به نام خدا

مفاهیم تکمیلی درس

مبانی کامپیوتر و برنامه‌سازی

پاییز 1404

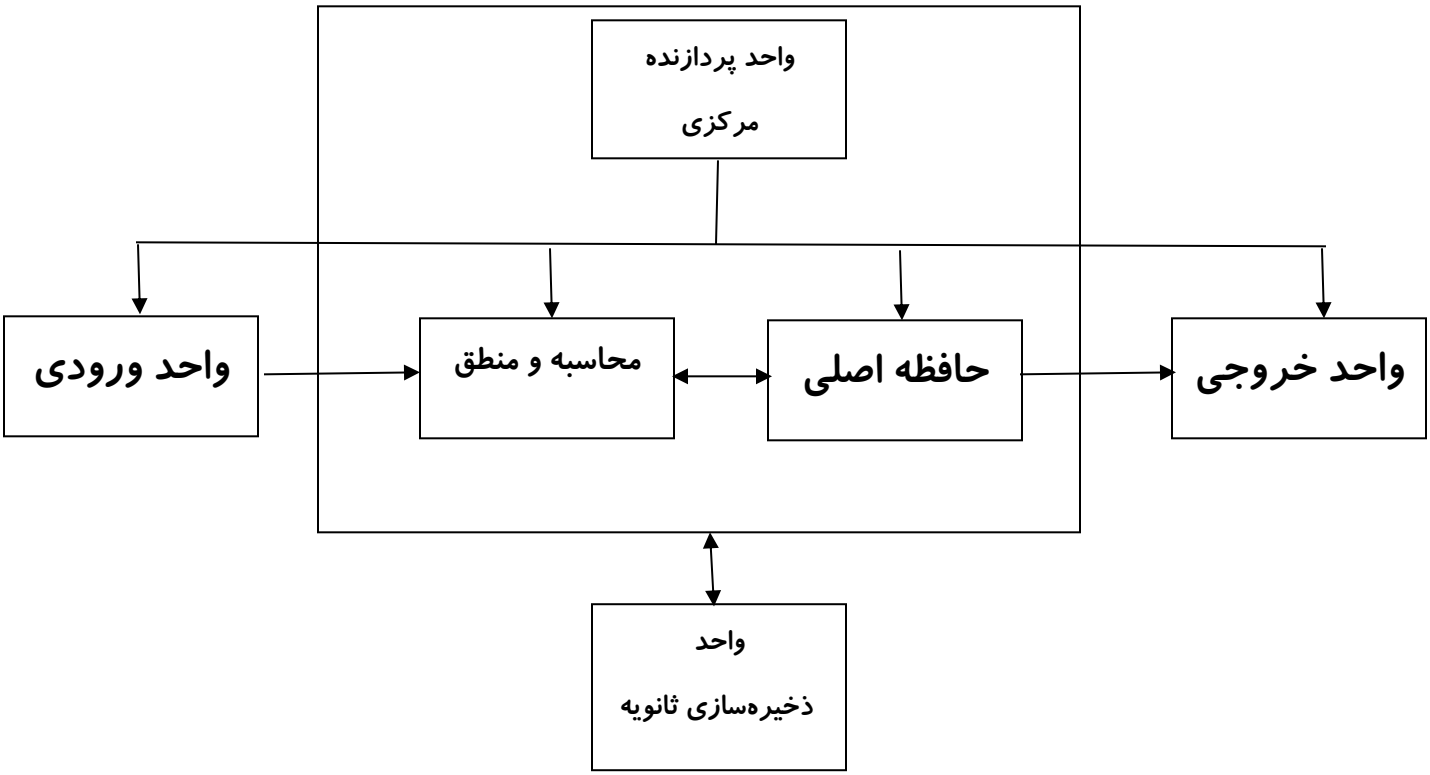
تعریفی از کامپیوتر: کامپیوتر ابزاری است که داده را از ورودی دریافت کرده و سپس تحت مجموعه‌ای از دستورالعمل‌ها (برنامه‌های کامپیوتری) آن را مورد پردازش قرار داده و اطلاعات حاصل را به خروجی ارسال می‌نماید.

فارغ از تفاوت‌های فیزیکی، هر کامپیوتر دارای شش واحد اصلی است که عبارت هستند از:

- واحد ورودی: این قسمت، واحد دریافت است و برای به دست‌آوردن داده و برنامه کامپیوتری از طریق وسایل ورودی مانند ماوس و صفحه کلید و قرار دادن آن‌ها در دسترس سایر واحدها برای پردازش اطلاعات می‌باشد.
- واحد خروجی: این واحد مسئولیت دریافت نتایج پردازش شده و تحویل آن‌ها به شکلی قابل درک برای کاربر یا یک دستگاه خارجی را بر عهده دارد. هدف اصلی این واحد، ایجاد یک پل ارتباطی بین واحدهای داخلی کامپیوتر و کاربر نهایی است.
- واحد حافظه (اولیه): این قسمت مانند یک انبار (موقت) است و دارای سرعت دسترسی نسبتا بالایی است. اطلاعات وارد شده از واحد ورودی در این قسمت نگهداری می‌شوند. همچنین واحد حافظه می‌تواند اطلاعات پردازش شده را در خود نگهداری کند تا زمانی که اطلاعات بتواند بر روی دستگاه خروجی (به وسیله واحد خروجی) قرار گیرد.
- واحد محاسبه و منطق: این بخش مسئول انجام اعمال محاسباتی مانند جمع، تفریق، ضرب و تقسیم و اعمال منطقی است. این بخش همچنین شامل مکانیزم‌های تصمیم‌گیری می‌باشد.
- واحد پردازش مرکزی: این قسمت واحد اجرایی کامپیوتر به حساب می‌آید و وظیفه هماهنگ کردن کامپیوتر و نظارت بر اجرای عملیات توسط سایر قسمت‌ها را بر عهده دارد. CPU به واحد ورودی اعلام می‌کند در چه زمانی باید اطلاعات به واحد حافظه وارد شود، به ALU هم اعلام می‌کند در چه زمانی اطلاعات از حافظه برداشته و به‌کار گرفته شود و به واحد خروجی اعلام می‌کند در چه زمانی اطلاعات از حافظه به واحد خروجی مشخص شده ارسال شود.
- واحد ذخیره‌سازی ثانویه: این قسمت نیز مانند انبار (دائمی) کامپیوتر است که دارای طول عمر و ظرفیت بالایی می‌باشد. هزینه اجزای این واحد نسبت به واحد حافظه اولیه بسیار کم‌تر است. اطلاعات ذخیره‌شده در این واحد تا زمانی که توسط کاربر حذف نشوند، برای استفاده در دسترس خواهند بود.

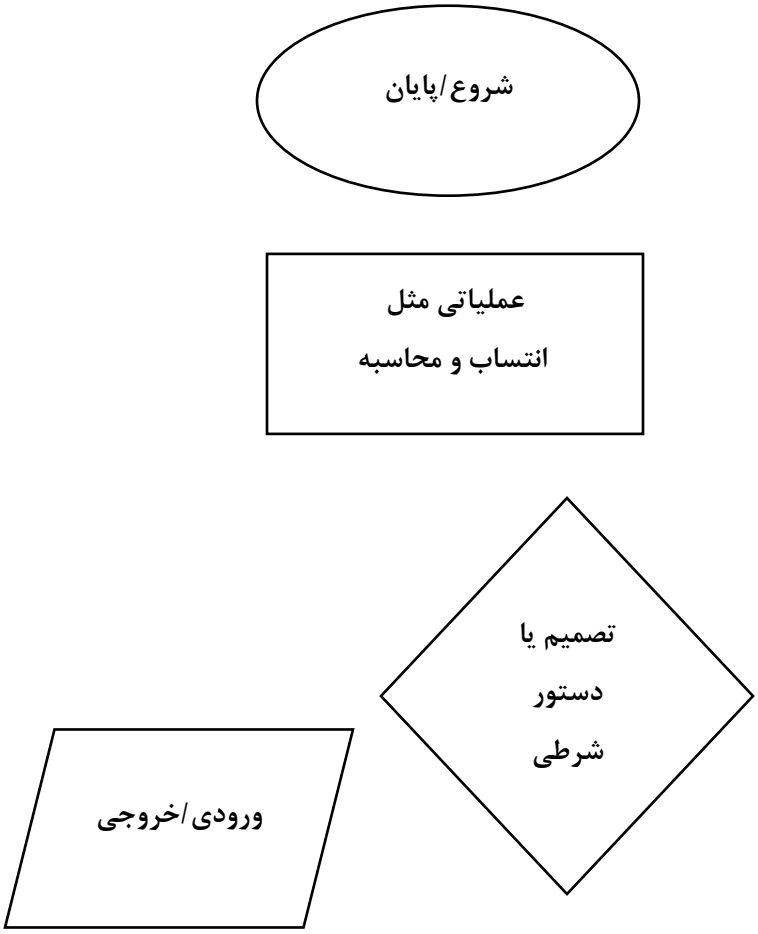
ثبات (رجیستر): ثبات‌ها حافظه‌های بسیار کوچک و فوق‌العاده سریع و با ظرفیت بسیار محدود هستند که درون هسته مرکزی پردازنده تعبیه شده‌اند. نقش و هدف ثبات این است که پردازنده به‌جای مراجعه مکرر به حافظه اصلی، داده‌های در حال پردازش را در نزدیکی خود قرار دهد تا کارها با بیشترین سرعت ممکن جریان داشته باشد.

حافظه کش: یک حافظه سریع با ظرفیت محدود که به عنوان میانجی بین پردازنده و حافظه عمل می‌کند. هدف اصلی کش، کاهش تاخیر دسترسی به داده‌ها و در نتیجه افزایش قابل توجه کارایی سیستم است.



تعریف الگوریتم: مجموعه‌ای متناهی و مرتب از دستورالعمل‌های دقیق، پایان‌پذیر و بدون ابهام که برای حل مسئله‌ای مشخص به کار می‌رود.

مراحل انجام یک فرایند یا الگوریتم را می‌توان با استفاده از نمودار تصویری به نام فلوچارت به ترتیب و به وضوح نمایش داد. به عبارتی دیگر، فلوچارت یک نقشه راه برای حل یک مسئله است. معمولا از علائم و اشکال استانداردی برای طراحی و رسم فلوچارت استفاده می‌شود.



مثال:

الگوریتم تشخیص اول بودن یک عدد

برای تشخیص اول بودن یک عدد مانند n کافی است که بخش‌پذیری آن بر هر عدد صحیح i بین 2 تا n/2 بررسی شود. برای بررسی این موضوع از یک حلقه تکرار استفاده می‌کنیم.

1. **شروع**
2. **n را بگیر.**
3. **i=2**
4. **اگر i<=n/2 آنگاه**

5 اگر باقیمانده تقسیم n بر i برابر با صفر است آنگاه

6 "Not Prime" را چاپ کن.

7 پایان

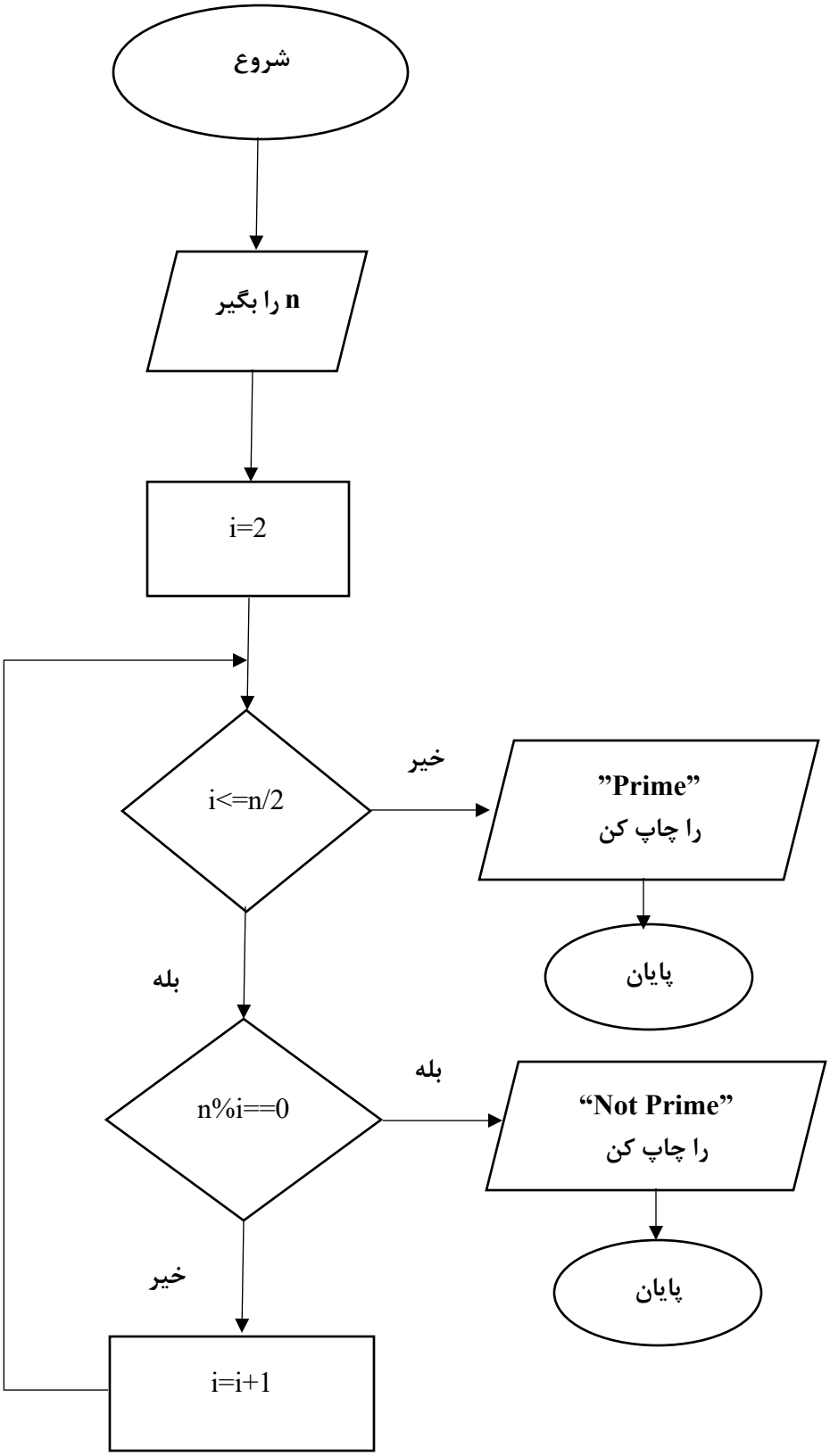
8 در غیر این صورت

9 i=i+1

10 به مرحله 4 برو

11 پایان شرط

12. **پایان شرط**
13. **"Prime" را چاپ کن**
14. **پایان**


شروع
n را بگیر
i=2
i<=n/2
خیر
"Prime"
را چاپ کن
پایان
بله
n%i==0
بله
"Not Prime"
را چاپ کن
پایان
خیر
i=i+1

الگوریتم محاسبه فاکتوریل یک عدد

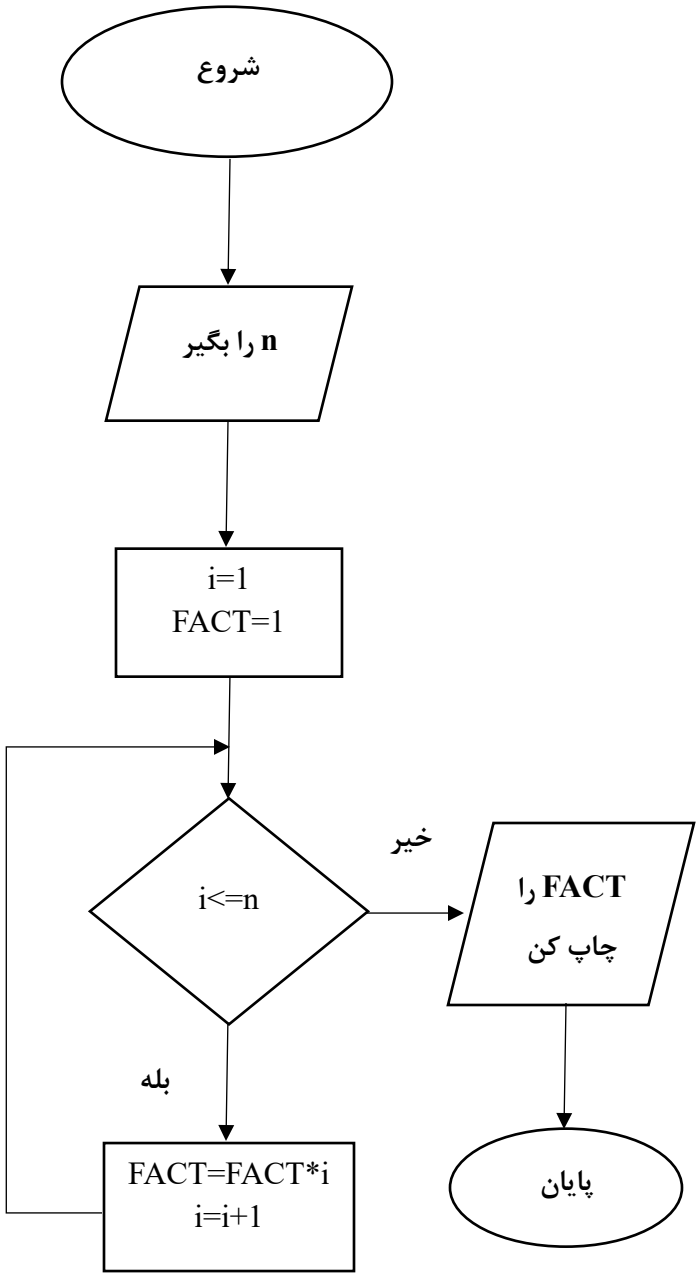
برای حل این مسئله باید حاصلضرب اعداد یک تا n را محاسبه کنیم. برای این منظور، از یک شمارنده مانند i استفاده می‌کنیم که مقدار اولیه آن 1 است و در یک حلقه تکرار، هربار یکی به آن اضافه می‌شود و هربار عدد i را در یک متغیر که برای نگهداری !n درنظر گرفته شده است و مقدار اولیه آن 1 است ضرب می‌کنیم. این روند تا وقتی که i کمتر یا مساوی n است تکرار می‌شود و به محض اینکه i بزرگتر از n شود، تکرار قطع می‌شود.

1. **شروع**
2. **n را بگیر**
3. **i=1 و FACT=1**
4. **اگر i<=n آنگاه**

5 FACT=FACT*i
6 i=i+1
7 به مرحله 4 برو

8. **پایان شرط**
9. **FACT را چاپ کن**
10. **پایان**

تذکر: کلمه FACT صرفا یک کلمه دلخواه به عنوان متغیر است. (مانند n,i,...)


شروع
n را بگیر
i=1
FACT=1
i<=n
خیر
FACT را
چاپ کن
بله
FACT=FACT*i
i=i+1
پایان

الگوریتم محاسبه بزرگترین مقسوم‌علیه مشترک دو عدد طبیعی

می‌دانیم ب.م.م دو عدد A و B بزرگترین عدد صحیحی است که A و B هر دو بر آن بخش‌پذیر باشند. بدیهی است که ب.م.م دو عدد از هر عدد کوچکتر یا مساوی است. لذا برای حل این مسئله، از کوچکترین عدد بین A و B شروع کرده و بررسی می‌کنیم آیا هر دو عدد بر آن بخش‌پذیر است یا خیر. اگر بخش‌پذیر باشد، این عدد همان ب.م.م است. در غیر این صورت، از این عدد یکی کم کرده و دوباره بخش‌پذیری را بررسی می‌کنیم. اولین عددی که هر دوی A و B بر آن بخش‌پذیر باشند، همان ب.م.م است. بر این مبنا، الگوریتم زیر را برای حل این مسئله مینویسیم. باید توجه داشت که در هربار تکرار، یکی از عددی که برای ب.م.م بودن بررسی می‌کنیم کم می‌شود و این روند حتما پایان‌پذیر است زیرا در نهایت در جایی عدد مورد بررسی به یک می‌رسد که مطمئنا هر دو عدد بر آن بخش‌پذیر است. این روند وقتی اتفاق می‌افتد که دو عدد نسبت به هم اول باشند.

1. **شروع**
2. **A و B را بگیر**
3. **اگر A>B آنگاه**

i=B 4

5 در غیر اینصورت

i=A 6

7 پایان شرط

8 اگر A%i != 0 یا B%i != 0 آنگاه

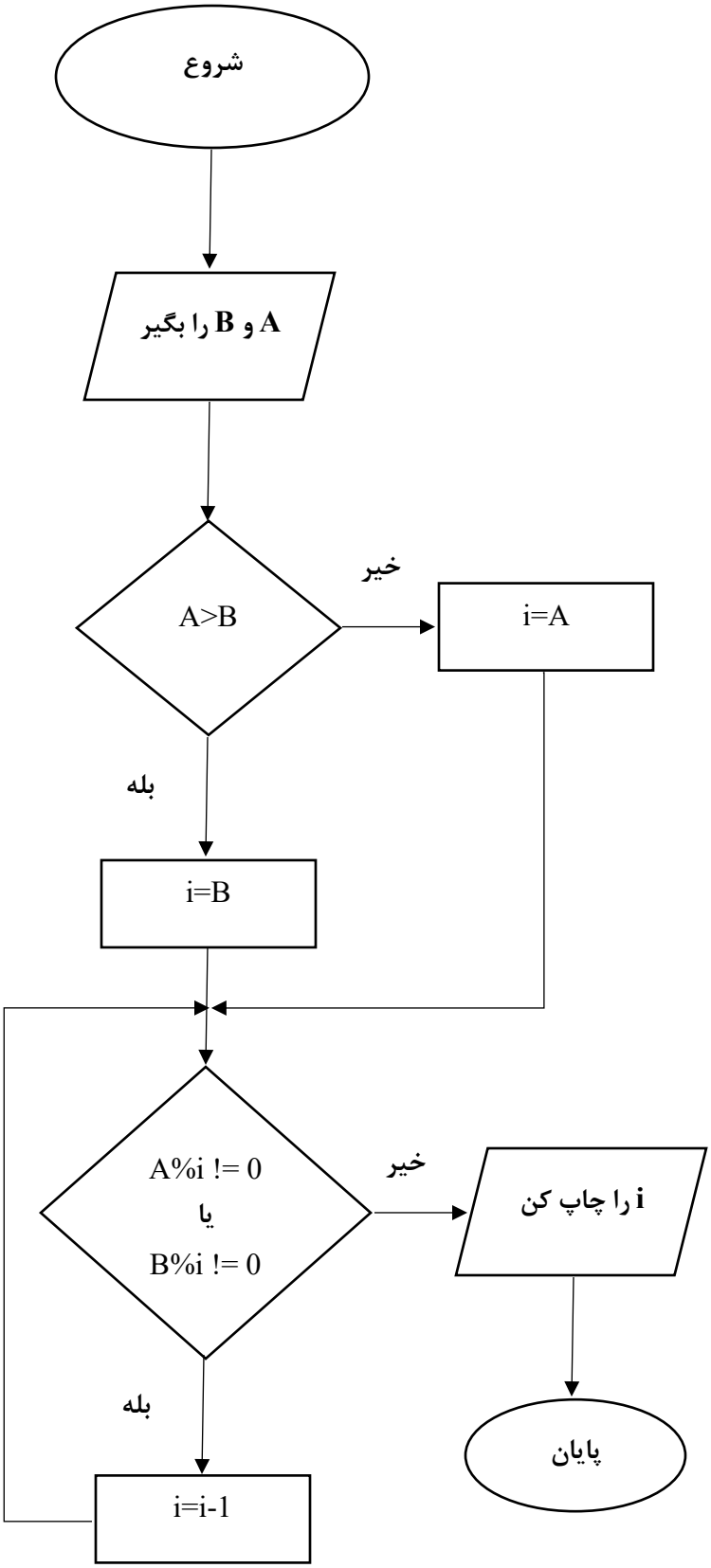
i=i-1 9

10 به مرحله 8 برو

11 پایان شرط

12 i را چاپ کن

13 پایان


شروع
A و B را بگیر
A>B
خیر
i=A
بله
i=B
A%i != 0
یا
B%i != 0
خیر
i را چاپ کن
بله
پایان
i=i-1

معرفی زبان C++

سی پلاس پلاس یکی از زبان‌های برنامه‌نویسی محبوب و قدرتمند است که توسط بیارنه استراستروپ به عنوان توسعه‌ای بر زبان سی ایجاد شد. این زبان **چندسکویی** امکان تولید برنامه‌های پرسرعت و کارآمد را در حوزه‌های مختلف از جمله سیستم‌عامل‌ها، رابط‌های گرافیکی و سیستم‌های نهفته فراهم می‌کند. سی پلاس پلاس با ارائه کنترل بالا بر منابع و حافظه، ساختار روشنی برای برنامه‌نویسی شیءگرا ایجاد می‌کند که موجب قابلیت استفاده مجدد از کد و کاهش هزینه‌های توسعه می‌شود. این زبان که از نظر ساختاری به زبان‌های سی، سی‌شارپ و جاوا نزدیک است، نه تنها یادگیری آسانی دارد، بلکه با قابلیت انتقال آسان بین پلتفرم‌های مختلف، گزینه‌ای ایده‌آل برای برنامه‌نویسان محسوب می‌شود.

انواع محیط‌های برنامه‌نویسی برای کار با C++

در سطوح پایه‌ای و مقدماتی برای کار با C++ می‌توان از انواع محیط‌های برنامه‌نویسی زیر استفاده کرد:

- محیط توسعه یکپارچه (IDE)

این محیط‌ها، کامل‌ترین و متداول‌ترین انتخاب برای توسعه حرفه‌ای و آموزشی هستند و معمولا تمامی ابزار لازم برای کارهای مختلف را فراهم می‌نمایند. برخی محیط‌های موجود در این دسته، روی یک یا چند زبان خاص تمرکز دارند، در حالی که برخی دیگر نیز عمومی‌تر هستند و برای کار با تعداد زبان‌های زیاد و متنوعی طراحی شده‌اند. از جمله IDE های متداول و مطلوب که می‌توان در آن‌ها به راحتی با C++ کار کرد عبارت هستند از Visual Studio، Code Blocks، Clion و Dev C++.

- ویرایشگر کد

در این محیط، از یک ویرایشگر متن سبک ولی قدرتمند برای نوشتن کد استفاده می‌شود اما کامپایل و اجرا از طریق خط فرمان انجام می‌شود و نیاز به نصب یا استفاده از ابزارهای بیرونی دارد. از ویژگی‌های

کلیدی این محیط‌ها می‌توان به کنترل بالا بر روی فرایند کامپایل در کنار سبکی و سرعت بالا اشاره کرد. VS Code ، Notepad++ و Sublime Text از جمله ویرایشگرهای کد مناسب و محبوب هستند.

- محیط‌های توسعه آنلاین

این محیط‌ها شرایطی را فراهم می‌کنند تا بدون نصب نرم‌افزاری بر روی کامپیوتر شخصی، کد را نوشته و آن را کامپایل و اجرا نماییم. این محیط‌ها بخصوص برای تست کدهای نسبتا کوچک، کلاس‌های درسی و آموزشی و هم‌چنین مسابقات برنامه‌نویسی و انواع آزمون‌ها مناسب به حساب می‌آیند. اما به دلیل کنترل کمتر روی فرایند کامپایل و هم‌چنین برخی محدودیت دیگر برای پروژه های بزرگ یا ایجاد تعداد فایل زیاد چندان انتخاب دقیقی نیستند. از بهترین محیط‌های آنلاین برای کدنویسی به زبان C++ می‌توان به Replit یا OnlineGDB اشاره کرد.

- محیط‌های برنامه‌نویسی موبایلی

این محیط‌ها در حقیقت، اپلیکیشن‌هایی هستند که امکان نوشتن، کامپایل و اجرای کد را مستقیما روی گوشی‌های هوشمند فراهم می‌نمایند. با وجود اینکه این محیط‌ها جایگزین محیط‌های دسکتاپ برای کارهای حرفه ای و جدی نمی‌شوند، اما برای تمرین و یادگیری و مشاهده خروجی بخصوص در محیط‌هایی که امکان دسترسی به لپتاپ یا کامپیوتر شخصی وجود ندارد، گزینه قابل قبولی است. مانند Coding C++ و Cxxdroid.

انواع خطاهای برنامه‌نویسی

- خطاهای زمان کامپایل

این خطاها قبل از اجرای برنامه و در مرحله تبدیل کد به زبان ماشین توسط کامپایلر شناسایی می‌شوند. از مهمترین خطاهای موجود در این دسته می‌توان به خطای نحوی و خطاهای نوع اشاره کرد. خطای نحوی در زمان نقض قواعد دستوری زبان برنامه‌نویسی یا به عبارت دیگر گرامر زبان مربوط می‌شود. خطای نوع هم زمانی رخ می‌دهد که بین نوع داده‌ها در عملیات تطابق وجود ندارد. در هر دو نوع خطای مذکور، کامپایلر عمل کامپایل را متوقف کرده و سپس خطا را اعلام می‌نماید.

- خطاهای زمان اجرا

این خطاها پس از کامپایل موفق و در حین اجرای برنامه رخ می‌دهد. مانند خطای منطقی که معمولا به دلیل پیاده‌سازی اشتباه یک الگوریتم رخ می‌دهد و خروجی آن نادرست و برخلاف هدف و انتظار ما است درحالیکه که هیچ خطایی گزارش نمی‌شود. همچنین خطاهای بحرانی هم دسته دیگری از خطاهای زمان اجرا را تشکیل می‌دهند که در مواردی مانند انجام عملیاتی مانند تقسیم بر صفر، سرریز و مواردی از این دست رخ خواهد داد. پیامد چنین خطاهایی هم به طور معمول توقف ناگهانی برنامه و یا تولید خروجی‌های نادرست و غیرمنطقی است.

- خطاهای معنایی

به طور معمول، منظور از خطاهای معنایی، تولید خروجی اشتباه و غیرواقعی حاصل از طراحی اشتباه یک الگوریتم است. در این موارد کد به درستی نوشته شده و هیچ مشکل فنی ندارد اما به دلیل به کار بردن اشتباهی یک الگوریتم یا طراحی غلط آن، خروجی مطلوب را در اختیارمان قرار نخواهد داد. تشخیص دشوار این نوع خطا به همراه ریشه داشتن در مراحل قبل از کدنویسی، باعث شده که خطای معنایی جزء خطرناک‌ترین و بدترین خطاهای برنامه‌نویسی شناخته شود.

قالب کلی یک برنامه به زبان C++

```
#include <iostream>
using namespace std;
int main(){


        return 0;
}
```

دستورهای شرطی

```
if (شرط یا شرط ترکیبی){
        دستورات//
}
else if (شرط یا شرط ترکیبی){
        دستورات//
}
else{
        دستورات//
}
```

حلقه‌های تکرار

```
int i ; //تعریف شمارنده
while(شرط یا شرط ترکیبی){
        دستورات//
        به‌روزرسانی شمارنده//
}
```

```
for( شمارنده ; شرط پایان ; به‌روزرسانی){
        دستورات//
```

مثالی از تعریف یک آرایه

```
float grade[5] = [19.5, 20, 12 ,14.75, 10];
cout<<grade[0];
```

تابع در C++

```
نوع داده خروجی    نام تابع ( پارامترهای ورودی){
        //دستورات
}

int  main(){
نام تابع (آرگومان‌ها)   // فراخوانی تابع
}
```

دانشجویان عزیز توجه کنند، مطالب فوق صرفا برخی مباحث تکمیلی و مروری درس هستند و برای امتحان تمام مواردی که در کلاس (چه حضوری و چه ویدویی و مجازی) تدریس شده نیز لحاظ خواهد شد.

مطالب غیرامتحانی

تبدیل مبنای اعداد اعشاری

در خصوص اعداد اعشاری، بحث ارزش مکانی رقم مشابه مطالب بیان شده در کلاس درس است. تنها تفاوت این است که ارزش ارقام بعد از ممیز، هر چه به سمت راست حرکت کنیم کمتر می‌شود. به عبارت دقیق‌تر، ارزش اولین رقم بعد از ممیز برابر با 10^{-1} و در حالت کلی، ارزش رقم i ام برابر با 10^{-i} است.

با این ترتیب برای تبدیل اعداد اعشاری از مبنای 10 به مبنای 2، ابتدا قسمت صحیح عدد را به روش گفته شده، به مبنای 2 تبدیل می‌کنیم. سپس قسمت اعشاری آن را با روش ضرب‌های متوالی مشابه روش تقسیم متوالی ولی با ضرب عدد در 2 در هر مرحله به مبنای 2 تبدیل می‌کنیم. با هربار ضرب، قسمت صحیح عدد به عنوان رقم مربوطه در تبدیل عدد به مبنای 2 استفاده می‌شود و قسمت اعشاری عدد دوباره وارد روند ضرب‌های متوالی می‌شود. روند ضرب‌های متوالی وقتی متوقف می‌شود که عدد صفر شود. در روش تقسیم‌های متوالی حتما پس از تعداد متناهی مرحله به پایان روند می‌رسیم ولی در تبدیل قسمت اعشاری ممکن است هیچ موقع به پایان نرسیم.

مثال: عدد 13.25 را به مبنای دو تبدیل می‌کنیم.

با استفاده از روش تبدیل اعداد صحیح به مبنای دو (که در کلاس گفته شده) داریم: $13=(1101)_2$. برای تبدیل عدد 0.25 به مبنای 2، ابتدا آن را در 2 ضرب می‌کنیم. قسمت صحیح عدد حاصل برابر 0 و قسمت اعشاری آن برابر 0.5 است. دوباره عدد را در دو ضرب می‌کنیم. قسمت صحیح عدد برابر با 1 و قسمت اعشاری برابر با صفر است. این به معنی پایان روند است و لذا داریم:

$$(0.01)_2=0.25$$

و در نهایت خواهیم داشت:

$$(1101.01)_2=13.25$$

نحوه ذخیره‌سازی

برای نگهداری اعداد منفی باید به گونه‌ای مثبت یا منفی بودن عدد مشخص شود. برای این کار روش‌های مختلفی وجود دارد که در اینجا به تعدادی از آن‌ها اشاره می‌شود:

روش علامت-مقدار: در این روش، با ارزش‌ترین بیت از فضای درنظر گرفته شده برای نگهداری عدد صحیح به علامت اختصاص داده می‌شود. اگر مقدار این بیت برابر با صفر باشد به معنی مثبت بودن عدد و در صورت 1 بودن این بیت به معنی منفی بودن عدد است. با توجه به اینکه فضای ذخیره‌سازی عدد یک بیت کاهش می‌یابد، بازه محدودتری از اعداد قابل نگهداری در فضا است. به عنوان مثال اگر دو بایت برای نگهداری عدد صحیح استفاده شود، عملا 15 بیت برای نگهداری عدد در اختیار داریم و لذا اعداد بین 2^{15}- تا 2^{15}-1 یعنی اعداد 32768- تا 32767 قابل نگداری در این فضا است.

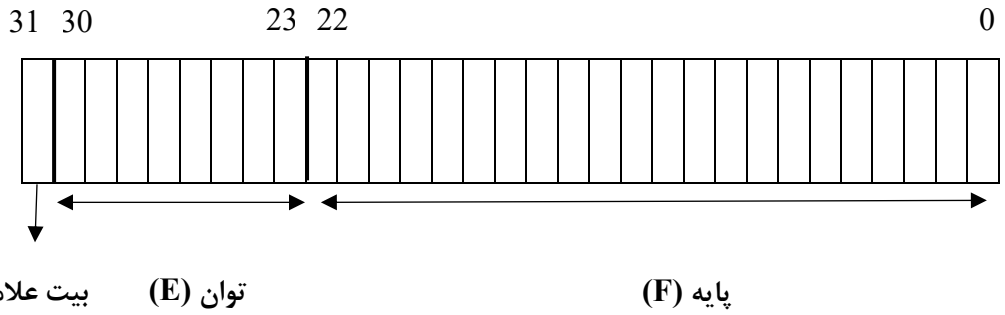
روش مکمل اول: در این روش، برای تبدیل عدد منفی به مبنای دو، ابتدا قدر مطلق آن عدد به مبنای دو تبدیل می‌شود. اگر عدد کوچکتر از فضای مورد نظر برای نگهداری عدد است، به تعداد لازم صفر در پشت عدد اضافه می‌شود. سپس تمام ارقام 1 به 0 و تمام ارقام 0 به 1 تبدیل می‌شود. به عنوان مثال عدد 97- در روش مکمل اول با استفاده از یک بایت برای ذخیره‌سازی به صورت 10011110 است.

روش مکمل دوم: در این روش، مشابه روش مکمل اول عمل می‌کنیم و در انتها به عدد حاصل عدد 1 را اضافه می‌کنیم. به عنوان مثال، عدد 97- در این نمایش و در 8 بیت به صورت 10011111 است.

اعداد اعشاری:

اعداد اعشاری، علاوه بر اینکه اعداد با اندازه‌های مختلف را دارند، ممکن است دارای تعداد ارقام اعشاری مختلف نیز باشند. در اینجا فقط به روش ممیز شناور اشاره می‌شود.

عدد ممیز شناور، اعداد را به صورت علمی یعنی $F * r^F$ در آورده و سپس E و F را نگهداری می‌کند که در آن r مبنای در نظرگرفته شده برای عدد است. برای ذخیره‌سازی اعداد اعشاری در کامپیوتر از r=2 استفاده می‌کنیم. برای این منظور ابتدا عدد را به نماد علمی در مبنای 2 تبدیل کرده و سپس فضای درنظر گرفته‌شده برای ذخیره‌سازی را به دو قسمت تقسیم کرده و در یک قسمت E و در قسمت دیگر F را ذخیره می‌کنیم. البته باید دقت کرد که نمایش یک عدد اعشاری منحصر به فرد نیست. مثلا عدد 55.66 در سیستم دهدهی را می‌توان به صورت مختلفی نوشت مثل $5.566*10^1$ یا $0.5566*10^2$. معمولا از یک شکل نرمال‌شده برای ذخیره‌سازی استفاده می‌شود. در شکل نرمال شده پایه بین 0 و 1 درنظر گرفته می‌شود. این عدد در نمایش دودویی در محل درنظر گرفته شده برای پایه و توان که یک عدد صحیح است در قسمت توان ذخیره می‌شود. همچنین یک بیت نیز برای علامت عدد اعشاری درنظر گرفته شده است.



با توجه به شکل فوق، مشخص است که برای اعداد اعشاری نیز مشابه اعداد صحیح، دارای بازه اعداد قابل نگهداری هستیم که فضای در نظرگرفته شده برای توان ، این بازه را تعیین میکند. علاوه بر این، در تعداد ارثام اعشاری قابل ذخیره‌سازی با دقت نیز محدودیت وجود دارد. نتیجه سریع این محدودیت این است که امکان نگهداری اعداد اعشاری که نمایش آنها دارای تعداد رقم اعشاری متناهی نیست به صورت دقیق در کامپیوتر وجود ندارد و برای ذخیرهسازی آنها بسته به فضای اختصاص داده شده به عدد، گردکردن عدد انجام شده و سپس ذخیره میشود. لذا انجام محاسابت دقیق در این خصوص ممکن نیست و این امر باید در کار با کامپیوتر مدنظر قرار گیرد.

حروف و نمادها

حروف و نمادها مشابه اعداد نیستند که بتوان آن را در مبنای دو نمایش داد و در کامپیوتر ذخیره کرد. برای حل این مشکل، از کدگذاری استفاده میکنیم، بعنی به هر حرف یا نماد، عددی نسبت میدهیم و عدد متناظر با هر حرف را به حای آ ذخیره میکنیم. با این ترتیب تمام اطلاعات متنی با استفاده از کد کردن به صورت تعتدادی عدد ذخیهر میشود و برای استفاده از آن یا خروجی آن، عمل کدگشایی آنها صورت میپذیرد.

کدگذاری اسکی (ASCII)

این روش جزء اولین روشهای کدگذاری استاندارد برای حروف است. در این روش ابتدا از 7 بیت برای نگهداری کد مربوط به حروف و نمادها استفاده میشد که بعدا جهت هماهنگی با کامپیوترهای 8 بیتی، به هشت بیت افزایش یافت. برای مثال در این کدگذاری، برای نگهداری حروف A تا Z از کدهای دهدهی 65 تا 90، برای a تا z از کدهای 97 تا 122 و برای ارقام 0 تا هم از کدهای 48 تا 57 استفاده شده است.

گسترشهای متعددی از این استاندارد مانند ANSI یا EBCDIC وجود دارد. با توجه به محدودیت تعداد کدها بدیهی است تعداد حروف محدودی را میتوان در این روش نگهداری کرد که عمدتا همان حروف زبان انگلیسی است.

یونیکد

قبل از یونیکد، هیچ نمایش دیگری از حروف، تمام زبان ها را پوشش نمیداد. هدف از یونیکد، ارائه یک روش کدگذاری استاندارد است که جهانی و کارا و غیرمبهم باشد. یونیکد از 16 بیست برای کدگذاری متغیرها استفاده میکند که میتواند 65536 حرف را کدگذاری کند. این فضا تا 21 بیت یز گسترش یافته است که تقریبا دو میلیون حرف جدید و قدیمی را پوشش میدهد. روشهایی مانند UTF-8 و UTF-16 برای کاراتر کردن این نمایش ارائه شده است.