- اگر زیرالگوریتم هیچ پارامتری ندارد، داخل پرانتز جلو نام آن را خالی می‌گذاریم یا void قرار می‌دهیم.

مثال: double test () یا doَuble test(void)

- دقت کنید که در انتهای دستور و بعد از پرانتز بسته سمی کالن نمی‌آید.
- برای عبارت «برگردان» از دستور return استفاده می‌کنیم. اگر مقداری باید برگشت داده شود آن مقدار با یک فاصله بعد از دستور return قرار می‌گیرد و سپس سمی کالن قرار می‌گیرد.

مثال: return ; یا return sum;

- تابع حداکثر یک مقدار برمی‌گرداند.

تابع

بیان زیرالگوریتم در C++:

- معرفی زیرالگوریتم به صورت زیر است:

(... ,نام پارامتر دوم نوع پارامتر دوم ,نام پارامتر اول نوع پارامتر اول)نام تابع نوع مقدار بازگشتی

- مثال: int IsPrime(int n) double BMM(int m,float n, int constant)
- اگر زیرالگوریتمی مقدار بازگشتی ندارد، نوع مقدار بازگشتی را void قرار می‌دهیم. مثال:

void print (int m,int n)

- اگر زیرالگوریتم هیچ پارامتری ندارد، داخل پرانتز جلو نام آن را خالی می‌گذاریم یا void قرار می‌دهیم.

مثال: double test () یا doَuble test(void)

- دقت کنید که در انتهای دستور و بعد از پرانتز بسته سمی کالن نمی‌آید.
- برای عبارت «برگردان» از دستور return استفاده می‌کنیم. اگر مقداری باید برگشت داده شود آن مقدار با یک فاصله بعد از دستور return قرار می‌گیرد و سپس سمی کالن قرار می‌گیرد.

مثال: return ; یا return sum;

- تابع حداکثر یک مقدار برمی‌گرداند.

تابع

فراخوانی تابع:

- در بحث زیرالگوریتم، ما یک زیرالگوریتم را با نام آن به همراه لیست آرگومان‌های مورد نظرمان فراخوانی می‌کردیم و اگر زیرالگوریتم، مقدار بازگشت داده شده داشت، آن را در متغیری ذخیره می‌کردیم.
- در زیرفلوچارت نیز همین روند با قرار گرفتن موارد ذکر شده در داخل یک مستطیل با اضلاع عمودی دوتایی بیان می‌شد.
- در فراخوانی صرفاً نام تابع و لیست آرگومان‌ها کافی است و نوع مقدار بازگشتی تابع و نوع آرگومان‌ها بیان نمی‌شود.
- نوع و تعداد آرگومان‌های تابع در فراخوانی دقیقاً مشابه نوع و تعداد پارامترهای آن در تعریف تابع باشد. در صورت تفاوت در تعداد پارامترها، برنامه با خطای گرامری در زمان ترجمه یا خطای منطقی مواجه می‌شود.
- مثال:

```
print (3,x);          R=IsPrime(i);
```

تابع

فراخوانی تابع:

- در بحث زیرالگوریتم، ما یک زیرالگوریتم را با نام آن به همراه لیست آرگومان‌های مورد نظرمان فراخوانی می‌کردیم و اگر زیرالگوریتم، مقدار بازگشت داده شده داشت، آن را در متغیری ذخیره می‌کردیم.
- در زیرفلوچارت نیز همین روند با قرار گرفتن موارد ذکر شده در داخل یک مستطیل با اضلاع عمودی دوتایی بیان می‌شد.
- در فراخوانی صرفاً نام تابع و لیست آرگومان‌ها کافی است و نوع مقدار بازگشتی تابع و نوع آرگومان‌ها بیان نمی‌شود.
- نوع و تعداد آرگومان‌های تابع در فراخوانی دقیقاً مشابه نوع و تعداد پارامترهای آن در تعریف تابع باشد. در صورت تفاوت در تعداد پارامترها، برنامه با خطای گرامری در زمان ترجمه یا خطای منطقی مواجه می‌شود.
- مثال:

```
print (3,x);          R=IsPrime(i);
```

تابع

فراخوانی تابع:

- در بحث زیرالگوریتم، ما یک زیرالگوریتم را با نام آن به همراه لیست آرگومان‌های مورد نظرمان فراخوانی می‌کردیم و اگر زیرالگوریتم، مقدار بازگشت داده شده داشت، آن را در متغیری ذخیره می‌کردیم.
- در زیرفلوچارت نیز همین روند با قرار گرفتن موارد ذکر شده در داخل یک مستطیل با اضلاع عمودی دوتایی بیان می‌شد.
- در فراخوانی صرفاً نام تابع و لیست آرگومان‌ها کافی است و نوع مقدار بازگشتی تابع و نوع آرگومان‌ها بیان نمی‌شود.
- نوع و تعداد آرگومان‌های تابع در فراخوانی دقیقاً مشابه نوع و تعداد پارامترهای آن در تعریف تابع باشد. در صورت تفاوت در تعداد پارامترها، برنامه با خطای گرامری در زمان ترجمه یا خطای منطقی مواجه می‌شود.
- مثال:

```
print (3,x);          R=IsPrime(i);
```

تابع

فراخوانی تابع:

- در بحث زیرالگوریتم، ما یک زیرالگوریتم را با نام آن به همراه لیست آرگومان‌های مورد نظرمان فراخوانی می‌کردیم و اگر زیرالگوریتم، مقدار بازگشت داده شده داشت، آن را در متغیری ذخیره می‌کردیم.
- در زیرفلوچارت نیز همین روند با قرار گرفتن موارد ذکر شده در داخل یک مستطیل با اضلاع عمودی دوتایی بیان می‌شد.
- در فراخوانی صرفاً نام تابع و لیست آرگومان‌ها کافی است و نوع مقدار بازگشتی تابع و نوع آرگومان‌ها بیان نمی‌شود.
- نوع و تعداد آرگومان‌های تابع در فراخوانی دقیقاً مشابه نوع و تعداد پارامترهای آن در تعریف تابع باشد. در صورت تفاوت در تعداد پارامترها، برنامه با خطای گرامری در زمان ترجمه یا خطای منطقی مواجه می‌شود.
- مثال:

```
print (3,x);          R=IsPrime(i);
```

تابع

فراخوانی تابع:

- در بحث زیرالگوریتم، ما یک زیرالگوریتم را با نام آن به همراه لیست آرگومان‌های مورد نظرمان فراخوانی می‌کردیم و اگر زیرالگوریتم، مقدار بازگشت داده شده داشت، آن را در متغیری ذخیره می‌کردیم.
- در زیرفلوچارت نیز همین روند با قرار گرفتن موارد ذکر شده در داخل یک مستطیل با اضلاع عمودی دوتایی بیان می‌شد.
- در فراخوانی صرفاً نام تابع و لیست آرگومان‌ها کافی است و نوع مقدار بازگشتی تابع و نوع آرگومان‌ها بیان نمی‌شود.
- نوع و تعداد آرگومان‌های تابع در فراخوانی دقیقاً مشابه نوع و تعداد پارامترهای آن در تعریف تابع باشد. در صورت تفاوت در تعداد پارامترها، برنامه با خطای گرامری در زمان ترجمه یا خطای منطقی مواجه می‌شود.
- مثال:

```
print (3,x);          R=IsPrime(i);
```

تابع

نکات مهم:

- توابع مربوط به زیرالگوریتم‌ها باید در همان فایل برنامه مربوط به الگوریتم موجود باشد.
- الگوریتم اصلی در داخل تابع main قرار می‌گیرد.
- با اجرای یک برنامه، فقط تابع main اجرا می‌شود و در صورتی توابع دیگر اجرا می‌شوند که دستور فراخوانی مربوط به آنها اجرا شود.
- ترتیب قرار گرفتن توابع در برنامه، تاثیری در اجرای برنامه ندارد ولی تابع باید قبل از اولین فراخوانی‌اش در ترتیب آمدن توابع در برنامه، آمده باشد، یا تعریف شده باشد.
- منظور از تعریف کردن تابع آن است که همان خط اول تعریف تابع که شامل نوع بازگشت داده شده توسط تابع، نام تابع و لیست پارامترها و نوع آنها بود به محلی که نیاز داریم تابع تعریف شود کپی شده و در انتها یک سمی کالن قرار دهیم. این کار، هیچ تاثیر اجرایی ندارد. مثال:

```
double BMM(int m,float n,int  constant);
```

تابع

نکات مهم:

- توابع مربوط به زیرالگوریتم‌ها باید در همان فایل برنامه مربوط به الگوریتم موجود باشد.
- الگوریتم اصلی در داخل تابع main قرار می‌گیرد.
- با اجرای یک برنامه، فقط تابع main اجرا می‌شود و در صورتی توابع دیگر اجرا می‌شوند که دستور فراخوانی مربوط به آنها اجرا شود.
- ترتیب قرار گرفتن توابع در برنامه، تاثیری در اجرای برنامه ندارد ولی تابع باید قبل از اولین فراخوانی‌اش در ترتیب آمدن توابع در برنامه، آمده باشد، یا تعریف شده باشد.
- منظور از تعریف کردن تابع آن است که همان خط اول تعریف تابع که شامل نوع بازگشت داده شده توسط تابع، نام تابع و لیست پارامترها و نوع آنها بود به محلی که نیاز داریم تابع تعریف شود کپی شده و در انتها یک سمی کالن قرار دهیم. این کار، هیچ تاثیر اجرایی ندارد. مثال:

```
double BMM(int m,float n,int  constant);
```

تابع

نکات مهم:

- توابع مربوط به زیرالگوریتم‌ها باید در همان فایل برنامه مربوط به الگوریتم موجود باشد.
- الگوریتم اصلی در داخل تابع main قرار می‌گیرد.
- با اجرای یک برنامه، فقط تابع main اجرا می‌شود و در صورتی توابع دیگر اجرا می‌شوند که دستور فراخوانی مربوط به آنها اجرا شود.
- ترتیب قرار گرفتن توابع در برنامه، تاثیری در اجرای برنامه ندارد ولی تابع باید قبل از اولین فراخوانی‌اش در ترتیب آمدن توابع در برنامه، آمده باشد، یا تعریف شده باشد.
- منظور از تعریف کردن تابع آن است که همان خط اول تعریف تابع که شامل نوع بازگشت داده شده توسط تابع، نام تابع و لیست پارامترها و نوع آنها بود به محلی که نیاز داریم تابع تعریف شود کپی شده و در انتها یک سمی کالن قرار دهیم. این کار، هیچ تاثیر اجرایی ندارد. مثال:

```
double BMM(int m,float n,int  constant);
```

تابع

نکات مهم:

- توابع مربوط به زیرالگوریتم‌ها باید در همان فایل برنامه مربوط به الگوریتم موجود باشد.
- الگوریتم اصلی در داخل تابع main قرار می‌گیرد.
- با اجرای یک برنامه، **فقط** تابع main اجرا می‌شود و در صورتی توابع دیگر اجرا می‌شوند که دستور فراخوانی مربوط به آنها اجرا شود.
- ترتیب قرار گرفتن توابع در برنامه، تاثیری در اجرای برنامه ندارد ولی تابع باید قبل از اولین فراخوانی‌اش در ترتیب آمدن توابع در برنامه، آمده باشد، یا تعریف شده باشد.
- منظور از تعریف کردن تابع آن است که همان خط اول تعریف تابع که شامل نوع بازگشت داده شده توسط تابع، نام تابع و لیست پارامترها و نوع آنها بود به محلی که نیاز داریم تابع تعریف شود کپی شده و در انتها یک سمی کالن قرار دهیم. این کار، هیچ تاثیر اجرایی ندارد. مثال:

```
double BMM(int m,float n,int  constant);
```

تابع

نکات مهم:

- توابع مربوط به زیرالگوریتم‌ها باید در همان فایل برنامه مربوط به الگوریتم موجود باشد.
- الگوریتم اصلی در داخل تابع main قرار می‌گیرد.
- با اجرای یک برنامه، **فقط** تابع main اجرا می‌شود و در صورتی توابع دیگر اجرا می‌شوند که دستور فراخوانی مربوط به آنها اجرا شود.
- ترتیب قرار گرفتن توابع در برنامه، تاثیری در اجرای برنامه ندارد ولی تابع باید قبل از اولین فراخوانی‌اش در ترتیب آمدن توابع در برنامه، آمده باشد، یا تعریف شده باشد.
- منظور از تعریف کردن تابع آن است که همان خط اول تعریف تابع که شامل نوع بازگشت داده شده توسط تابع، نام تابع و لیست پارامترها و نوع آنها بود به محلی که نیاز داریم تابع تعریف شود کپی شده و در انتها یک سمی کالن قرار دهیم. این کار، هیچ تاثیر اجرایی ندارد. مثال:

```
double BMM(int m,float n,int  constant);
```

مثال ۱

زیرالگوریتمی بنویسید که عدد n را به عنوان پارامتر دریافت کند و در صورت اول بودن n، عدد ۱ و در غیر این صورت عدد ۰ را برگرداند.

الگوریتم:

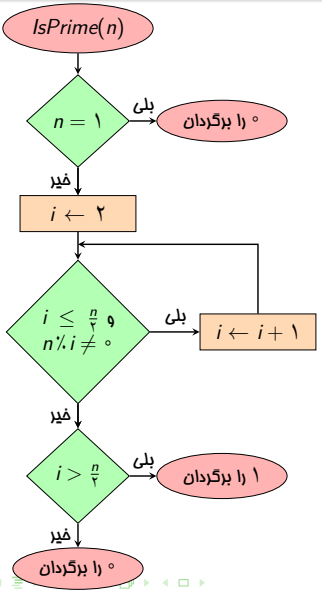
```
۱  شروع زیرالگوریتم IsPrime(n)
۲  اگر n = ۱ آنگاه
۳  |  ۰ را برگردان
۴  پایان اگر
۵  i ← ۲
۶  اگر i ≤ n/۲ و i ≠ ۰ ٪ n آنگاه
۷  |  i ← i + ۱
۸  |  به مرحلهٔ ۳ برو
۹  پایان اگر
۱۰ اگر i > n/۲ آنگاه
۱۱ |  ۱ را برگردان
۱۲ درغیراینصورت
۱۳ |  ۰ را برگردان
۱۴ پایان اگر
```

مثال ۱

زیرالگوریتمی بنویسید که عدد n را به عنوان پارامتر دریافت کند و در صورت اول بودن n، عدد ۱ و در غیر این صورت عدد ۰ را برگرداند.

الگوریتم:

```
۱  شروع زیرالگوریتم IsPrime(n)
۲  اگر n = ۱ آنگاه
۳  |  ۰ را برگردان
۴  پایان اگر
۵  i ← ۲
۶  اگر i ≤ n/۲ و n ٪ i ≠ ۰ آنگاه
۷  |  i ← i + ۱
۸  |  به مرحلهٔ ۳ برو
۹  پایان اگر
۱۰ اگر i > n/۲ آنگاه
۱۱ |  ۱ را برگردان
۱۲ درغیراینصورت
۱۳ |  ۰ را برگردان
۱۴ پایان اگر
```



مثال ۱

زیرالگوریتمی بنویسید که عدد n را به عنوان پارامتر دریافت کند و در صورت اول بودن n، عدد ۱ و در غیر این صورت عدد ۰ را برگرداند.

الگوریتم:

۱ شروع زیرالگوریتم $IsPrime(n)$
۲ اگر $n = ۱$ آنگاه
۳ ۰ را برگردان
۴ پایان اگر
۵ $i \leftarrow ۲$
۶ اگر $i \leq \frac{n}{۲}$ و $i \neq ۰$ و $n \,\%\, i$ آنگاه
۷ $i \leftarrow i + ۱$
۸ به مرحلهٔ ۳ برو
۹ پایان اگر
۱۰ اگر $i > \frac{n}{۲}$ آنگاه
۱۱ ۱ را برگردان
۱۲ درغیراینصورت
۱۳ ۰ را برگردان
۱۴ پایان اگر

```
1  int  IsPrime(unsigned int n)
2  {
3   if (n == 1)
4    return  0;
5   int  i  = 2;
6   while(i<=n/2 && n%i !=0)
7           i++;
8   if (i> n/2)
9           return  1;
10     else
11          return  0;
12 }
```

مثال ۲

الگوریتمی بنویسید که تعداد اعداد اول بین ۲ و ۱۰۰۰ را محاسبه و چاپ کند.

الگوریتم:

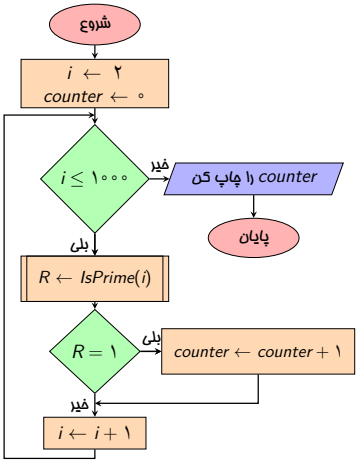
۱ شروع
۲ $i \leftarrow ۲$، $counter \leftarrow ۰$
۳ اگر $i \leq ۱۰۰۰$ آنگاه
۴ $R \leftarrow \text{IsPrime}(i)$
۵ اگر $R = ۱$ آنگاه
۶ $counter \leftarrow counter + ۱$
۷ پایان اگر
۸ $i \leftarrow i + ۱$
۹ به مرحلهٔ ۳ برو
۱۰ پایان اگر
۱۱ $counter$ را چاپ کن
۱۲ پایان

مثال ۲

الگوریتمی بنویسید که تعداد اعداد اول بین ۲ و ۱۰۰۰ را محاسبه و چاپ کند.

الگوریتم:

۱ شروع
۲ $i \leftarrow ۲$، $counter \leftarrow ۰$
۳ اگر $i \leq ۱۰۰۰$ آنگاه
۴ $R \leftarrow \text{IsPrime}(i)$
۵ اگر $R = ۱$ آنگاه
۶ $counter \leftarrow counter + ۱$
۷ پایان اگر
۸ $i \leftarrow i + ۱$
۹ به مرحلهٔ ۳ برو
۱۰ پایان اگر
۱۱ $counter$ را چاپ کن
۱۲ پایان



```
#include <iostream>
using namespace std;
int IsPrime(unsigned int n)
{
  int i = 2;
  if (n == 1)
        return 0;
  while(i<=n/2 && n%i !=0)
        i++;
  if (i> n/2)
        return 1;
    else
        return 0;
}
int main()
{
 unsigned int i,counter=0,R;
 for(i = 2; i <= 1000; i++)
    {
        R=IsPrime(i);
        if (R == 1)
            counter++;
    }
 cout << "# of Primes:" <<counter;
 return 0;
}
```

مثال ۲

الگوریتمی بنویسید که تعداد اعداد اول بین ۲ و ۱۰۰۰ را محاسبه و چاپ کند.

الگوریتم:

۱ شروع
۲ $i \leftarrow ۲$، $counter \leftarrow ۰$
۳ اگر $i \leq ۱۰۰۰$ آنگاه
۴ $R \leftarrow \text{IsPrime}(i)$
۵ اگر $R = ۱$ آنگاه
۶ $counter \leftarrow counter + ۱$
۷ پایان اگر
۸ $i \leftarrow i + ۱$
۹ به مرحلهٔ ۳ برو
۱۰ پایان اگر
۱۱ $counter$ را چاپ کن
۱۲ پایان

مثال ۳

الگوریتمی بنویسید که یک عدد طبیعی را از ورودی بگیرد و تشخیص دهد اول است یا خیر.

الگوریتم:

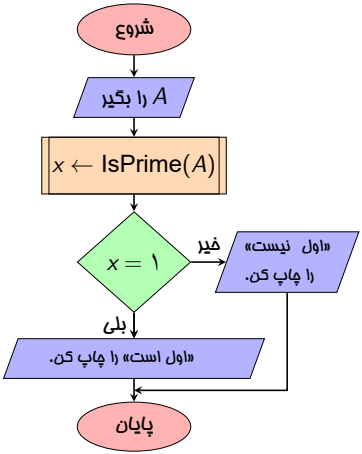
۱ شروع
۲ A را بگیر
۳ $x \leftarrow \text{IsPrime}(A)$
۴ اگر $x = ۱$ آنگاه
۵ | «اول است» را چاپ کن
۶ درغیراینصورت
۷ | «اول نیست» را چاپ کن
۸ پایان اگر
۹ پایان

مثال ۳

الگوریتمی بنویسید که یک عدد طبیعی را از ورودی بگیرد و تشخیص دهد اول است یا خیر.

الگوریتم:

۱ شروع
۲ A را بگیر
۳ $x \leftarrow \text{IsPrime}(A)$
۴ اگر $x = ۱$ آنگاه
۵ | «اول است» را چاپ کن
۶ درغیراینصورت
۷ | «اول نیست» را چاپ کن
۸ پایان اگر
۹ پایان



```
1  #include <iostream>
2  using namespace std;
3   int  IsPrime(unsigned int n)
4  {
5       int  i  = 2;
6    if (n ==1)
7     return  0;
8       while(i<=n/2 && n%i !=0)
9            i++;
10       if (i> n/2)
11            return  1;
12       else
13            return  0;
14 }
15  int  main()
16 {
17       unsigned int  A,x;
18       cout << "Please enter a positive  integer :";
19       cin  >> A;
20       x  = IsPrime(A);
21        if (x  == 1)
22            cout<<"Is Prime.";
23       else
24            cout<<"Not Prime.";
25       return  0;
26 }
```

مثال ۳

الگوریتمی بنویسید که یک عدد طبیعی را از ورودی بگیرد و تشخیص دهد اول است یا خیر.

الگوریتم:

۱ شروع
۲ A را بگیر
۳ $x \leftarrow \text{IsPrime}(A)$
۴ اگر $x = ۱$ آنگاه
۵ | «اول است» را چاپ کن
۶ درغیراینصورت
۷ | «اول نیست» را چاپ کن
۸ پایان اگر
۹ پایان

```
1  #include <iostream>
2  using namespace std;
3   int  IsPrime(unsigned int n)
4  {
5       int  i  = 2;
6    if (n ==1)
7     return  0;
8       while(i<=n/2 && n%i !=0)
9            i++;
10       if (i> n/2)
11            return  1;
12       else
13            return  0;
14 }
15  int  main()
16 {
17       unsigned int  A,x;
18       cout << "Please enter a positive  integer :";
19       cin  >> A;
20       x  = IsPrime(A);
21        if (x  == 1)
22            cout<<"Is Prime.";
23       else
24            cout<<"Not Prime.";
25       return  0;
26 }
```

مثال ۳

الگوریتمی بنویسید که یک عدد طبیعی را از ورودی بگیرد و تشخیص دهد اول است یا خیر.

الگوریتم:

۱ شروع
۲ A را بگیر
۳ $x \leftarrow \text{IsPrime}(A)$
۴ اگر $x = ۱$ آنگاه
۵ | «اول است» را چاپ کن
۶ درغیراینصورت
۷ | «اول نیست» را چاپ کن
۸ پایان اگر
۹ پایان

```
1  #include <iostream>
2  using namespace std;
3  int IsPrime(unsigned int n)
4  {
5      int i = 2;
6    if (n ==1)
7     return 0;
8      while(i<=n/2 && n%i !=0)
9          i++;
10     if (i> n/2)
11         return 1;
12     else
13         return 0;
14 }
15 int main()
16 {
17     unsigned int A,x;
18     cout << "Please enter a positive integer :";
19     cin >> A;
20     x = IsPrime(A);
21     if (x == 1)
22         cout<<"Is Prime.";
23     else
24         cout<<"Not Prime.";
25     return 0;
26 }
```

مثال ۳

الگوریتمی بنویسید که یک عدد طبیعی را از ورودی بگیرد و تشخیص دهد اول است یا خیر.

الگوریتم:

۱ شروع
۲ A را بگیر
۳ $x \leftarrow \text{IsPrime}(A)$
۴ اگر $x = ۱$ آنگاه
۵ | «اول است» را چاپ کن
۶ درغیراینصورت
۷ | «اول نیست» را چاپ کن
۸ پایان اگر
۹ پایان

استفاده مجدد!

مثال ۴

الگوریتم و فلوچارتی ارائه کنید که یک عدد طبیعی را بگیرد و تجزیهٔ آن را به عوامل اول محاسبه و چاپ کند.

الگوریتم:

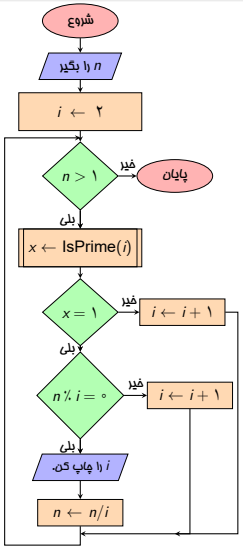
۱ شروع
۲ n را بگیر
۳ $i \leftarrow ۲$
۴ اگر $n > ۱$ آنگاه
۵ $x \leftarrow \text{IsPrime}(i)$
۶ اگر $x = ۱$ آنگاه
۷ اگر $n \,\%\, i = ۰$ آنگاه
۸ i را چاپ کن
۹ $n \leftarrow n/i$
۱۰ درغیراینصورت
۱۱ $i \leftarrow i + ۱$
۱۲ پایان اگر
۱۳ درغیراینصورت
۱۴ $i \leftarrow i + ۱$
۱۵ پایان اگر
۱۶ به مرحلهٔ ۴ برو
۱۷ پایان اگر
۱۸ پایان

مثال ۴

الگوریتم و فلوچارتی ارائه کنید که یک عدد طبیعی را بگیرد و تجزیهٔ آن را به عوامل اول محاسبه و چاپ کند.

الگوریتم:

```
۱  شروع
۲  n را بگیر
۳  i ← ۲
۴  اگر n > ۱ آنگاه
۵  |  x ← IsPrime(i)
۶  |  اگر x = ۱ آنگاه
۷  |  |  اگر n ٪ i = ۰ آنگاه
۸  |  |  |  i را چاپ کن
۹  |  |  |  n ← n/i
۱۰ |  |  درغیراینصورت
۱۱ |  |  |  i ← i + ۱
۱۲ |  |  پایان اگر
۱۳ |  درغیراینصورت
۱۴ |  |  i ← i + ۱
۱۵ |  پایان اگر
۱۶ |  به مرحلهٔ ۴ برو
۱۷ پایان اگر
۱۸ پایان
```



مثال ۴

الگوریتم و فلوچارتی ارائه کنید که یک عدد طبیعی را بگیرد و تجزیهٔ آن را به عوامل اول محاسبه و چاپ کند.

الگوریتم:

۱ شروع
۲ n را بگیر
۳ $i \leftarrow ۲$
۴ اگر $n > ۱$ آنگاه
۵ $x \leftarrow \text{IsPrime}(i)$
۶ اگر $x = ۱$ آنگاه
۷ اگر $n \,\%\, i = ۰$ آنگاه
۸ i را چاپ کن
۹ $n \leftarrow n/i$
۱۰ درغیراینصورت
۱۱ $i \leftarrow i + ۱$
۱۲ پایان اگر
۱۳ درغیراینصورت
۱۴ $i \leftarrow i + ۱$
۱۵ پایان اگر
۱۶ به مرحلهٔ ۴ برو
۱۷ پایان اگر
۱۸ پایان

```
1  #include <iostream>
2  using namespace std;
3  int IsPrime(unsigned int n)
4  {
5    int i = 2;
6    while(i<=n/2 && n%i !=0)
7           i++;
8    if (i> n/2)
9           return 1;
10     else
11          return 0;
12 }
13 int main()
14 {
15     unsigned int n,i,x;
16     cout << "Please enter a positive integer :";
17     cin >> n;
18     for(i=2; n>1; )
19     {
20         x = IsPrime(i);
21         if (x == 1)
22             if (n%i == 0)
23             {
24                 cout<<i<<"*";
25                 n/=i;
26             }
27             else
28                 i++;
29         else
30             i++;
31     }
32     return 0;
33 }
```

مثال ۵

الگوریتم و فلوچارتی ارائه کنید که صورت و مخرج دو کسر را بگیرد و مجموع آن دو کسر را محاسبه و به صورت کسر تا حد امکان ساده شده چاپ کند.

الگوریتم:

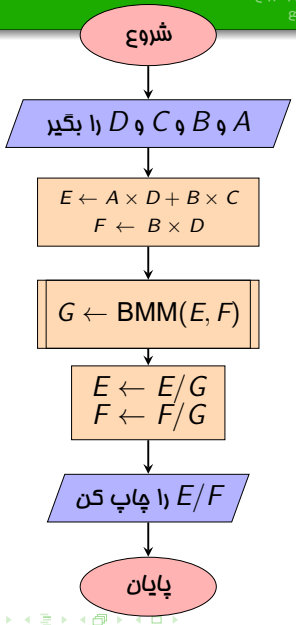
۱ شروع
۲ A و B و C و D را بگیر
۳ $E \leftarrow A \times D + B \times C$
۴ $F \leftarrow B \times D$
۵ $G \leftarrow \mathrm{BMM}(E, F)$
۶ $E \leftarrow E/G$ ،$F \leftarrow F/G$
۷ E/F را چاپ کن
۸ پایان

مثال ۵

الگوریتم و فلوچارتی ارائه کنید که صورت و مخرج دو کسر را بگیرد و مجموع آن دو کسر را محاسبه و به صورت کسر تا حد امکان ساده شده چاپ کند.

الگوریتم:

۱ شروع
۲ A و B و C و D را بگیر
۳ $E \leftarrow A \times D + B \times C$
۴ $F \leftarrow B \times D$
۵ $G \leftarrow \text{BMM}(E, F)$
۶ $E \leftarrow E/G$ ،$F \leftarrow F/G$
۷ E/F را چاپ کن
۸ پایان



مثال ۵

الگوریتم و فلوچارتی ارائه کنید که صورت و مخرج دو کسر را بگیرد و مجموع آن دو کسر را محاسبه و به صورت کسر تا حد امکان ساده شده چاپ کند.

الگوریتم:

۱ شروع
۲ A و B و C و D را بگیر
۳ $E \leftarrow A \times D + B \times C$
۴ $F \leftarrow B \times D$
۵ $G \leftarrow \mathrm{BMM}(E, F)$
۶ $F \leftarrow F/G$ ،$E \leftarrow E/G$
۷ E/F را چاپ کن
۸ پایان

```
1  #include <iostream>
2  using namespace std;
3  unsigned int BMM(unsigned int A, unsigned int B)
4  {
5      unsigned int i;
6      if (A > B)
7          i=B;
8      else
9          i=A;
10     while ( A % i != 0 || B % i !=0)
11         i--;
12     return i;
13 }
14 int main()
15 {
16     int A,B,C,D,E,F,G;
17     cout << "I want to compute A / B + C / D.\n";
18     cin>>A>>B>>C>>D;
19     E=A*D+B*C; F=B*D;
20     G=BMM(E,F);
21     E/=G; F/=G;
22     cout << "The result:" << E << "/" << F;
23     return 0;
24 }
```

مثال ۶

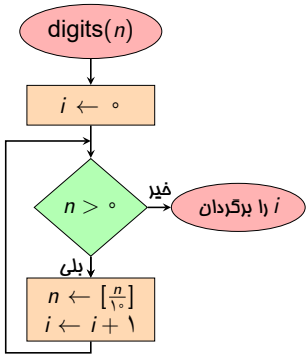
الگوریتم و فلوچارتی ارائه کنید که ۴ عدد طبیعی را بگیرد و عدد حاصل از قرار دادن آن ۴ عدد در کنار یکدیگر را محاسبه و چاپ کند.

الگوریتم:

۱ شروع زیرالگوریتم $\text{digits}(n)$
۲ $i \leftarrow ۰$
۳ اگر $n > ۰$ آنگاه
۴ $\quad n \leftarrow \left[\frac{n}{۱۰}\right]$
۵ $\quad i \leftarrow i + ۱$
۶ $\quad$ به مرحلهٔ ۳ برو
۷ پایان اگر
۸ i را برگردان

مثال ۶

الگوریتم و فلوچارتی ارائه کنید که ۴ عدد طبیعی را بگیرد و عدد حاصل از قرار دادن آن ۴ عدد در کنار یکدیگر را محاسبه و چاپ کند.



الگوریتم:

۱ شروع زیرالگوریتم digits(n)
۲ $i \leftarrow ۰$
۳ اگر $n > ۰$ آنگاه
۴ $\quad n \leftarrow \left[\frac{n}{۱۰}\right]$
۵ $\quad i \leftarrow i + ۱$
۶ $\quad$ به مرحلهٔ ۳ برو
۷ پایان اگر
۸ i را برگردان

مثال ۶

الگوریتم و فلوچارتی ارائه کنید که ۴ عدد طبیعی را بگیرد و عدد حاصل از قرار دادن آن ۴ عدد در کنار یکدیگر را محاسبه و چاپ کند.

الگوریتم:

۱ شروع زیرالگوریتم digits(n)
۲ $i \leftarrow ۰$
۳ اگر $n > ۰$ آنگاه
۴ $\quad n \leftarrow \left[\frac{n}{۱۰}\right]$
۵ $\quad i \leftarrow i + ۱$
۶ $\quad$ به مرحلهٔ ۳ برو
۷ پایان اگر
۸ i را برگردان

```
1  unsigned int  digits (unsigned long int  n)
2  {
3      unsigned int  i ;
4      for ( i=0; n>0; i++)
5          n/=10;
6      return  i ;
7  }
```

مثال ۷

الگوریتم و فلوچارتی ارائه کنید که ۴ عدد طبیعی را بگیرد و عدد حاصل از قرار دادن آن ۴ عدد در کنار یکدیگر را محاسبه و چاپ کند.

الگوریتم:

۱ **شروع** زیرالگوریتم $concat2(x, y)$

۲ $i \leftarrow \text{digits}(x)$

۳ $A \leftarrow y \times ۱۰^{i} + x$

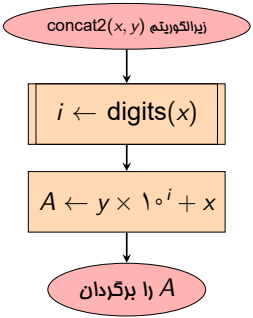
۴ A را برگردان

مثال ۷

الگوریتم و فلوچارتی ارائه کنید که ۴ عدد طبیعی را بگیرد و عدد حاصل از قرار دادن آن ۴ عدد در کنار یکدیگر را محاسبه و چاپ کند.

الگوریتم:

۱ **شروع** زیرالگوریتم $concat2(x, y)$
۲ $i \leftarrow \text{digits}(x)$
۳ $A \leftarrow y \times ۱۰^{i} + x$
۴ A را برگردان



```
1  long int pow(int base, int power)
2  {
3      long int p=1;
4      int i;
5      for(i=0; i<power; i++)
6          p*=base;
7      return p;
8  }
9  int digits (long int n)
10 {
11     int i;
12     for(i=0; n>0; i++)
13         n/=10;
14     return i;
15 }
16 long int concat2(long int x,long int y)
17 {
18     long int A;
19     int i;
20     i=digits(x);
21     A=y*pow(10,i)+x;
22     return A;
23 }
```

مثال ۷

الگوریتم و فلوچارتی ارائه کنید که ۴ عدد طبیعی را بگیرد و عدد حاصل از قرار دادن آن ۴ عدد در کنار یکدیگر را محاسبه و چاپ کند.

الگوریتم:

۱ **شروع** زیرالگوریتم $concat2(x, y)$

۲ $i \leftarrow \text{digits}(x)$

۳ $A \leftarrow y \times ۱۰^{i} + x$

۴ A را برگردان

مثال ۸

الگوریتم و فلوچارتی ارائه کنید که ۴ عدد طبیعی را بگیرد و عدد حاصل از قرار دادن آن ۴ عدد در کنار یکدیگر را محاسبه و چاپ کند.

الگوریتم:

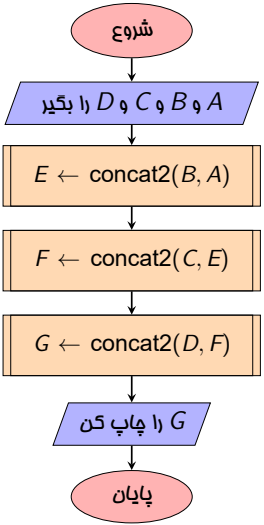
۱ شروع
۲ A و B و C و D را بگیر
۳ $E \leftarrow \text{concat2}(B, A)$
۴ $F \leftarrow \text{concat2}(C, E)$
۵ $G \leftarrow \text{concat2}(D, F)$
۶ G را چاپ کن
۷ پایان

مثال ۸

الگوریتم و فلوچارتی ارائه کنید که ۴ عدد طبیعی را بگیرد و عدد حاصل از قرار دادن آن ۴ عدد در کنار یکدیگر را محاسبه و چاپ کند.

الگوریتم:

۱ شروع
۲ A و B و C و D را بگیر
۳ $E \leftarrow \text{concat2}(B, A)$
۴ $F \leftarrow \text{concat2}(C, E)$
۵ $G \leftarrow \text{concat2}(D, F)$
۶ G را چاپ کن
۷ پایان



```
1  #include <iostream>
2  using namespace std;
3  long int pow(int base,int power)
4  {
5   ....
6  }
7  int digits (long int n)
8  {
9   ....
10 }
11 long int concat2(long int x,long int y)
12 {
13  .....
14 }
15 int main()
16 {
17    long int A,B,C,D,E,F,G;
18    cout << "Give me four numbers: ";
19    cin >> A >> B >> C >> D;
20    E=concat2(B,A);
21    F=concat2(C,E);
22    G=concat2(D,F);
23    cout << "The result= " << G;
24    return 0;
25 }
```

مثال ۸

الگوریتم و فلوچارتی ارائه کنید که ۴ عدد طبیعی را بگیرد و عدد حاصل از قرار دادن آن ۴ عدد در کنار یکدیگر را محاسبه و چاپ کند.

الگوریتم:

۱ شروع
۲ A و B و C و D را بگیر
۳ $E \leftarrow \text{concat2}(B, A)$
۴ $F \leftarrow \text{concat2}(C, E)$
۵ $G \leftarrow \text{concat2}(D, F)$
۶ G را چاپ کن
۷ پایان

فراخوانی با مقدار، فراخوانی با ارجاع

فراخوانی با مقدار:

- فرض کنید تابعی به صورت int func(int x,double y) داریم و در جای دیگری، این تابع با دستور func(a,b); فراخوانی شده است. در این فراخوانی مراحل زیر اتفاق می‌افتد:
  - مقدار متغیر a در متغیر x و مقدار متغیر b در y کپی می‌شود.
  - دستورات تابع func یکی پس از دیگری اجرا می‌شود تا اجرای تابع به پایان برسد یا در روند اجرای تابع به دستور return برسد. در اینجا اجرای تابع پایان می‌یابد و در صورتی که تابع، مقداری را بازگشت می‌دهد، این مقدار به تابع فراخوانده برگشت داده می‌شود.
  - این شکل فراخوانی تابع را فراخوانی با مقدار می‌نامند. دلیل این امر آن است که مقدار آرگومان‌ها به پارامترهای متناظر کپی می‌شود و سپس تابع اجرا می‌شود. حال اگر مقدار متغیرهای پارامتر (یعنی x و y) در تابع عوض شود، این مقدار تاثیری در متغیر متناظر در فراخوانی تابع (یعنی a و b) ندارد.

فراخوانی با مقدار، فراخوانی با ارجاع

فراخوانی با مقدار:

- فرض کنید تابعی به صورت int func(int x,double y) داریم و در جای دیگری، این تابع با دستور func(a,b); فراخوانی شده است. در این فراخوانی مراحل زیر اتفاق می‌افتد:
- مقدار متغیر a در متغیر x و مقدار متغیر b در y کپی می‌شود.
- دستورات تابع func یکی پس از دیگری اجرا می‌شود تا اجرای تابع به پایان برسد یا در روند اجرای تابع به دستور return برسد. در اینجا اجرای تابع پایان می‌یابد و در صورتی که تابع، مقداری را بازگشت می‌دهد، این مقدار به تابع فراخوانده برگشت داده می‌شود.
- این شکل فراخوانی تابع را فراخوانی با مقدار می‌نامند. دلیل این امر آن است که مقدار آرگومان‌ها به پارامترهای متناظر کپی می‌شود و سپس تابع اجرا می‌شود. حال اگر مقدار متغیرهای پارامتر (یعنی x و y) در تابع عوض شود، این مقدار تاثیری در متغیر متناظر در فراخوانی تابع (یعنی a و b) ندارد.

فراخوانی با مقدار، فراخوانی با ارجاع

فراخوانی با مقدار:

- فرض کنید تابعی به صورت int func(int x,double y) داریم و در جای دیگری، این تابع با دستور func(a,b); فراخوانی شده است. در این فراخوانی مراحل زیر اتفاق می‌افتد:
- مقدار متغیر a در متغیر x و مقدار متغیر b در y کپی می‌شود.
- دستورات تابع func یکی پس از دیگری اجرا می‌شود تا اجرای تابع به پایان برسد یا در روند اجرای تابع به دستور return برسد. در اینجا اجرای تابع پایان می‌یابد و در صورتی که تابع، مقداری را بازگشت می‌دهد، این مقدار به تابع فراخواننده برگشت داده می‌شود.
- این شکل فراخوانی تابع را فراخوانی با مقدار می‌نامند. دلیل این امر آن است که مقدار آرگومان‌ها به پارامترهای متناظر کپی می‌شود و سپس تابع اجرا می‌شود. حال اگر مقدار متغیرهای پارامتر (یعنی x و y) در تابع عوض شود، این مقدار تاثیری در متغیر متناظر در فراخوانی تابع (یعنی a و b) ندارد.

فراخوانی با مقدار، فراخوانی با ارجاع

فراخوانی با مقدار:

- فرض کنید تابعی به صورت int func(int x,double y) داریم و در جای دیگری، این تابع با دستور func(a,b); فراخوانی شده است. در این فراخوانی مراحل زیر اتفاق می‌افتد:
- مقدار متغیر a در متغیر x و مقدار متغیر b در y کپی می‌شود.
- دستورات تابع func یکی پس از دیگری اجرا می‌شود تا اجرای تابع به پایان برسد یا در روند اجرای تابع به دستور return برسد. در اینجا اجرای تابع پایان می‌یابد و در صورتی که تابع، مقداری را بازگشت می‌دهد، این مقدار به تابع فراخواننده برگشت داده می‌شود.
- این شکل فراخوانی تابع را فراخوانی با مقدار می‌نامند. دلیل این امر آن است که مقدار آرگومان‌ها به پارامترهای متناظر کپی می‌شود و سپس تابع اجرا می‌شود. حال اگر مقدار متغیرهای پارامتر (یعنی x و y) در تابع عوض شود، این مقدار تاثیری در متغیر متناظر در فراخوانی تابع (یعنی a و b) ندارد.

فراخوانی با مقدار، فراخوانی با ارجاع

فراخوانی با مقدار:

```
1  #include <iostream>
2  using namespace std;
3  void func(int x, double y)
4  {
5      x=x+y;
6      y=2*y;
7  }
8  int main()
9  {
10     int a=2;
11     double b=3.14;
12     func(a,b);
13     cout<<"\na="<<a<<", b="<<b;
14     return 0;
15 }
```

خروجی برنامه: a=2, b=3.14

فراخوانی با مقدار، فراخوانی با ارجاع

فراخوانی با مقدار:

```
1  #include <iostream>
2  using namespace std;
3  void func(int x, double y)
4  {
5      x=x+y;
6      y=2*y;
7  }
8  int main()
9  {
10     int a=2;
11     double b=3.14;
12     func(a,b);
13     cout<<"\na="<<a<<", b="<<b;
14     return 0;
15 }
```

خروجی برنامه: a=2, b=3.14

فراخوانی با مقدار، فراخوانی با ارجاع

فراخوانی با ارجاع:

- تغییر در پارامترهای تابع که به صورت ارجاعی تعریف شده‌اند در داخل تابع، روی آرگومان متناظر نیز اعمال می‌شود.
- این کار با قرار دادن علامت & در جلو پارامتر مربوطه در تعریف تابع، انجام می‌شود.
- از این کار، می‌توان برای برگرداندن بیش از یک مقدار از تابع هم استفاده کرد.

فراخوانی با مقدار، فراخوانی با ارجاع

فراخوانی با ارجاع:

- تغییر در پارامترهای تابع که به صورت ارجاعی تعریف شده‌اند در داخل تابع، روی آرگومان متناظر نیز اعمال می‌شود.
- این کار با قرار دادن علامت & در جلو پارامتر مربوطه در تعریف تابع، انجام می‌شود.
- از این کار، می‌توان برای برگرداندن بیش از یک مقدار از تابع هم استفاده کرد.

فراخوانی با مقدار، فراخوانی با ارجاع

فراخوانی با ارجاع:

- تغییر در پارامترهای تابع که به صورت ارجاعی تعریف شده‌اند در داخل تابع، روی آرگومان متناظر نیز اعمال می‌شود.
- این کار با قرار دادن علامت & در جلو پارامتر مربوطه در تعریف تابع، انجام می‌شود.
- از این کار، می‌توان برای برگرداندن بیش از یک مقدار از تابع هم استفاده کرد.

فراخوانی با مقدار، فراخوانی با ارجاع

فراخوانی با ارجاع:

```
1  #include <iostream>
2  using namespace std;
3  void func( int  &x, double y)
4  {
5       x=x+y;
6       y=2*y;
7  }
8  int  main()
9  {
10       int  a=2;
11       double b=3.14;
12       func(a,b);
13       cout<<"\na="<<a<<", b="<<b;
14       return  0;
15  }
```

خروجی برنامه: a=5, b=3.14

فراخوانی با مقدار، فراخوانی با ارجاع

فراخوانی با ارجاع:

```
1  #include <iostream>
2  using namespace std;
3  void func( int  &x, double y)
4  {
5       x=x+y;
6       y=2*y;
7  }
8  int  main()
9  {
10       int  a=2;
11       double b=3.14;
12       func(a,b);
13       cout<<"\na="<<a<<", b="<<b;
14       return  0;
15  }
```

خروجی برنامه: a=5, b=3.14

آرایه به عنوان آرگومان تابع

آرایه به عنوان آرگومان تابع:

- انتقال آرایه‌ها به توابع نیز مشابه سایر متغیرها است. تنها تفاوت این است که در تعریف پارامترهای تابع، باید آرایه‌ها نیز مشابه تعریف آرایه‌ها، یعنی با استفاده از کروشه و مشخص کردن اندازهٔ آنها آورده شود.
- آوردن اندازه اولین بعد آرایه لازم نیست و می‌توان کروشه اول را خالی گذاشت.
- در فراخوانی یک تابع با پارامترهای آرایه، این پارامترها لزوماً به صورت فراخوانی با ارجاع هستند و هر تغییری در آرایه در داخل تابع، روی آرایهٔ اصلی نیز اعمال می‌شود.
- امکان بازگرداندن یک آرایه از یک تابع نیست و لذا در صورتی که لازم است یک آرایه از یک تابع برگشت داده شود، آن را در بین پارامترهای تابع قرار می‌دهیم.

آرایه به عنوان آرگومان تابع

آرایه به عنوان آرگومان تابع:

- انتقال آرایه‌ها به توابع نیز مشابه سایر متغیرها است. تنها تفاوت این است که در تعریف پارامترهای تابع، باید آرایه‌ها نیز مشابه تعریف آرایه‌ها، یعنی با استفاده از کروشه و مشخص کردن اندازهٔ آنها آورده شود.
- آوردن اندازه اولین بعد آرایه لازم نیست و می‌توان کروشه اول را خالی گذاشت.
- در فراخوانی یک تابع با پارامترهای آرایه، این پارامترها لزوماً به صورت فراخوانی با ارجاع هستند و هر تغییری در آرایه در داخل تابع، روی آرایهٔ اصلی نیز اعمال می‌شود.
- امکان بازگرداندن یک آرایه از یک تابع نیست و لذا در صورتی که لازم است یک آرایه از یک تابع برگشت داده شود، آن را در بین پارامترهای تابع قرار می‌دهیم.

آرایه به عنوان آرگومان تابع

آرایه به عنوان آرگومان تابع:

- انتقال آرایه‌ها به توابع نیز مشابه سایر متغیرها است. تنها تفاوت این است که در تعریف پارامترهای تابع، باید آرایه‌ها نیز مشابه تعریف آرایه‌ها، یعنی با استفاده از کروشه و مشخص کردن اندازهٔ آنها آورده شود.
- آوردن اندازه اولین بعد آرایه لازم نیست و می‌توان کروشه اول را خالی گذاشت.
- در فراخوانی یک تابع با پارامترهای آرایه، این پارامترها لزوماً به صورت **فراخوانی با ارجاع** هستند و هر تغییری در آرایه در داخل تابع، روی آرایهٔ اصلی نیز اعمال می‌شود.
- امکان بازگرداندن یک آرایه از یک تابع نیست و لذا در صورتی که لازم است یک آرایه از یک تابع برگشت داده شود، آن را در بین پارامترهای تابع قرار می‌دهیم.

آرایه به عنوان آرگومان تابع

آرایه به عنوان آرگومان تابع:

- انتقال آرایه‌ها به توابع نیز مشابه سایر متغیرها است. تنها تفاوت این است که در تعریف پارامترهای تابع، باید آرایه‌ها نیز مشابه تعریف آرایه‌ها، یعنی با استفاده از کروشه و مشخص کردن اندازهٔ آنها آورده شود.
- آوردن اندازه اولین بعد آرایه لازم نیست و می‌توان کروشه اول را خالی گذاشت.
- در فراخوانی یک تابع با پارامترهای آرایه، این پارامترها لزوماً به صورت **فراخوانی با ارجاع** هستند و هر تغییری در آرایه در داخل تابع، روی آرایهٔ اصلی نیز اعمال می‌شود.
- امکان بازگرداندن یک آرایه از یک تابع نیست و لذا در صورتی که لازم است یک آرایه از یک تابع برگشت داده شود، آن را در بین پارامترهای تابع قرار می‌دهیم.

مثال ۹

زیرالگوریتمی بنویسید که یک آرایه از اعداد و تعداد آن‌ها را دریافت کرده و عناصر موجود در آرایه را به‌ترتیب صعودی مرتب کند.

الگوریتم:

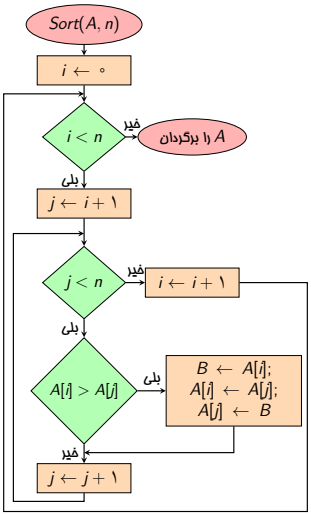
```
۱  شروع Sort(A, n)
۲      i ← ۰
۳      اگر i < n آنگاه
۴          j ← i + ۱
۵          اگر j < n آنگاه
۶              اگر A[i] > A[j] آنگاه
۷                  B ← A[i]
۸                  A[i] ← A[j]
۹                  A[j] ← B
۱۰             پایان اگر
۱۱             j ← j + ۱
۱۲             به مرحلهٔ ۵ برو
۱۳         پایان اگر
۱۴         i ← i + ۱
۱۵         به مرحلهٔ ۳ برو
۱۶     پایان اگر
۱۷     A را برگردان
```

مثال ۹

زیرالگوریتمی بنویسید که یک آرایه از اعداد و تعداد آن‌ها را دریافت کرده و عناصر موجود در آرایه را به‌ترتیب صعودی مرتب کند.

الگوریتم:

```
۱  شروع Sort(A, n)
۲      i ← ۰
۳      اگر i < n آنگاه
۴          j ← i + ۱
۵          اگر j < n آنگاه
۶              اگر A[i] > A[j] آنگاه
۷                  B ← A[i]
۸                  A[i] ← A[j]
۹                  A[j] ← B
۱۰             پایان اگر
۱۱             j ← j + ۱
۱۲             به مرحلهٔ ۵ برو
۱۳         پایان اگر
۱۴         i ← i + ۱
۱۵         به مرحلهٔ ۳ برو
۱۶     پایان اگر
۱۷     A را برگردان
```



مثال ۹

زیرالگوریتمی بنویسید که یک آرایه از اعداد و تعداد آن‌ها را دریافت کرده و عناصر موجود در آرایه را به‌ترتیب صعودی مرتب کند.

الگوریتم:

```
۱  شروع Sort(A, n)
۲      i ← ۰
۳      اگر i < n − ۱ آنگاه
۴          j ← i + ۱
۵          اگر j < n آنگاه
۶              اگر A[i] > A[j] آنگاه
۷                  B ← A[i]
۸                  A[i] ← A[j]
۹                  A[j] ← B
۱۰             پایان اگر
۱۱             j ← j + ۱
۱۲             به مرحلهٔ ۵ برو
۱۳         پایان اگر
۱۴         i ← i + ۱
۱۵         به مرحلهٔ ۳ برو
۱۶     پایان اگر
۱۷     A را برگردان
```

```
1  void Sort(double A[500], unsigned int n)
2  {
3      unsigned int i,j;
4      double B;
5      for(i=0; i<n; i++)
6          for(j=i+1; j<n; j++)
7              if (A[i]>A[j])
8              {
9                  B=A[i];
10                 A[i]=A[j];
11                 A[j]=B;
12             }
13     return;
14 }
```

مثال ۱۰

الگوریتم و فلوچارتی ارائه کنید که ابتدا عدد n و سپس n عدد را بگیرد و آنها را به ترتیب صعودی مرتب کرده و چاپ کند.

الگوریتم:

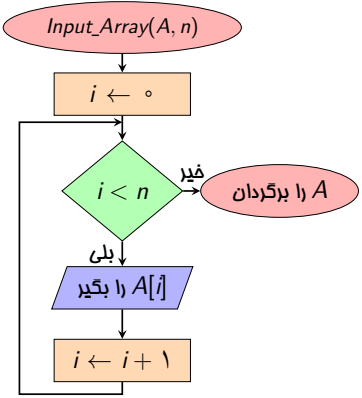
۱ شروع $Input\_Array(A, n)$
۲ $i \leftarrow 0$
۳ اگر $i < n$ آنگاه
۴ | $A[i]$ را بگیر
۵ | $i \leftarrow i + 1$
۶ | به مرحلهٔ ۳ برو
۷ پایان اگر
۸ A را برگردان

مثال ۱۰

الگوریتم و فلوچارتی ارائه کنید که ابتدا عدد n و سپس n عدد را بگیرد و آنها را به ترتیب صعودی مرتب کرده و چاپ کند.

الگوریتم:

۱ شروع $Input\_Array(A, n)$
۲ $i \leftarrow 0$
۳ اگر $i < n$ آنگاه
۴ | $A[i]$ را بگیر
۵ | $i \leftarrow i + 1$
۶ | به مرحلهٔ ۳ برو
۷ پایان اگر
۸ A را برگردان



مثال ۱۰

الگوریتم و فلوچارتی ارائه کنید که ابتدا عدد n و سپس n عدد را بگیرد و آنها را به ترتیب صعودی مرتب کرده و چاپ کند.

الگوریتم:

۱ شروع $Input\_Array(A, n)$
۲ $i \leftarrow 0$
۳ اگر $i < n$ آنگاه
۴ | $A[i]$ را بگیر
۵ | $i \leftarrow i + 1$
۶ | به مرحلهٔ ۳ برو
۷ پایان اگر
۸ A را برگردان

```
1  void Input_Array(double A[500], unsigned int n)
2  {
3      unsigned int i;
4      for(i=0; i<n; i++)
5      {
6          cout<<"Enter Next Element:";
7          cin>>A[i];
8      }
9      return;
10 }
```

مثال ۱۱

الگوریتم و فلوچارتی ارائه کنید که ابتدا عدد n و سپس n عدد را بگیرد و آنها را به ترتیب صعودی مرتب کرده و چاپ کند.

الگوریتم:

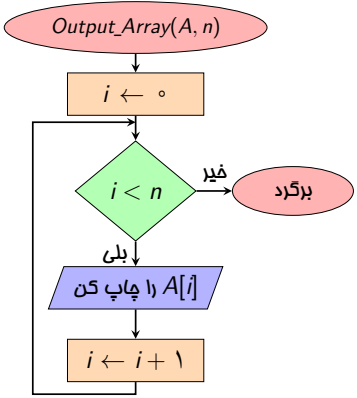
۱ شروع $Output\_Array(A, n)$
۲ $i \leftarrow 0$
۳ اگر $i < n$ آنگاه
۴ | $A[i]$ را چاپ کن
۵ | $i \leftarrow i + 1$
۶ | به مرحلهٔ ۳ برو
۷ پایان اگر
۸ برگرد

مثال ۱۱

الگوریتم و فلوچارتی ارائه کنید که ابتدا عدد n و سپس n عدد را بگیرد و آنها را به ترتیب صعودی مرتب کرده و چاپ کند.

الگوریتم:

۱ شروع $Output\_Array(A, n)$
۲ $i \leftarrow 0$
۳ اگر $i < n$ آنگاه
۴ | $A[i]$ را چاپ کن
۵ | $i \leftarrow i + 1$
۶ | به مرحلهٔ ۳ برو
۷ پایان اگر
۸ برگرد



مثال ۱۱

الگوریتم و فلوچارتی ارائه کنید که ابتدا عدد n و سپس n عدد را بگیرد و آنها را به ترتیب صعودی مرتب کرده و چاپ کند.

الگوریتم:

۱ شروع $Output\_Array(A, n)$
۲ $i \leftarrow 0$
۳ اگر $i < n$ آنگاه
۴ | $A[i]$ را چاپ کن
۵ | $i \leftarrow i + 1$
۶ | به مرحلهٔ ۳ برو
۷ پایان اگر
۸ برگرد

```
1  void Output_Array(double A[500], unsigned int n)
2  {
3      unsigned int i;
4      for(i=0; i<n; i++)
5          cout<<A[i]<<",";
6      return;
7  }
```

مثال ۱۲

الگوریتم و فلوچارتی ارائه کنید که ابتدا عدد n و سپس n عدد را بگیرد و آنها را به ترتیب صعودی مرتب کرده و چاپ کند.

الگوریتم:

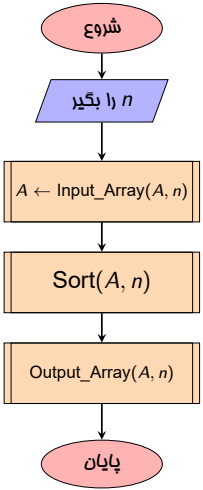
۱ شروع
۲ n را بگیر
۳ $A \leftarrow \text{Input\_Array}(A, n)$
۴ $\text{Sort}(A, n)$
۵ $\text{Output\_Array}(A, n)$
۶ پایان

مثال ۱۲

الگوریتم و فلوچارتی ارائه کنید که ابتدا عدد n و سپس n عدد را بگیرد و آنها را به ترتیب صعودی مرتب کرده و چاپ کند.

الگوریتم:

۱ شروع
۲ n را بگیر
۳ $A \leftarrow \text{Input\_Array}(A, n)$
۴ $\text{Sort}(A, n)$
۵ $\text{Output\_Array}(A, n)$
۶ پایان



مثال ۱۲

الگوریتم و فلوچارتی ارائه کنید که ابتدا عدد n و سپس n عدد را بگیرد و آنها را به ترتیب صعودی مرتب کرده و چاپ کند.

الگوریتم:

۱ شروع
۲ n را بگیر
۳ $A \leftarrow \text{Input\_Array}(A, n)$
۴ $\text{Sort}(A, n)$
۵ $\text{Output\_Array}(A, n)$
۶ پایان

```
1  int main()
2  {
3      unsigned int n;
4      double A[500];
5      cout << "Enter n:  ";
6      cin >> n;
7      Input_Array(A,n);
8      Sort(A,n);
9      Output_Array(A,n);
10     return 0;
11 }
```

مثال ۱۳

جستجوی داده‌ها: آیا عدد x در آرایهٔ A شامل n عدد موجود است؟

الگوریتم:

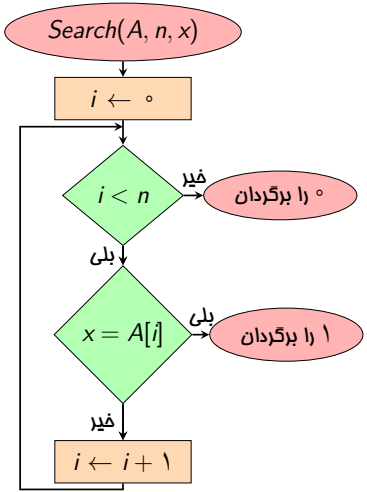
۱ شروع $Search(A, n, x)$
۲ $i \leftarrow 0$
۳ اگر $i < n$ آنگاه
۴ | اگر $x = A[i]$ آنگاه
۵ | | ۱ را برگردان
۶ | پایان اگر
۷ | $i \leftarrow i + 1$
۸ | به مرحلهٔ ۳ برو
۹ پایان اگر
۱۰ ۰ را برگردان

مثال ۱۳

جستجوی داده‌ها: آیا عدد x در آرایهٔ A شامل n عدد موجود است؟

الگوریتم:

۱ شروع $Search(A, n, x)$
۲ $i \leftarrow ۰$
۳ اگر $i < n$ آنگاه
۴ | اگر $x = A[i]$ آنگاه
۵ | | ۱ را برگردان
۶ | پایان اگر
۷ | $i \leftarrow i + ۱$
۸ | به مرحلهٔ ۳ برو
۹ پایان اگر
۱۰ ۰ را برگردان



مثال ۱۳

جستجوی داده‌ها: آیا عدد x در آرایهٔ A شامل n عدد موجود است؟

الگوریتم:

۱ شروع $Search(A, n, x)$
۲ $i \leftarrow 0$
۳ اگر $i < n$ آنگاه
۴ اگر $x = A[i]$ آنگاه
۵ ۱ را برگردان
۶ پایان اگر
۷ $i \leftarrow i + 1$
۸ به مرحلهٔ ۳ برو
۹ پایان اگر
۱۰ ۰ را برگردان

```
1  int Search(double A[ ], int n, double x)
2  {
3      int i=0;
4      for ( i=0;i<n;i++)
5      {
6          if (x == A[i ])
7              return 1;
8      }
9      return 0;
10 }
```

مثال ۱۴

جستجوی داده‌ها: آیا عدد x در آرایهٔ A شامل n عدد موجود است؟

الگوریتم:

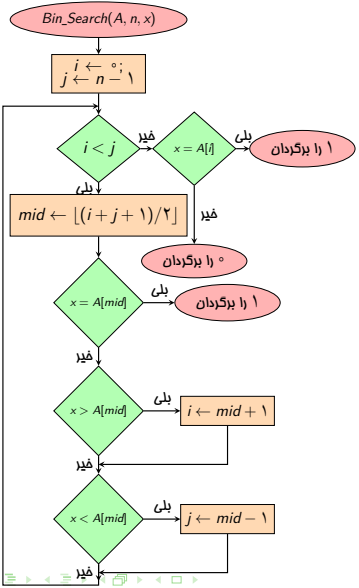
```
۱  شروع Bin_Search(A, n, x)
۲      i ← ۰ ; j ← n − ۱
۳      اگر i < j آنگاه
۴          mid ← ⌊(i + j + ۱)/۲⌋
۵          اگر x = A[mid] آنگاه
۶              ۱ را برگردان
۷          پایان اگر
۸          اگر x > A[mid] آنگاه
۹              i ← mid + ۱
۱۰         پایان اگر
۱۱         اگر x < A[mid] آنگاه
۱۲             j ← mid − ۱
۱۳         پایان اگر
۱۴         به مرحلهٔ ۳ برو
۱۵     پایان اگر
۱۶     اگر x = A[i] آنگاه
۱۷         ۱ را برگردان
۱۸     درغیراینصورت
۱۹         ۰ را برگردان
۲۰     پایان اگر
```

مثال ۱۴

جستجوی داده‌ها: آیا عدد x در آرایهٔ A شامل n عدد موجود است؟

الگوریتم:

```
۱  شروع Bin_Search(A, n, x)
۲      i ← ۰ ; j ← n − ۱
۳      اگر i < j آنگاه
۴          mid ← ⌊(i + j + ۱)/۲⌋
۵          اگر x = A[mid] آنگاه
۶              ۱ را برگردان
۷          پایان اگر
۸          اگر x > A[mid] آنگاه
۹              i ← mid + ۱
۱۰         پایان اگر
۱۱         اگر x < A[mid] آنگاه
۱۲             j ← mid − ۱
۱۳         پایان اگر
۱۴         به مرحلهٔ ۳ برو
۱۵     پایان اگر
۱۶     اگر x = A[i] آنگاه
۱۷         ۱ را برگردان
۱۸     درغیراینصورت
۱۹         ۰ را برگردان
۲۰     پایان اگر
```



مثال ۱۴

جستجوی داده‌ها: آیا عدد x در آرایهٔ A شامل n عدد موجود است؟

الگوریتم:

```
۱  شروع Bin_Search(A, n, x)
۲      i ← ۰ ; j ← n − ۱
۳      اگر i < j آنگاه
۴          mid ← ⌊(i + j + ۱)/۲⌋
۵          اگر x = A[mid] آنگاه
۶              ۱ را برگردان
۷          پایان اگر
۸          اگر x > A[mid] آنگاه
۹              i ← mid + ۱
۱۰         پایان اگر
۱۱         اگر x < A[mid] آنگاه
۱۲             j ← mid − ۱
۱۳         پایان اگر
۱۴         به مرحلهٔ ۳ برو
۱۵     پایان اگر
۱۶     اگر x = A[i] آنگاه
۱۷         ۱ را برگردان
۱۸     درغیراینصورت
۱۹         ۰ را برگردان
۲۰     پایان اگر
```

۰	۱	۲	۳	۴	۵	۶	۷	۸	۹	۱۰	۱۱	۱۲	۱۳
۲	۱۱	۲۰	۳۵	۳۶	۸۱	۱۰۲	۲۱۲	۳۴۲	۴۲۰	۵۵۵	۵۸۰	۶۵۷	۷۲۱

n	x	i	j	mid
۱۴	۱۰۲	۰	۱۳	۷
			۶	۳
		۴		۵
		۶		

مثال ۱۴

جستجوی داده‌ها: آیا عدد x در آرایهٔ A شامل n عدد موجود است؟

الگوریتم:

۱ شروع $Bin\_Search(A, n, x)$
۲ $i \leftarrow ۰$; $j \leftarrow n - ۱$
۳ اگر $i < j$ آنگاه
۴ $mid \leftarrow \lfloor (i + j + ۱)/۲ \rfloor$
۵ اگر $x = A[mid]$ آنگاه
۶ ۱ را برگردان
۷ پایان اگر
۸ اگر $x > A[mid]$ آنگاه
۹ $i \leftarrow mid + ۱$
۱۰ پایان اگر
۱۱ اگر $x < A[mid]$ آنگاه
۱۲ $j \leftarrow mid - ۱$
۱۳ پایان اگر
۱۴ به مرحلهٔ ۳ برو
۱۵ پایان اگر
۱۶ اگر $x = A[i]$ آنگاه
۱۷ ۱ را برگردان
۱۸ درغیراینصورت
۱۹ ۰ را برگردان
۲۰ پایان اگر

```
1  int Bin_Search(double A[ ], int n, double x)
2  {
3      unsigned int i=0,j=n−1,mid;
4      while(i < j)
5      {
6          mid=(i+j+1)/2;
7          if (x == A[mid])
8              return 1;
9          if (x > A[mid])
10             i=mid+1;
11         if (x < A[mid])
12             j=mid−1;
13     }
14     if (x == A[i])
15         return 1;
16     else
17         return 0;
18 }
```

مثال ۱۵

الگوریتم و فلوچارتی ارائه کنید که ابتدا عدد n و سپس n نمره را بگیرد و تعداد افراد با بیشترین نمره را محاسبه و چاپ کند.

الگوریتم:

سه زیرمسأله:

- زیرمسألهٔ گرفتن n عدد و قرار دادن در آرایه.
- زیرمسألهٔ محاسبهٔ بزرگ‌ترین عدد در آرایه.
- زیرمسألهٔ محاسبهٔ تعداد تکرار یک عدد در آرایه.

```
۱  شروع Max_Element(A, n)
۲      Max ← A[۰]
۳      i ← ۱
۴      اگر i < n آنگاه
۵          اگر A[i] > Max آنگاه
۶              Max ← A[i]
۷          پایان اگر
۸          i ← i + ۱
۹          به مرحلهٔ ۳ برو
۱۰     پایان اگر
۱۱     Max را برگردان
```

مثال ۱۵

الگوریتم و فلوچارتی ارائه کنید که ابتدا عدد n و سپس n نمره را بگیرد و تعداد افراد با بیشترین نمره را محاسبه و چاپ کند.

الگوریتم:

سه زیرمسأله:

- زیرمسألهٔ گرفتن n عدد و قرار دادن در آرایه.
- زیرمسألهٔ محاسبهٔ بزرگ‌ترین عدد در آرایه.
- زیرمسألهٔ محاسبهٔ تعداد تکرار یک عدد در آرایه.

۱ شروع $Max\_Element(A, n)$
۲ $Max \leftarrow A[۰]$
۳ $i \leftarrow ۱$
۴ اگر $i < n$ آنگاه
۵ اگر $Max < A[i]$ آنگاه
۶ $Max \leftarrow A[i]$
۷ پایان اگر
۸ $i \leftarrow i + ۱$
۹ به مرحلهٔ ۳ برو
۱۰ پایان اگر
۱۱ Max را برگردان

مثال ۱۵

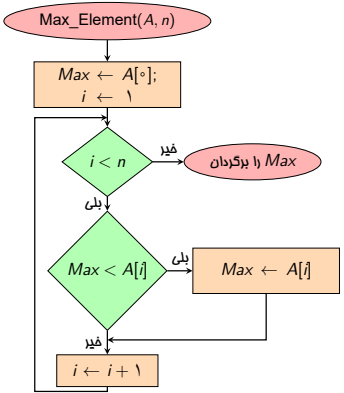
الگوریتم و فلوچارتی ارائه کنید که ابتدا عدد n و سپس n نمره را بگیرد و تعداد افراد با بیشترین نمره را محاسبه و چاپ کند.

الگوریتم:

سه زیرمسأله:

- زیرمسألهٔ گرفتن n عدد و قرار دادن در آرایه.
- زیرمسألهٔ محاسبهٔ بزرگ‌ترین عدد در آرایه.
- زیرمسألهٔ محاسبهٔ تعداد تکرار یک عدد در آرایه.

۱ شروع $Max\_Element(A, n)$
۲ $Max \leftarrow A[۰]$
۳ $i \leftarrow ۱$
۴ اگر $i < n$ آنگاه
۵ اگر $Max < A[i]$ آنگاه
۶ $Max \leftarrow A[i]$
۷ پایان اگر
۸ $i \leftarrow i + ۱$
۹ به مرحلهٔ ۳ برو
۱۰ پایان اگر
۱۱ Max را برگردان



مثال ۱۵

الگوریتم و فلوچارتی ارائه کنید که ابتدا عدد n و سپس n نمره را بگیرد و تعداد افراد با بیشترین نمره را محاسبه و چاپ کند.

الگوریتم:

سه زیرمسأله:

- زیرمسألهٔ گرفتن n عدد و قرار دادن در آرایه.
- زیرمسألهٔ محاسبهٔ بزرگ‌ترین عدد در آرایه.
- زیرمسألهٔ محاسبهٔ تعداد تکرار یک عدد در آرایه.

۱ شروع $Max\_Element(A, n)$
۲ $Max \leftarrow A[۰]$
۳ $i \leftarrow ۱$
۴ اگر $i < n$ آنگاه
۵ اگر $Max < A[i]$ آنگاه
۶ $Max \leftarrow A[i]$
۷ پایان اگر
۸ $i \leftarrow i + ۱$
۹ به مرحلهٔ ۳ برو
۱۰ پایان اگر
۱۱ Max را برگردان

```
1  double Max_Element(double A[ ],int n)
2  {
3       double Max=A[0];
4       int  i ;
5       for(i=1; i<n; i++)
6            if (Max<A[i])
7                 Max=A[i];
8       return  Max;
9  }
```

مثال ۱۶

الگوریتم و فلوچارتی ارائه کنید که ابتدا عدد n و سپس n نمره را بگیرد و تعداد افراد با بیشترین نمره را محاسبه و چاپ کند.

الگوریتم:

سه زیرمسأله:

- زیرمسألهٔ گرفتن n عدد و قرار دادن در آرایه.
- زیرمسألهٔ محاسبهٔ بزرگ‌ترین عدد در آرایه.
- زیرمسألهٔ محاسبهٔ تعداد تکرار یک عدد در آرایه.

```
۱  شروع Count(A, n, x)
۲      counter ← ۰ ؛ i ← ۰
۳      اگر i < n آنگاه
۴          اگر A[i] = x آنگاه
۵              counter ← counter + ۱
۶          پایان اگر
۷          i ← i + ۱
۸          به مرحلهٔ ۳ برو
۹      پایان اگر
۱۰     counter را برگردان
```

مثال ۱۶

الگوریتم و فلوچارتی ارائه کنید که ابتدا عدد n و سپس n نمره را بگیرد و تعداد افراد با بیشترین نمره را محاسبه و چاپ کند.

الگوریتم:

سه زیرمسأله:

- زیرمسألهٔ گرفتن n عدد و قرار دادن در آرایه.
- زیرمسألهٔ محاسبهٔ بزرگ‌ترین عدد در آرایه.
- زیرمسألهٔ محاسبهٔ تعداد تکرار یک عدد در آرایه.

۱ شروع $Count(A, n, x)$
۲ $counter \leftarrow ۰$؛ $i \leftarrow ۰$
۳ اگر $i < n$ آنگاه
۴ اگر $A[i] = x$ آنگاه
۵ $counter \leftarrow counter + ۱$
۶ پایان اگر
۷ $i \leftarrow i + ۱$
۸ به مرحلهٔ ۳ برو
۹ پایان اگر
۱۰ $counter$ را برگردان

مثال ۱۶

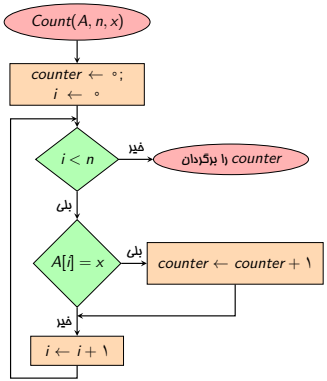
الگوریتم و فلوچارتی ارائه کنید که ابتدا عدد n و سپس n نمره را بگیرد و تعداد افراد با بیشترین نمره را محاسبه و چاپ کند.

الگوریتم:

سه زیرمسأله:

- زیرمسألهٔ گرفتن n عدد و قرار دادن در آرایه.
- زیرمسألهٔ محاسبهٔ بزرگ‌ترین عدد در آرایه.
- زیرمسألهٔ محاسبهٔ تعداد تکرار یک عدد در آرایه.

۱ شروع $Count(A, n, x)$
۲ $counter \leftarrow 0$؛ $i \leftarrow 0$
۳ اگر $i < n$ آنگاه
۴ اگر $A[i] = x$ آنگاه
۵ $counter \leftarrow counter + 1$
۶ پایان اگر
۷ $i \leftarrow i + 1$
۸ به مرحلهٔ ۳ برو
۹ پایان اگر
۱۰ $counter$ را برگردان



مثال ۱۶

الگوریتم و فلوچارتی ارائه کنید که ابتدا عدد n و سپس n نمره را بگیرد و تعداد افراد با بیشترین نمره را محاسبه و چاپ کند.

الگوریتم:

سه زیرمسأله:

- زیرمسألهٔ گرفتن n عدد و قرار دادن در آرایه.
- زیرمسألهٔ محاسبهٔ بزرگ‌ترین عدد در آرایه.
- زیرمسألهٔ محاسبهٔ تعداد تکرار یک عدد در آرایه.

۱ شروع $Count(A, n, x)$
۲ $counter \leftarrow 0$؛ $i \leftarrow 0$
۳ اگر $i < n$ آنگاه
۴ اگر $A[i] = x$ آنگاه
۵ $counter \leftarrow counter + 1$
۶ پایان اگر
۷ $i \leftarrow i + 1$
۸ به مرحلهٔ ۳ برو
۹ پایان اگر
۱۰ $counter$ را برگردان

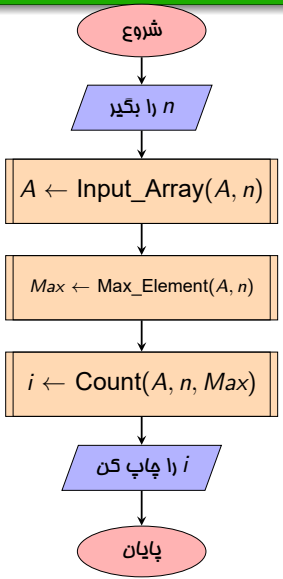
```
1  int Count(double A[ ], int n, double x)
2  {
3      int counter=0,i;
4      for(i=0; i<n; i++)
5          if (A[i] == x)
6              counter++;
7      return counter;
8  }
```

مثال ۱۷

الگوریتم و فلوچارتی ارائه کنید که ابتدا عدد n و سپس n نمره را بگیرد و تعداد افراد با بیشترین نمره را محاسبه و چاپ کند.

الگوریتم:

۱ شروع
۲ n را بگیر
۳ $A \leftarrow Input\_Array(A, n)$
۴ $Max \leftarrow Max\_Element(A, n)$
۵ $i \leftarrow Count(A, n, Max)$
۶ i را چاپ کن
۷ پایان



مثال ۱۷

الگوریتم و فلوچارتی ارائه کنید که ابتدا عدد n و سپس n نمره را بگیرد و تعداد افراد با بیشترین نمره را محاسبه و چاپ کند.

الگوریتم:

۱ شروع
۲ n را بگیر
۳ $A \leftarrow Input\_Array(A, n)$
۴ $Max \leftarrow Max\_Element(A, n)$
۵ $i \leftarrow Count(A, n, Max)$
۶ i را چاپ کن
۷ پایان

مثال ۱۷

الگوریتم و فلوچارتی ارائه کنید که ابتدا عدد n و سپس n نمره را بگیرد و تعداد افراد با بیشترین نمره را محاسبه و چاپ کند.

الگوریتم:

۱ شروع
۲ n را بگیر
۳ $A \leftarrow Input\_Array(A, n)$
۴ $Max \leftarrow Max\_Element(A, n)$
۵ $i \leftarrow Count(A, n, Max)$
۶ i را چاپ کن
۷ پایان

```
1  int main()
2  {
3      unsigned int n,i;
4      double A[500],Max;
5      cout << "Enter n: ";
6      cin >> n;
7      Input_Array(A,n);
8      Max=Max_Element(A,n);
9      i=Count(A,n,Max);
10     cout<<"# of Max Grades:"<<i;
11     return 0;
12 }
```

آرایهٔ دو بعدی

آرایهٔ دو بعدی

یک آرایه، از تعدادی متغیر هم‌نوع تشکیل شده است که در فضایی به‌هم پیوسته در حافظهٔ اصلی ذخیره می‌شوند و دارای ویژگی‌های زیر است.

- هر آرایه دو بعدی دارای یک نام و یک طول و یک عرض است که طول آرایه همان تعداد سطرهای آن و عرض، تعداد ستون‌های آن آرایه است.
- به هر عنصر آرایه با طول n و عرض m می‌توان به‌وسیلهٔ نام آرایه و دو اندیس، یک اندیس بین صفر و $n-1$ یک اندیس بین صفر و $m-1$ دسترسی داشت.

$A[0][0]$	$A[0][1]$	$A[0][2]$	$A[0][3]$
$A[1][0]$	$A[1][1]$	$A[1][2]$	$A[1][3]$
$A[2][0]$	$A[2][1]$	$A[2][2]$	$A[2][3]$

آرایهٔ دو بعدی

آرایهٔ دو بعدی

یک آرایه، از تعدادی متغیر هم‌نوع تشکیل شده است که در فضایی به‌هم پیوسته در حافظهٔ اصلی ذخیره می‌شوند و دارای ویژگی‌های زیر است.

- هر آرایه دو بعدی دارای یک نام و یک طول و یک عرض است که طول آرایه همان تعداد سطرهای آن و عرض، تعداد ستون‌های آن آرایه است.
- به هر عنصر آرایه با طول n و عرض m می‌توان به‌وسیلهٔ نام آرایه و دو اندیس، یک اندیس بین صفر و $n-1$ یک اندیس بین صفر و $m-1$ دسترسی داشت.

$A[0][0]$	$A[0][1]$	$A[0][2]$	$A[0][3]$
$A[1][0]$	$A[1][1]$	$A[1][2]$	$A[1][3]$
$A[2][0]$	$A[2][1]$	$A[2][2]$	$A[2][3]$

مثال ۱۸

زیرالگوریتم گرفتن عناصر یک آرایهٔ دو بعدی با m سطر و n ستون.

الگوریتم:

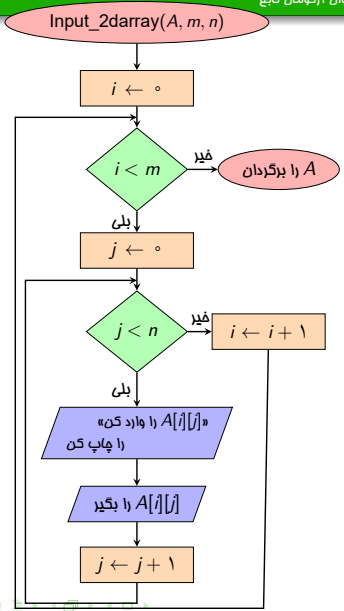
۱ شروع $Input\_2darray(A, m, n)$
۲ $i \leftarrow ۰$
۳ اگر $i < m$ آنگاه
۴ $j \leftarrow ۰$
۵ اگر $n < j$ آنگاه
۶ «$A[i][j]$ را وارد کن» را چاپ کن
۷ $A[i][j]$ را بگیر
۸ $j \leftarrow j + ۱$
۹ برو به مرحلهٔ ۵
۱۰ پایان اگر
۱۱ $i \leftarrow i + ۱$
۱۲ به مرحلهٔ ۳ برو
۱۳ پایان اگر
۱۴ A را برگردان

مثال ۱۸

زیرالگوریتم گرفتن عناصر یک آرایهٔ دو بعدی با m سطر و n ستون.

الگوریتم:

۱ شروع *Input_2darray*(A, m, n)
۲ $i \leftarrow ۰$
۳ اگر $i < m$ آنگاه
۴ $j \leftarrow ۰$
۵ اگر $j < n$ آنگاه
۶ «$A[i][j]$ را وارد کن» را چاپ کن
۷ $A[i][j]$ را بگیر
۸ $j \leftarrow j + ۱$
۹ برو به مرحلهٔ ۵
۱۰ پایان اگر
۱۱ $i \leftarrow i + ۱$
۱۲ به مرحلهٔ ۳ برو
۱۳ پایان اگر
۱۴ A را برگردان



مثال ۱۸

زیرالگوریتم گرفتن عناصر یک آرایهٔ دو بعدی با m سطر و n ستون.

الگوریتم:

۱ شروع $Input\_2darray(A, m, n)$
۲ $i \leftarrow ۰$
۳ اگر $i < m$ آنگاه
۴ $j \leftarrow ۰$
۵ اگر $j < n$ آنگاه
۶ «$A[i][j]$ را وارد کن» را چاپ کن
۷ $A[i][j]$ را بگیر
۸ $j \leftarrow j + ۱$
۹ برو به مرحلهٔ ۵
۱۰ پایان اگر
۱۱ $i \leftarrow i + ۱$
۱۲ به مرحلهٔ ۳ برو
۱۳ پایان اگر
۱۴ A را برگردان

```
1  void Input_2darray(double A[100][100],int m,int n)
2  {
3      int i,j;
4      for(i=0; i<m; i++)
5          for(j=0; j<n; j++)
6          {
7              cout<<"\nEnter A["<<i<<"]["<<j <<"]:";
8              cin>>A[i][j];
9          }
10 }
```

مثال ۱۹

زیرالگوریتم و زیرفلوچارتی رسم کنید که دو ماتریس $m \times n$ ی A و B و ابعاد m و n را به عنوان پارامتر دریافت کند و مجموع A و B یا $A + B$ را محاسبه کرده و برگرداند.

الگوریتم:

۱ شروع Matrix_Sum(A, B, m, n)
۲ $i \leftarrow ۰$
۳ اگر $i < m$ آنگاه
۴ $j \leftarrow ۰$
۵ اگر $j < n$ آنگاه
۶ $C[i][j] \leftarrow A[i][j] + B[i][j]$
۷ $j \leftarrow j + ۱$
۸ برو به مرحلهٔ ۵
۹ پایان اگر
۱۰ $i \leftarrow i + ۱$
۱۱ به مرحلهٔ ۳ برو
۱۲ پایان اگر
۱۳ C را برگردان

مثال ۱۹

زیرالگوریتم و زیرفلوچارتی رسم کنید که دو ماتریس $m \times n$ ی A و B و ابعاد m و n را به عنوان پارامتر دریافت کند و مجموع A و B یا $A + B$ را محاسبه کرده و برگرداند.

الگوریتم:

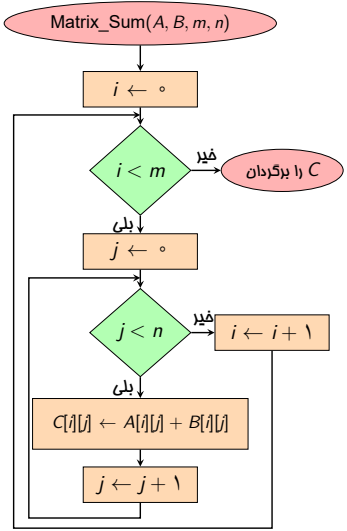
۱ شروع Matrix_Sum(A, B, m, n)
۲ $i \leftarrow ۰$
۳ اگر $i < m$ آنگاه
۴ $j \leftarrow ۰$
۵ اگر $j < n$ آنگاه
۶ $C[i][j] \leftarrow A[i][j] + B[i][j]$
۷ $j \leftarrow j + ۱$
۸ برو به مرحلهٔ ۵
۹ پایان اگر
۱۰ $i \leftarrow i + ۱$
۱۱ به مرحلهٔ ۳ برو
۱۲ پایان اگر
۱۳ C را برگردان

مثال ۱۹

زیرالگوریتم و زیرفلوچارتی رسم کنید که دو ماتریس $m \times n$ ی‌ A و B و ابعاد m و n را به عنوان پارامتر دریافت کند و مجموع A و B یا $A + B$ را محاسبه کرده و برگرداند.

الگوریتم:

۱ شروع Matrix_Sum(A, B, m, n)
۲ $i \leftarrow ۰$
۳ اگر $i < m$ آنگاه
۴ $j \leftarrow ۰$
۵ اگر $j < n$ آنگاه
۶ $C[i][j] \leftarrow A[i][j] + B[i][j]$
۷ $j \leftarrow j + ۱$
۸ برو به مرحلهٔ ۵
۹ پایان اگر
۱۰ $i \leftarrow i + ۱$
۱۱ به مرحلهٔ ۳ برو
۱۲ پایان اگر
۱۳ C را برگردان



مثال ۱۹

زیرالگوریتم و زیرفلوچارتی رسم کنید که دو ماتریس $m \times n$ ی A و B و ابعاد m و n را به عنوان پارامتر دریافت کند و مجموع A و B یا $A + B$ را محاسبه کرده و برگرداند.

الگوریتم:

۱ شروع Matrix_Sum(A, B, m, n)
۲ $i \leftarrow 0$
۳ اگر $i < m$ آنگاه
۴ $j \leftarrow 0$
۵ اگر $j < n$ آنگاه
۶ $C[i][j] \leftarrow A[i][j] + B[i][j]$
۷ $j \leftarrow j + 1$
۸ برو به مرحلهٔ ۵
۹ پایان اگر
۱۰ $i \leftarrow i + 1$
۱۱ به مرحلهٔ ۳ برو
۱۲ پایان اگر
۱۳ C را برگردان

```
1  void Matrix_Sum(
2        double A[100][100],
3        double B[100][100],
4        double C[100][100],
5        unsigned int m,
6        unsigned int n)
7  {
8     unsigned int i,j;
9     for(i=0; i<m; i++)
10       for(j=0; j<n; j++)
11          C[i][j]=A[i][j]+B[i][j];
12 }
```

مثال ۲۰

زیرالگوریتم و زیرفلوچارتی رسم کنید که یک ماتریس $n \times n$ به نام A و n را به عنوان پارامتر دریافت کند و اگر به ازای هر i بین ۰ و $n - ۱$، مجموع عناصر سطر iام با مجموع عناصر ستون iام برابر باشد مقدار ۱ و در غیر این صورت مقدار ۰ را برگرداند.

الگوریتم:

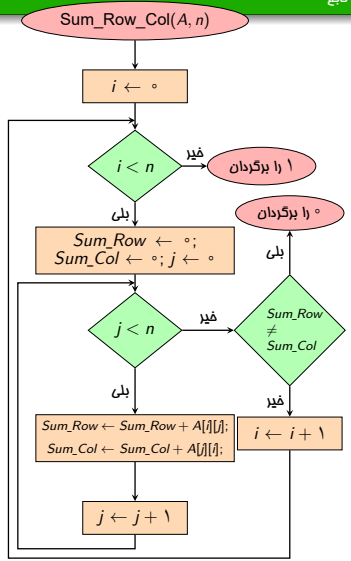
```
۱  شروع Sum_Row_Col(A, n)
۲      i ← ۰
۳      اگر i < n آنگاه
۴          Sum_Row ← ۰؛ Sum_Col ← ۰؛ j ← ۰
۵          اگر j < n آنگاه
۶              Sum_Row ← Sum_Row + A[i][j]
۷              Sum_Col ← Sum_Col + A[j][i]
۸              j ← j + ۱
۹              به مرحلهٔ ۵ برو
۱۰         پایان اگر
۱۱         اگر Sum_Row ≠ Sum_Col آنگاه
۱۲             ۰ را برگردان
۱۳         پایان اگر
۱۴         i ← i + ۱
۱۵         به مرحلهٔ ۳ برو
۱۶     پایان اگر
۱۷     ۱ را برگردان
```

مثال ۲۰

زیرالگوریتم و زیرفلوچارتی رسم کنید که یک ماتریس $n \times n$ به نام A و n را به عنوان پارامتر دریافت کند و اگر به ازای هر i بین ۰ و $n-1$، مجموع عناصر سطر iام با مجموع عناصر ستون iام برابر باشد مقدار ۱ و در غیر این صورت مقدار ۰ را برگرداند.

الگوریتم:

```
۱  شروع Sum_Row_Col(A, n)
۲      i ← ۰
۳      اگر i < n آنگاه
۴          Sum_Row ← ۰؛ Sum_Col ← ۰؛ j ← ۰
۵          اگر j < n آنگاه
۶              Sum_Row ← Sum_Row + A[i][j]
۷              Sum_Col ← Sum_Col + A[j][i]
۸              j ← j + ۱
۹              به مرحلهٔ ۵ برو
۱۰         پایان اگر
۱۱         اگر Sum_Row ≠ Sum_Col آنگاه
۱۲             ۰ را برگردان
۱۳         پایان اگر
۱۴         i ← i + ۱
۱۵         به مرحلهٔ ۳ برو
۱۶     پایان اگر
۱۷     ۱ را برگردان
```



```
1  bool Sum_Row_Col(
2      double A[100][100],
3      unsigned int  n)
4  {
5      unsigned int  i , j ;
6      double Sum_Row=0,
7           Sum_Col=0;
8      for ( i=0;  i<n;  i++)
9      {
10          for ( j=0;  j<n;  j++)
11          {
12              Sum_Row+=A[i][j];
13              Sum_Col+=A[j][i];
14          }
15          if (Sum_Row != Sum_Col)
16              return  0;
17     }
18     return  1;
19 }
```

مثال ۲۰

زیرالگوریتم و زیرفلوچارتی رسم کنید که یک ماتریس $n \times n$ به نام A و n را به عنوان پارامتر دریافت کند و اگر به ازای هر i بین ۰ و $n - ۱$، مجموع عناصر سطر iام با مجموع عناصر ستون iام برابر باشد مقدار ۱ و در غیر این صورت مقدار ۰ را برگرداند.

الگوریتم:

```
۱  شروع Sum_Row_Col(A, n)
۲      i ← ۰
۳      اگر i < n آنگاه
۴          Sum_Row ← ۰؛ Sum_Col ← ۰؛ j ← ۰
۵          اگر j < n آنگاه
۶              Sum_Row ← Sum_Row + A[i][j]
۷              Sum_Col ← Sum_Col + A[j][i]
۸              j ← j + ۱
۹              به مرحلهٔ ۵ برو
۱۰         پایان اگر
۱۱         اگر Sum_Row ≠ Sum_Col آنگاه
۱۲             ۰ را برگردان
۱۳         پایان اگر
۱۴         i ← i + ۱
۱۵         به مرحلهٔ ۳ برو
۱۶     پایان اگر
۱۷     ۱ را برگردان
```

مثال ۲۱

زیرالگوریتم و زیرفلوچارتی رسم کنید که یک ماتریس $m \times n$ به نام A، یک ماتریس $n \times k$ به نام B و m و n و k را به عنوان پارامتر بگیرد و $A \times B$ را محاسبه کرده و برگرداند.

الگوریتم:

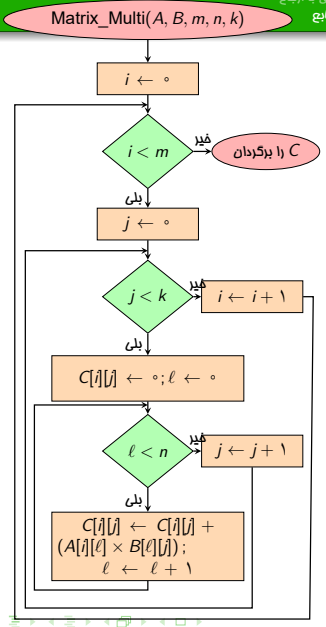
۱ شروع Matrix_Multi(A, B, m, n, k)
۲ $i \leftarrow ۰$
۳ اگر $i < m$ آنگاه
۴ $j \leftarrow ۰$
۵ اگر $j < k$ آنگاه
۶ $C[i][j] \leftarrow ۰$؛ $\ell \leftarrow ۰$
۷ اگر $\ell < n$ آنگاه
۸ $C[i][j] \leftarrow C[i][j] + (A[i][\ell] \times B[\ell][j])$
۹ $\ell \leftarrow \ell + ۱$
۱۰ برو به مرحلهٔ ۸
۱۱ پایان اگر
۱۲ $j \leftarrow j + ۱$
۱۳ برو به مرحلهٔ ۵
۱۴ پایان اگر
۱۵ $i \leftarrow i + ۱$
۱۶ به مرحلهٔ ۳ برو
۱۷ پایان اگر
۱۸ C را برگردان

مثال ۲۱

زیرالگوریتم و زیرفلوچارتی رسم کنید که یک ماتریس $m \times n$ به نام A، یک ماتریس $k \times n$ به نام B و m و n و k را به عنوان پارامتر بگیرد و $A \times B$ را محاسبه کرده و برگرداند.

الگوریتم:

```
۱  شروع Matrix_Multi(A, B, m, n, k)
۲      i ← ۰
۳      اگر i < m آنگاه
۴          j ← ۰
۵          اگر j < k آنگاه
۶              C[i][j] ← ۰ ؛ ℓ ← ۰
۷              اگر ℓ < n آنگاه
۸                  C[i][j] ← C[i][j] + (A[i][ℓ] × B[ℓ][j])
۹                  ℓ ← ℓ + ۱
۱۰                 برو به مرحلهٔ ۸
۱۱             پایان اگر
۱۲             j ← j + ۱
۱۳             برو به مرحلهٔ ۵
۱۴         پایان اگر
۱۵         i ← i + ۱
۱۶         به مرحلهٔ ۳ برو
۱۷     پایان اگر
۱۸     C را برگردان
```



```
1  void Matrix_Multi(
2       double A[ ][100],
3       double B[ ][100],
4       double C[ ][100],
5       unsigned int  m,
6       unsigned int  n,
7       unsigned int  k)
8  {
9    unsigned int  i , j , l ;
10   for ( i=0;  i<m;  i++)
11     for ( j=0;  j<k;  j++)
12     {
13       C[i ][ j ]=0;
14       for ( l=0;  l<n;  l++)
15         C[i ][ j ]+=(A[i ][ l ]*B[l ][ j ]) ;
16     }
17 }
```

مثال ۲۱

زیرالگوریتم و زیرفلوچارتی رسم کنید که یک ماتریس $m \times n$ به نام A، یک ماتریس $n \times k$ به نام B و m و n و k را به عنوان پارامتر بگیرد و $A \times B$ را محاسبه کرده و برگرداند.

الگوریتم:

۱ شروع Matrix_Multi(A, B, m, n, k)
۲ $i \leftarrow 0$
۳ اگر $i < m$ آنگاه
۴ $j \leftarrow 0$
۵ اگر $j < k$ آنگاه
۶ $C[i][j] \leftarrow 0$؛ $\ell \leftarrow 0$
۷ اگر $\ell < n$ آنگاه
۸ $C[i][j] \leftarrow C[i][j] + (A[i][\ell] \times B[\ell][j])$
۹ $\ell \leftarrow \ell + 1$
۱۰ برو به مرحلهٔ ۸
۱۱ پایان اگر
۱۲ $j \leftarrow j + 1$
۱۳ برو به مرحلهٔ ۵
۱۴ پایان اگر
۱۵ $i \leftarrow i + 1$
۱۶ به مرحلهٔ ۳ برو
۱۷ پایان اگر
۱۸ C را برگردان

قلمرو متغیرها

قلمرو متغیرها:

- قلمرو (scope) یک متغیر: محلهایی از برنامه است که آن متغیر قابل استفاده است، یعنی می‌توان با دستور انتساب به آن متغیر مقداری نسبت داد یا از محتوای ذخیره شده در آن متغیر استفاده کرد.
- هر متغیر از محل تعریف شدن تا آخر بلوکی که در آن تعریف شده است، قابل رویت و استفاده است و در خارج از محدوده قابل استفاده نیست.
- بلوک: مجموعه دستوراتی که با یک آکولاد باز شروع می‌شود و تا آکولاد بسته متناظر با آن ختم می‌شود.
- در یک قلمرو، نمی‌توان متغیرهای همنام داشت.
- در بحث فوق، یک استثنا وجود دارد و آن، زمانی است که بلوک‌های متداخل داشته باشیم. در این صورت می‌توانیم در بلوک داخلی، متغیری همنام با متغیری که در بلوک بیرونی تعریف شده، تعریف کنیم. در این حالت، استفاده از متغیر، بسته به محل استفاده از آن به آن متغیری اشاره می‌کند که در آن بلوک تعریف شده است.

قلمرو متغیرها

قلمرو متغیرها:

- قلمرو (scope) یک متغیر: محلهایی از برنامه است که آن متغیر قابل استفاده است، یعنی می‌توان با دستور انتساب به آن متغیر مقداری نسبت داد یا از محتوای ذخیره شده در آن متغیر استفاده کرد.
- هر متغیر از محل تعریف شدن تا آخر بلوکی که در آن تعریف شده است، قابل رویت و استفاده است و در خارج از محدوده قابل استفاده نیست.
- بلوک: مجموعه دستوراتی که با یک آکولاد باز شروع می‌شود و تا آکولاد بسته متناظر با آن ختم می‌شود.
- در یک قلمرو، نمی‌توان متغیرهای همنام داشت.
- در بحث فوق، یک استثنا وجود دارد و آن، زمانی است که بلوک‌های متداخل داشته باشیم. در این صورت می‌توانیم در بلوک داخلی، متغیری همنام با متغیری که در بلوک بیرونی تعریف شده، تعریف کنیم. در این حالت، استفاده از متغیر، بسته به محل استفاده از آن به آن متغیری اشاره می‌کند که در آن بلوک تعریف شده است.

قلمرو متغیرها

قلمرو متغیرها:

- قلمرو (scope) یک متغیر: محلهایی از برنامه است که آن متغیر قابل استفاده است، یعنی می‌توان با دستور انتساب به آن متغیر مقداری نسبت داد یا از محتوای ذخیره شده در آن متغیر استفاده کرد.
- هر متغیر از محل تعریف شدن تا آخر بلوکی که در آن تعریف شده است، قابل رویت و استفاده است و در خارج از محدوده قابل استفاده نیست.
- بلوک: مجموعه دستوراتی که با یک آکولاد باز شروع می‌شود و تا آکولاد بسته متناظر با آن ختم می‌شود.
- در یک قلمرو، نمی‌توان متغیرهای همنام داشت.
- در بحث فوق، یک استثنا وجود دارد و آن، زمانی است که بلوک‌های متداخل داشته باشیم. در این صورت می‌توانیم در بلوک داخلی، متغیری همنام با متغیری که در بلوک بیرونی تعریف شده، تعریف کنیم. در این حالت، استفاده از متغیر، بسته به محل استفاده از آن به آن متغیری اشاره می‌کند که در آن بلوک تعریف شده است.

قلمرو متغیرها

قلمرو متغیرها:

- قلمرو (scope) یک متغیر: محلهایی از برنامه است که آن متغیر قابل استفاده است، یعنی می‌توان با دستور انتساب به آن متغیر مقداری نسبت داد یا از محتوای ذخیره شده در آن متغیر استفاده کرد.
- هر متغیر از محل تعریف شدن تا آخر بلوکی که در آن تعریف شده است، قابل رویت و استفاده است و در خارج از محدوده قابل استفاده نیست.
- بلوک: مجموعه دستوراتی که با یک آکولاد باز شروع می‌شود و تا آکولاد بسته متناظر با آن ختم می‌شود.
- در یک قلمرو، نمی‌توان متغیرهای همنام داشت.
- در بحث فوق، یک استثنا وجود دارد و آن، زمانی است که بلوک‌های متداخل داشته باشیم. در این صورت می‌توانیم در بلوک داخلی، متغیری همنام با متغیری که در بلوک بیرونی تعریف شده، تعریف کنیم. در این حالت، استفاده از متغیر، بسته به محل استفاده از آن به آن متغیری اشاره می‌کند که در آن بلوک تعریف شده است.

قلمرو متغیرها

قلمرو متغیرها:

- قلمرو (scope) یک متغیر: محلهایی از برنامه است که آن متغیر قابل استفاده است، یعنی می‌توان با دستور انتساب به آن متغیر مقداری نسبت داد یا از محتوای ذخیره شده در آن متغیر استفاده کرد.
- هر متغیر از محل تعریف شدن تا آخر بلوکی که در آن تعریف شده است، قابل رویت و استفاده است و در خارج از محدوده قابل استفاده نیست.
- بلوک: مجموعه دستوراتی که با یک آکولاد باز شروع می‌شود و تا آکولاد بسته متناظر با آن ختم می‌شود.
- در یک قلمرو، نمی‌توان متغیرهای همنام داشت.
- در بحث فوق، یک استثنا وجود دارد و آن، زمانی است که بلوک‌های متداخل داشته باشیم. در این صورت می‌توانیم در بلوک داخلی، متغیری همنام با متغیری که در بلوک بیرونی تعریف شده، تعریف کنیم. در این حالت، استفاده از متغیر، بسته به محل استفاده از آن به آن متغیری اشاره می‌کند که در آن بلوک تعریف شده است.

قلمرو متغیرها

قلمرو متغیرها:

```
1  #include <iostream>
2  using namespace std;
3  int  main()
4  {
5      int  a=0;
6      {
7          int  a=1;
8          cout<<a;
9      }
10     cout<<a;
11     return  0;
12 }
```

مثال: خروجی برنامه ابتدا عدد ۱ و سپس عدد ۰ است.

طول عمر متغیرها

طول عمر متغیرها:

- **طول عمر یک متغیر: بازهٔ زمانی که به متغیر حافظه تخصیص داده شده است.**
- طول عمر متغیر معمول یا محلی متناسب با قلمرو آن است، یعنی از ابتدای بلوکی که متغیر تعریف شده است تا انتهای آن بلوک.
- با هر بار اجرای تابع، ممکن است حافظه‌ای در محل‌های مختلف به متغیرهای آن نسبت داده شود.
- متغیرهای ایستا (static): طول عمر آنها، از ابتدا تا انتهای اجرای برنامه است. تعریف این متغیرها با اضافه کردن کلمه کلیدی static در قبل از نوع متغیر انجام میشود. به عنوان مثال، static int x;
- اگر در یک اجرای تابع مقدار مثلاً ۱ در یک متغیر ایستا قرار گرفت، در فراخوانی بعدی تابع، این متغیر همچنان دارای همین مقدار خواهد بود. این متغیرها در بسیاری موارد مفید هستند.
- متغیرهای خارجی: متغیرهایی که خارج از توابع تعریف می‌شوند، دارای طول عمر مشابه متغیرهای ایستا هستند و قلمرو آنها نیز در سرتاسر برنامه است.

طول عمر متغیرها

طول عمر متغیرها:

- طول عمر یک متغیر: بازهٔ زمانی که به متغیر حافظه تخصیص داده شده است.
- طول عمر متغیر معمول یا محلی متناسب با قلمرو آن است، یعنی از ابتدای بلوکی که متغیر تعریف شده است تا انتهای آن بلوک.
- با هر بار اجرای تابع، ممکن است حافظه‌ای در محل‌های مختلف به متغیرهای آن نسبت داده شود.
- متغیرهای ایستا (static): طول عمر آنها، از ابتدا تا انتهای اجرای برنامه است. تعریف این متغیرها با اضافه کردن کلمه کلیدی static در قبل از نوع متغیر انجام میشود. به عنوان مثال، static int x;
- اگر در یک اجرای تابع مقدار مثلاً ۱ در یک متغیر ایستا قرار گرفت، در فراخوانی بعدی تابع، این متغیر همچنان دارای همین مقدار خواهد بود. این متغیرها در بسیاری موارد مفید هستند.
- متغیرهای خارجی: متغیرهایی که خارج از توابع تعریف می‌شوند، دارای طول عمر مشابه متغیرهای ایستا هستند و قلمرو آنها نیز در سرتاسر برنامه است.

طول عمر متغیرها

طول عمر متغیرها:

- طول عمر یک متغیر: بازهٔ زمانی که به متغیر حافظه تخصیص داده شده است.
- طول عمر متغیر معمول یا محلی متناسب با قلمرو آن است، یعنی از ابتدای بلوکی که متغیر تعریف شده است تا انتهای آن بلوک.
- با هر بار اجرای تابع، ممکن است حافظه‌ای در محل‌های مختلف به متغیرهای آن نسبت داده شود.
- متغیرهای ایستا (static): طول عمر آنها، از ابتدا تا انتهای اجرای برنامه است. تعریف این متغیرها با اضافه کردن کلمه کلیدی static در قبل از نوع متغیر انجام میشود. به عنوان مثال، static int x;
- اگر در یک اجرای تابع مقدار مثلاً ۱ در یک متغیر ایستا قرار گرفت، در فراخوانی بعدی تابع، این متغیر همچنان دارای همین مقدار خواهد بود. این متغیرها در بسیاری موارد مفید هستند.
- متغیرهای خارجی: متغیرهایی که خارج از توابع تعریف می‌شوند، دارای طول عمر مشابه متغیرهای ایستا هستند و قلمرو آنها نیز در سرتاسر برنامه است.

طول عمر متغیرها

طول عمر متغیرها:

- طول عمر یک متغیر: بازهٔ زمانی که به متغیر حافظه تخصیص داده شده است.
- طول عمر متغیر معمول یا محلی متناسب با قلمرو آن است، یعنی از ابتدای بلوکی که متغیر تعریف شده است تا انتهای آن بلوک.
- با هر بار اجرای تابع، ممکن است حافظه‌ای در محل‌های مختلف به متغیرهای آن نسبت داده شود.
- متغیرهای ایستا (static): طول عمر آنها، از ابتدا تا انتهای اجرای برنامه است. تعریف این متغیرها با اضافه کردن کلمه کلیدی static در قبل از نوع متغیر انجام میشود. به عنوان مثال، static int x;
- اگر در یک اجرای تابع مقدار مثلاً ۱ در یک متغیر ایستا قرار گرفت، در فراخوانی بعدی تابع، این متغیر همچنان دارای همین مقدار خواهد بود. این متغیرها در بسیاری موارد مفید هستند.
- متغیرهای خارجی: متغیرهایی که خارج از توابع تعریف می‌شوند، دارای طول عمر مشابه متغیرهای ایستا هستند و قلمرو آنها نیز در سرتاسر برنامه است.

طول عمر متغیرها

طول عمر متغیرها:

- طول عمر یک متغیر: بازهٔ زمانی که به متغیر حافظه تخصیص داده شده است.
- طول عمر متغیر معمول یا محلی متناسب با قلمرو آن است، یعنی از ابتدای بلوکی که متغیر تعریف شده است تا انتهای آن بلوک.
- با هر بار اجرای تابع، ممکن است حافظه‌ای در محل‌های مختلف به متغیرهای آن نسبت داده شود.
- متغیرهای ایستا (static): طول عمر آنها، از ابتدا تا انتهای اجرای برنامه است. تعریف این متغیرها با اضافه کردن کلمه کلیدی static در قبل از نوع متغیر انجام میشود. به عنوان مثال، static int x;
- اگر در یک اجرای تابع مقدار مثلاً ۱ در یک متغیر ایستا قرار گرفت، در فراخوانی بعدی تابع، این متغیر همچنان دارای همین مقدار خواهد بود. این متغیرها در بسیاری موارد مفید هستند.
- متغیرهای خارجی: متغیرهایی که خارج از توابع تعریف می‌شوند، دارای طول عمر مشابه متغیرهای ایستا هستند و قلمرو آنها نیز در سرتاسر برنامه است.

طول عمر متغیرها

طول عمر متغیرها:

- طول عمر یک متغیر: بازهٔ زمانی که به متغیر حافظه تخصیص داده شده است.
- طول عمر متغیر معمول یا محلی متناسب با قلمرو آن است، یعنی از ابتدای بلوکی که متغیر تعریف شده است تا انتهای آن بلوک.
- با هر بار اجرای تابع، ممکن است حافظه‌ای در محل‌های مختلف به متغیرهای آن نسبت داده شود.
- متغیرهای ایستا (static): طول عمر آنها، از ابتدا تا انتهای اجرای برنامه است. تعریف این متغیرها با اضافه کردن کلمه کلیدی static در قبل از نوع متغیر انجام میشود. به عنوان مثال، `static int x;`
- اگر در یک اجرای تابع مقدار مثلاً ۱ در یک متغیر ایستا قرار گرفت، در فراخوانی بعدی تابع، این متغیر همچنان دارای همین مقدار خواهد بود. این متغیرها در بسیاری موارد مفید هستند.
- متغیرهای خارجی: متغیرهایی که خارج از توابع تعریف می‌شوند، دارای طول عمر مشابه متغیرهای ایستا هستند و قلمرو آنها نیز در سرتاسر برنامه است.

الگوریتم و توابع بازگشتی

الگوریتم و توابع بازگشتی:

- یکی از روش‌های مرسوم در حل مسائل، روش بازگشتی است.
- در ریاضی، بعضاً از خود مفهوم برای تعریف خودش استفاده می‌شود.
- مثال: تعریف فاکتوریل: تعریف مستقیم: $n! = n \times (n-1) \times \cdots \times 2 \times 1$
- تعریف بازگشتی: $n! = \begin{cases} 1 & \text{اگر } n = 0 \\ n \times (n-1)! & \text{در غیر اینصورت} \end{cases}$
- برای محاسبهٔ فاکتوریل با استفاده از فرمول بازگشتی، تابعی که برای محاسبهٔ فاکتوریل می‌نویسیم، خودش را فراخوانی کند.
- الگوریتمی (تابعی) که خودش، خودش را فراخوانی کند را الگوریتم (تابع) بازگشتی می‌نامیم.
- تابع بازگشتی، ممکن است مستقیم، خودش، خودش را فراخوانی نکند و با واسطه خودش را فراخوانی کند.

الگوریتم و توابع بازگشتی

الگوریتم و توابع بازگشتی:

- یکی از روش‌های مرسوم در حل مسائل، روش بازگشتی است.
- در ریاضی، بعضاً از خود مفهوم برای تعریف خودش استفاده می‌شود.
- مثال: تعریف فاکتوریل: تعریف مستقیم: $n! = n \times (n-1) \times \cdots \times ۲ \times ۱$
- تعریف بازگشتی: $n! = \begin{cases} ۱ & \text{اگر } n = ۰ \\ n \times (n-1)! & \text{در غیر اینصورت} \end{cases}$
- برای محاسبهٔ فاکتوریل با استفاده از فرمول بازگشتی، تابعی که برای محاسبهٔ فاکتوریل می‌نویسیم، خودش را فراخوانی کند.
- الگوریتمی (تابعی) که خودش، خودش را فراخوانی کند را الگوریتم (تابع) بازگشتی می‌نامیم.
- تابع بازگشتی، ممکن است مستقیم، خودش، خودش را فراخوانی نکند و با واسطه خودش را فراخوانی کند.

الگوریتم و توابع بازگشتی

الگوریتم و توابع بازگشتی:

- یکی از روش‌های مرسوم در حل مسائل، روش بازگشتی است.
- در ریاضی، بعضاً از خود مفهوم برای تعریف خودش استفاده می‌شود.
- مثال: تعریف فاکتوریل: تعریف مستقیم: $n! = n \times (n-1) \times \cdots \times 2 \times 1$
- تعریف بازگشتی: $n! = \begin{cases} 1 & \text{اگر } n = 0 \\ n \times (n-1)! & \text{در غیر اینصورت} \end{cases}$
- برای محاسبهٔ فاکتوریل با استفاده از فرمول بازگشتی، تابعی که برای محاسبهٔ فاکتوریل می‌نویسیم، خودش را فراخوانی کند.
- الگوریتمی (تابعی) که خودش، خودش را فراخوانی کند را الگوریتم (تابع) بازگشتی می‌نامیم.
- تابع بازگشتی، ممکن است مستقیم، خودش، خودش را فراخوانی نکند و با واسطه خودش را فراخوانی کند.

الگوریتم و توابع بازگشتی

الگوریتم و توابع بازگشتی:

- یکی از روش‌های مرسوم در حل مسائل، روش بازگشتی است.
- در ریاضی، بعضاً از خود مفهوم برای تعریف خودش استفاده می‌شود.
- مثال: تعریف فاکتوریل: تعریف مستقیم: $n! = n \times (n-1) \times \cdots \times ۲ \times ۱$
- تعریف بازگشتی: $n! = \begin{cases} ۱ & \text{اگر } n = ۰ \\ n \times (n-1)! & \text{در غیر اینصورت} \end{cases}$
- برای محاسبهٔ فاکتوریل با استفاده از فرمول بازگشتی، تابعی که برای محاسبهٔ فاکتوریل می‌نویسیم، خودش را فراخوانی کند.
- الگوریتمی (تابعی) که خودش، خودش را فراخوانی کند را الگوریتم (تابع) بازگشتی می‌نامیم.
- تابع بازگشتی، ممکن است مستقیم، خودش، خودش را فراخوانی نکند و با واسطه خودش را فراخوانی کند.

الگوریتم و توابع بازگشتی

الگوریتم و توابع بازگشتی:

- یکی از روش‌های مرسوم در حل مسائل، روش بازگشتی است.
- در ریاضی، بعضاً از خود مفهوم برای تعریف خودش استفاده می‌شود.
- مثال: تعریف فاکتوریل: تعریف مستقیم: $n! = n \times (n-1) \times \cdots \times 2 \times 1$
- تعریف بازگشتی:

$$n! = \begin{cases} 1 & \text{اگر } n = 0 \\ n \times (n-1)! & \text{در غیر اینصورت} \end{cases}$$

- برای محاسبهٔ فاکتوریل با استفاده از فرمول بازگشتی، تابعی که برای محاسبهٔ فاکتوریل می‌نویسیم، خودش را فراخوانی کند.
- الگوریتمی (تابعی) که خودش، خودش را فراخوانی کند را الگوریتم (تابع) بازگشتی می‌نامیم.
- تابع بازگشتی، ممکن است مستقیم، خودش، خودش را فراخوانی نکند و با واسطه خودش را فراخوانی کند.

الگوریتم و توابع بازگشتی

الگوریتم و توابع بازگشتی:

- یکی از روش‌های مرسوم در حل مسائل، روش بازگشتی است.
- در ریاضی، بعضاً از خود مفهوم برای تعریف خودش استفاده می‌شود.
- مثال: تعریف فاکتوریل: تعریف مستقیم: $n! = n \times (n-1) \times \cdots \times 2 \times 1$
- تعریف بازگشتی: $n! = \begin{cases} 1 & \text{اگر } n = 0 \\ n \times (n-1)! & \text{در غیر اینصورت} \end{cases}$
- برای محاسبهٔ فاکتوریل با استفاده از فرمول بازگشتی، تابعی که برای محاسبهٔ فاکتوریل می‌نویسیم، خودش را فراخوانی کند.
- الگوریتمی (تابعی) که خودش، خودش را فراخوانی کند را الگوریتم (تابع) بازگشتی می‌نامیم.
- تابع بازگشتی، ممکن است مستقیم، خودش، خودش را فراخوانی نکند و با واسطه خودش را فراخوانی کند.

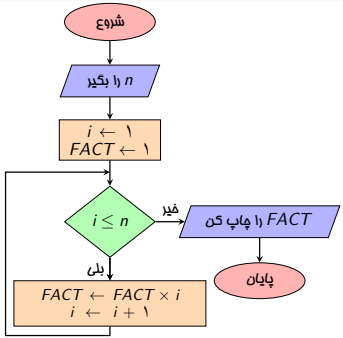
الگوریتم و توابع بازگشتی

الگوریتم و توابع بازگشتی:

- یکی از روش‌های مرسوم در حل مسائل، روش بازگشتی است.
- در ریاضی، بعضاً از خود مفهوم برای تعریف خودش استفاده می‌شود.
- مثال: تعریف فاکتوریل: تعریف مستقیم: $n! = n \times (n-1) \times \cdots \times 2 \times 1$
- تعریف بازگشتی: $n! = \begin{cases} 1 & \text{اگر } n = 0 \\ n \times (n-1)! & \text{در غیر اینصورت} \end{cases}$
- برای محاسبهٔ فاکتوریل با استفاده از فرمول بازگشتی، تابعی که برای محاسبهٔ فاکتوریل می‌نویسیم، خودش را فراخوانی کند.
- الگوریتمی (تابعی) که خودش، خودش را فراخوانی کند را الگوریتم (تابع) بازگشتی می‌نامیم.
- تابع بازگشتی، ممکن است مستقیم، خودش، خودش را فراخوانی نکند و با واسطه خودش را فراخوانی کند.

مثال ۲۲

برنامه‌ای بنویسید که یک عدد طبیعی را از ورودی دریافت کند و فاکتوریل آن را محاسبه و چاپ کند.

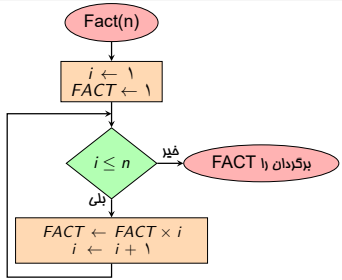


برنامه:

```
1  #include <iostream>
2  using namespace std;
3  int  main()
4  {
5       int  i  ,  n;
6       long  int  FACT =1 ;
7       cout<<"Enter n:";
8       cin  >> n;
9       for  (  i=1  ;  i  <=n  ;  i++)
10           FACT*=i;
11      cout<<" \n Factorial  is  :"<< FACT;
12      return  0;
13 }
```

مثال ۲۳

زیرالگوریتم (تابعی) بنویسید که یک عدد طبیعی را به عنوان پارامتر دریافت کند و فاکتوریل آن را محاسبه کرده و برگرداند.



برنامه:

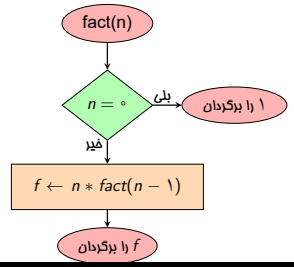
```
1  #include <iostream>
2  using namespace std;
3  long long Fact(int  n)
4  {
5       int  i ;
6       long long FACT =1 ;
7       for  ( i=1 ;  i  <=n ;  i++)
8            FACT*=i;
9       return  FACT;
10 }
11 int  main()
12 {
13      int  n;
14      cout<<"Enter n:";
15      cin  >> n;
16      cout<<" \n  Factorial  is  :"<< Fact(n);
17      return  0;
18 }
```

مثال ۲۴

زیرالگوریتمی (تابعی) بازگشتی بنویسید که یک عدد طبیعی را به عنوان پارامتر دریافت کند و فاکتوریل آن را محاسبه کرده و برگرداند.

تعریف بازگشتی فاکتوریل:

$$n! = \begin{cases} ۱ & \text{اگر } n = ۰ \\ n \times (n-۱)! & \text{در غیر اینصورت} \end{cases}$$



برنامه:

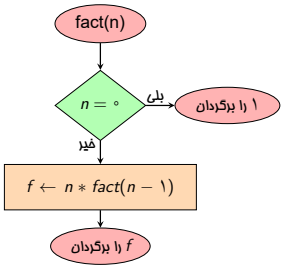
```
1  #include <iostream>
2  using namespace std;
3  long long fact(int n)
4  {
5      long long f;
6      if (n == 0)
7          return 1;
8      else
9      {
10         f=n*fact(n-1);
11         return f;
12     }
13 }
14 int main()
15 {
16     int n;
17     cout<<"Enter n:";
18     cin >> n;
19     cout<<" \n Factorial is :"<< fact(n);
20     return 0;
21 }
```

مثال ۲۴

زیرالگوریتمی (تابعی) بازگشتی بنویسید که یک عدد طبیعی را به عنوان پارامتر دریافت کند و فاکتوریل آن را محاسبه کرده و برگرداند.

تعریف بازگشتی فاکتوریل:

$$n! = \begin{cases} ۱ & \text{اگر } n = ۰ \\ n \times (n-۱)! & \text{در غیر اینصورت} \end{cases}$$



برنامه:

```
1  #include <iostream>
2  using namespace std;
3  long int fact(long int n)
4  {
5       if (n == 0)
6            return 1;
7       else
8            return n*fact(n-1);
9  }
10 int main()
11 {
12      int n;
13      cout<<"Enter n:";
14      cin >> n;
15      cout<<" \n Factorial is :"<< fact(n);
16      return 0;
17 }
```

```
1  #include <iostream>
2  using namespace std;
3  long int Fib( int n)
4  {
5       if (n == 1 || n == 2)
6            return 1;
7       else
8            return Fib(n−1)+Fib(n−2);
9  }
10 int main()
11 {
12      int n;
13      cout<<"Enter n:";
14      cin >> n;
15      cout<<" \n nth element of Fibonacci
                sequence is :"<< Fib(n);
16      return 0;
17 }
```

مثال ۲۵

مطلوب است، تابع بازگشتی برای محاسبه جملهٔ nام دنبالهٔ فیبوناچی.

حل:

تعریف بازگشتی دنبالهٔ فیبوناچی:

$$Fib(n) = \begin{cases} ۱ & \text{اگر } n = ۱ \text{ یا } ۲ \\ Fib(n-۱) + Fib(n-۲) & \text{اگر } n \geq ۳ \end{cases}$$

مثال ۲۶

تابعی بازگشتی بنویسید که یک آرایه از اعداد اعشاری و تعداد عناصر آرایه را به عنوان آرگومان بگیرد و مجموع اعداد موجود در آرایه را محاسبه کرده و برگرداند.

حل:

- برای حل مسئله به صورت بازگشتی، باید الگوریتمی بازگشتی برای آن ارائه کرد.
- لذا از ساده‌ترین حالت شروع می‌کنیم و سعی خواهیم کرد که فرمولی بازگشتی برای حل مسئله ارائه کنیم.
- بسیار مواردی وجود دارد که راه حل مستقیم آن بسیار پیچیده است ولی با استفاده از روش بازگشتی به سادگی مسئله حل می‌شود.

مثال ۲۶

تابعی بازگشتی بنویسید که یک آرایه از اعداد اعشاری و تعداد عناصر آرایه را به عنوان آرگومان بگیرد و مجموع اعداد موجود در آرایه را محاسبه کرده و برگرداند.

حل:

- برای حل مسئله به صورت بازگشتی، باید الگوریتمی بازگشتی برای آن ارائه کرد.
- لذا از ساده‌ترین حالت شروع می‌کنیم و سعی خواهیم کرد که فرمولی بازگشتی برای حل مسئله ارائه کنیم.
- بسیار مواردی وجود دارد که راه حل مستقیم آن بسیار پیچیده است ولی با استفاده از روش بازگشتی به سادگی مسئله حل می‌شود.

مثال ۲۶

تابعی بازگشتی بنویسید که یک آرایه از اعداد اعشاری و تعداد عناصر آرایه را به عنوان آرگومان بگیرد و مجموع اعداد موجود در آرایه را محاسبه کرده و برگرداند.

حل:

- برای حل مسئله به صورت بازگشتی، باید الگوریتمی بازگشتی برای آن ارائه کرد.
- لذا از ساده‌ترین حالت شروع می‌کنیم و سعی خواهیم کرد که فرمولی بازگشتی برای حل مسئله ارائه کنیم.
- بسیار مواردی وجود دارد که راه حل مستقیم آن بسیار پیچیده است ولی با استفاده از روش بازگشتی به سادگی مسئله حل می‌شود.

مثال ۲۶

تابعی بازگشتی بنویسید که یک آرایه از اعداد اعشاری و تعداد عناصر آرایه را به عنوان آرگومان بگیرد و مجموع اعداد موجود در آرایه را محاسبه کرده و برگرداند.

حل:

- برای حل مسئله به صورت بازگشتی، باید الگوریتمی بازگشتی برای آن ارائه کرد.
- لذا از ساده‌ترین حالت شروع می‌کنیم و سعی خواهیم کرد که فرمولی بازگشتی برای حل مسئله ارائه کنیم.
- بسیار مواردی وجود دارد که راه حل مستقیم آن بسیار پیچیده است ولی با استفاده از روش بازگشتی به سادگی مسئله حل می‌شود.

$$Sum(A[], n) = \begin{cases} A[\circ] & \text{اگر } n = ۱ \\ Sum(A[], n-۱) + A[n-۱] & \text{اگر } n \geq ۲ \end{cases}$$

مثال ۲۶

تابعی بازگشتی بنویسید که یک آرایه از اعداد اعشاری و تعداد عناصر آرایه را به عنوان آرگومان بگیرد و مجموع اعداد موجود در آرایه را محاسبه کرده و برگرداند.

$$Sum(A[], n) = \begin{cases} A[0] & n = 1 \text{ اگر} \\ Sum(A[], n-1) + A[n-1] & n \geq 2 \text{ اگر} \end{cases}$$

```
1  double Sum_Array(double A[ ], int n)
2  {
3       if (n == 1)
4            return  A[0];
5       else
6            return  Sum_Array(A,n−1)+A[n−1];
7  }
```

حل:

- برای حل مسئله به صورت بازگشتی، باید الگوریتمی بازگشتی برای آن ارائه کرد.
- لذا از ساده‌ترین حالت شروع می‌کنیم و سعی خواهیم کرد که فرمولی بازگشتی برای حل مسئله ارائه کنیم.
- بسیار مواردی وجود دارد که راه حل مستقیم آن بسیار پیچیده است ولی با استفاده از روش بازگشتی به سادگی مسئله حل می‌شود.

مثال ۲۷

جستجوی دودویی یک عدد را در یک آرایهٔ مرتب به صورت بازگشتی.

حل:

روش بازگشتی جستجوی دودویی:

- عنصر مورد جستجو را با عضو وسط آرایه مقایسه می‌کنیم. اگر برابر است که جستجو پایان می‌یابد.
- در غیر این صورت در صورتی که مقدار مورد جستجو بزرگ‌تر از عضو وسط است، باید عضو را در قسمت دوم آرایه جستجو کرد.
- در غیر این صورت، جستجوی دودویی روی نیمهٔ اول آرایه ادامه پیدا می‌کند.

مثال ۲۷

جستجوی دودویی یک عدد را در یک آرایهٔ مرتب به صورت بازگشتی.

حل:

روش بازگشتی جستجوی دودویی:

- عنصر مورد جستجو را با عضو وسط آرایه مقایسه می‌کنیم. اگر برابر است که جستجو پایان می‌یابد.
- در غیر این صورت در صورتی که مقدار مورد جستجو بزرگ‌تر از عضو وسط است، باید عضو را در قسمت دوم آرایه جستجو کرد.
- در غیر این صورت، جستجوی دودویی روی نیمهٔ اول آرایه ادامه پیدا می‌کند.

مثال ۲۷

جستجوی دودویی یک عدد را در یک آرایهٔ مرتب به صورت بازگشتی.

حل:

روش بازگشتی جستجوی دودویی:

- عنصر مورد جستجو را با عضو وسط آرایه مقایسه می‌کنیم. اگر برابر است که جستجو پایان می‌یابد.
- در غیر این صورت در صورتی که مقدار مورد جستجو بزرگ‌تر از عضو وسط است، باید عضو را در قسمت دوم آرایه جستجو کرد.
- در غیر این صورت، جستجوی دودویی روی نیمهٔ اول آرایه ادامه پیدا می‌کند.

مثال ۲۷

جستجوی دودویی یک عدد را در یک آرایهٔ مرتب به صورت بازگشتی.

حل:

روش بازگشتی جستجوی دودویی:

- عنصر مورد جستجو را با عضو وسط آرایه مقایسه می‌کنیم. اگر برابر است که جستجو پایان می‌یابد.
- در غیر این صورت در صورتی که مقدار مورد جستجو بزرگ‌تر از عضو وسط است، باید عضو را در قسمت دوم آرایه جستجو کرد.
- در غیر این صورت، جستجوی دودویی روی نیمهٔ اول آرایه ادامه پیدا می‌کند.

```
1  int  Bin_Search_rec(double A[500],
2                      double x,
3                       int  i,
4                       int  j)
5  {
6    int  mid;
7    if (i  == j)
8      if (x == A[i])
9        return  1;
10     else
11       return  0;
12   mid=(i+j)/2;
13   if (x == A[mid])
14     return  1;
15   if (x < A[mid])
16     return  Bin_Search_rec(A,x,i,mid-1);
17   if (x > A[mid])
18     return  Bin_Search_rec(A,x,mid+1,j);
19 }
```


باآرزوی موفقیت
وبهروزی