به نام خداوند بخشنده‌ی مهربان

شماره‌ی ۲ ۲۲ مهر ۱۴۰۰

کد نامه

ویژه‌ی دانش‌جویان مبانی برنامه‌سازی نیم‌سال اول ۱۴۰۱-۱۴۰۰ دانشکده‌ی مهندسی کامپیوتر دانشگاه صنعتی شریف

مثبت یا منفی، مسئله این است!

آشنایی با ذخیره و محاسبات اعداد در کامپیوتر

با نگاهی به تاریخچه‌ی کامپیوترها و مراجعه به تعاریف لغت‌نامه‌ای، درمی‌یابیم که مهم‌ترین وظیفه‌ی آن‌ها، انجام سریع، دقیق و «هوشمندانه»ی محاسبات است. برای این کار، پیش از هر چیز نیاز به ذخیره‌سازی اعداد در حافظه داریم. در این مقاله به صورت اجمالی بررسی می‌کنیم که کامپیوتر چگونه اعداد را ذخیره کرده و چگونه روی آن‌ها محاسبات انجام می‌دهد. با کدنامه همراه باشید.

تقویم درس در هفته‌ی پیش‌رو

۲۵ مهر

مقدمات برنامه‌سازی در زبان C

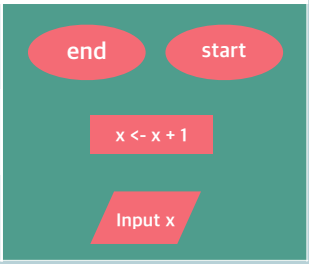
۲۷ مهر

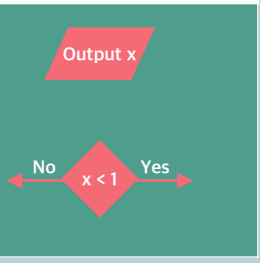
دستورات شرط و حلقه

۲۸ - ۲۹ مهر

مهلت تمرین ۱ و انتشار تمرین ۲ - I/O و محاسبات

نمادهای مرسوم در رسم فلوچارت

نماد	شرح عملکرد
	شروع و پایان الگوریتم (فلوچارت) انجام محاسبات عددی و اعمال اصلی گرفتن ورودی از کاربر و ریختن آن در متغیر

نماد	شرح عملکرد
	نمایش مقدار متغیر مشخص شده در خروجی به کاربر دستور شرط؛ اگر عبارت داخل لوزی، «درست» باشد، فلوچارت به شاخه Yes و در غیر این صورت، به شاخه‌ی No می‌رود

مثبت یا منفی، مسئله این است!

سینا ایمانی

امروزه از کامپیوتر برای نگهداری انواع و اقسام اطّلاعات استفاده می‌شود. از موسیقی و فیلم تا متن و عدد – محتواهایی گوناگون و متنوّع. امّا آن‌چه در حافظه‌ی کامپیوتر قابل ذخیره‌سازی است، چیزی نیست جز دنباله‌ای از bitها. بنابراین ناگزیریم که هر آن‌چه می‌خواهیم ذخیره شود را ابتدا به تعدادی بیت تبدیل کرده و سپس آن را در حافظه ثبت کنیم؛ به این ترتیب، نیاز به یک «تناظر»، یک «پروتکل» یا یک «فرمت» برای ذخیره‌سازی داریم تا هر چیزی که قرار است ثبت شود را طبق قوانین آن، به بیت‌ها تبدیل کنیم و نیز برعکس، از روی بیت‌های ثبت شده بفهمیم که چه چیزی قرار بوده ذخیره شود. برای این کار می‌توان از تناظرهای مختلفی استفاده کرد که هرکدام مزایا و معایب ویژه‌ی خود را دارند؛ مثلاً برای ذخیره‌ی صوت، فرمت‌های wav و mp3 از فرمت‌های رایج هستند.

هنگامی که با یک Media Player (مثلاً VLC) موسیقی دل‌خواهتان را پخش می‌کنید، آن نرم‌افزار بیت‌های حافظه را می‌خواند و با دانستن فرمتی که موسیقی در آن ذخیره شده، می‌تواند بیت‌ها را دوباره به موسیقی تبدیل کند. در برنامه‌هایی که می‌نویسید نیز برای داده‌هایی که ذخیره می‌کنید، اتفاق مشابهی رخ می‌دهد؛ با قراردادن نوع (type) برای داده‌ی خود، در واقع به برنامه می‌گویید چه مقدار حافظه به آن اختصاص دهد و bitهایی که در آن ریخته شده را چگونه «تفسیر» کند، یعنی آن را معادل چه چیزی بداند.

سیستم های محاسبه

اجازه دهید برای ادامه‌ی بحث، تمرکز خود را بر روی «پروتکل‌های ذخیره‌سازی اعداد **صحیح**» (یا به اختصار، سیستم‌های محاسبه) قرار دهیم. فرض کنید برای ذخیره‌سازی یک عدد صحیح، تعدادی بیت (از این به بعد، فرض کنید مثلاً ۵ بیت) از حافظه را در اختیارمان قرار داده‌اند و اکنون می‌خواهیم تناظری بین 2^5 دنباله‌ی مختلف از بیت‌ها و یک بازه از اعداد صحیح برقرار کنیم. برای این منظور، سیستم‌های محاسباتی متفاوتی ارائه شده که در ادامه چند مورد مشهورتر را بررسی می‌کنیم.

پیش از اینکه جلوتر برویم، اجازه بدهید که دو اصطلاح من‌درآوردی – امّا مفید – را تعریف کنیم، به نام‌های «عدد ظاهری» و «عدد واقعی». عدد ظاهری همان عددی است که اگر سیستم محاسباتی ما سیستم باینری می‌بود، بیت‌های ذخیره شده در حافظه به معنای آن عدد تلقّی می‌شد. مثلاً ممکن است در یک تناظر فرضی، منظور شما از ۰۱۱۰۱ این باشد که «چه بعدازظهر بارانی دل‌انگیزی!»، امّا عدد ظاهری ذخیره شده چیزی نیست جز ۱۳. مشابهاً «عدد واقعی» هم آن عددی است که منظور از بیت‌های ذخیره شده، واقعاً آن عدد است. برگردیم به سیستم‌های عددی.

سیستم دودویی (باینری)

در این سیستم، به سادگی، عدد واقعی را همان عدد ظاهری در نظر می‌گیریم. در این روش با داشتن ۵ بیت، اعداد ۰ تا $2^5 - 1$ را می‌توان نمایش داد. ایرادی که این سیستم دارد، این است که اعداد صحیح با علامت منفی در آن قابل ذخیره‌سازی نیستند؛ در حالی که در بسیاری از موارد، نیاز به نگهداری اعداد صحیح منفی داریم. برای این منظور، باید به دنبال سیستم دیگری باشیم. انتظار ما از سیستم جدید این است که:

الف) حدود نیمی از حالات ممکن بیت‌ها (مثلاً آن‌هایی که با صفر شروع می‌شوند) را به اعداد مثبت و نیم دیگر را به اعداد منفی اختصاص دهد. مثلاً اگر ۵ بیت در اختیار داریم، حدوداً اعداد از ۱۶– تا ۱۶ را بتواند ذخیره کند.

ب) بتوان به آسانی و با قواعدی مشابه سیستم باینری، روی اعداد ذخیره‌شده در آن سیستم، عملیات حسابی (مثل جمع کردن و به‌دست‌آوردن قرینه) را انجام داد.

سیستم مقدار–علامت

نخستین سیستمی که معمولاً به ذهن می‌رسد، سیستم «مقدار–علامت» است. در این سیستم، یک بیت قراردادی (معمولاً بیت سمت چپ) را برای تعیین علامت عدد استفاده کرده (صفر یعنی مثبت، یک یعنی منفی) و باقی بیت‌ها را برای تعیین قدر مطلق آن استفاده می‌کنیم. مثلاً با ۵ بیت، برای نشان دادن ۹+ از ۰۱۰۰۱ و برای ۱۱– از ۱۱۰۱۱ استفاده می‌شود. بنابراین، اگر عدد واقعی‌مان مثبت باشد، عدد ظاهری برابر عدد واقعی خواهد بود.

قرینه کردن یک عدد در این سیستم آسان است؛ کافی است بیت علامت را not کنیم. هم‌چنین، به علّت شباهت به سیستم دودویی معمولی، متوجّه شدن آن برای انسان راحت است؛ ولی متأسّفانه انجام عملیات جمع در آن به طور قابل توجّهی سخت‌تر و پیچیده‌تر از سیستم باینری معمولی است. در واقع، بسته به هم‌علامت بودن یا نبودن جمع‌وندها، جمع کردن آن‌ها قواعد متفاوتی پیدا می‌کند. مشکل دیگری که این سیستم دارد، این است که دو نمایش متفاوت برای عدد ۰ دارد! بنابراین با فرض در اختیارداشتن ۵ بیت، به جای ۳۲ عدد متفاوت، ۳۱ عدد را می‌توان ذخیره کرد. در نتیجه، لازم است دنبال سیستم بهتری برای حل مشکل بگردیم.

سیستم مکمّل ۱

در سیستم «مکمّل یک» نیز مانند سیستم مقدار–علامت، در دنباله‌هایی که بیت سمت چپ آن‌ها ۰ است، عدد واقعی را همان عدد ظاهری در نظر می‌گیریم؛ امّا دنباله‌هایی که با ۱ شروع می‌شوند (مثلاً ۱۱۰۰۱) را چگونه می‌توانیم به اعداد منفی اختصاص دهیم؟

یک ایده‌ی جالب، استفاده از این واقعیت است که «اگر هر بیتِ دنباله‌ای که بیت سمت چپ آن ۱ است را نقیض کنیم، به دنباله‌ای می‌رسیم که برای نمایش یک عدد مثبت استفاده می شود». می‌توانیم دنباله‌های شروع شونده با ۱ را ابتدا بیت به بیت نقیض کنیم تا ببینیم به بسط باینری چه عددی می رسیم. اگر به عدد x رسیدیم، دنباله ی اصلی را معادل x- در نظر می گیریم. مثلاً ۱۰۱۰۱ برابر ۰۱۰۱۰- یا همان ۱۰- و ۱۱۱۱۰ برابر ۱- خواهد شد. مشخّص است که برای قرینه کردن یک عدد در این سیستم، کافی است که تمام بیت‌های آن را نقیض کنیم.

اگرچه انجام عملیات های حسابی در این سیستم آسان‌تر از سیستم قبلی است، امّا هنوز هم دو نمایش متفاوت برای ۰ وجود دارد؛ بنابراین، نیاز به یک مرحله ارتقای بیشتر داریم.

سیستم مکمّل ۲

آخرین و کامل‌ترین سیستم که در این درس با آن آشنا می‌شویم، سیستم مکمّل دو است که از خواص هم‌نهشتی استفاده می‌کند. به این ترتیب، این سیستم می‌تواند هر کدام از دنباله‌ها را با یک عدد متمایز در بازه‌ی ۱۶- تا ۱۵ متناظر کند . در این سیستم نیز دنباله‌های شروع شونده با صفر را بسط باینری یک عدد نامنفی در نظر می‌گیریم (عدد واقعی = عدد ظاهری). اگر هم دنباله‌ای با یک شروع شود، عدد واقعی برابر منفی شده‌ی حاصل کم کردن عدد ظاهری از ۳۲ (به فرض داشتن ۵ بیت) خواهد شد.

اگر دقت کنیم، در می‌یابیم که در هر صورت برای ذخیره‌ی یک عدد در این سیستم، باقیمانده‌ی تقسیم آن عدد بر ۳۲ را در نظر گرفته و بسط باینری آن را ذخیره می‌کنیم. این کار باعث می‌شود تا همواره عدد واقعی و عدد ظاهری، به پیمانه ی ۳۲ هم‌نهشت باشند. به این ترتیب، می‌توان از عملیات معمولی جمع و ضرب باینری برای جمع و ضرب اعداد در این سیستم استفاده کرد. به نظر می‌رسد که بالاخره پاسخ را یافته ایم!

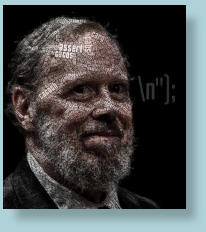
آن‌چه کامپیوتر واقعاً انجام می‌دهد

پس از نوشته شدن یک برنامه‌ی رایانه‌ای، هنگام «کامپایل» و اجرا، برنامه به دستورات «ساده‌تر»ی تبدیل می‌شود تا برای کامپیوتر قابل اجرا باشد. این روندِ ساده‌تر شدن تا جایی ادامه پیدا می‌کند که به دستوراتِ قابل اجرا برای سخت‌افزار کامپیوتر برسیم؛ آنچه سخت‌افزار کامپیوتر می‌تواند انجام دهد، چیزی نیست جز عملیات ساده‌ی بیتی- یعنی، not، and، xor، or و نظایر آن‌ها. هر عملیات حسابی، باید نهایتاً به این دستورات تبدیل شود. می‌توان با بررسی حالات مختلف، برای عملیات حسابی مورد نظر «مدار»ی طراحی کرد تا سخت‌افزار بتواند آن عملیات را انجام دهد. مثلاً دستور جمع را در نظر بگیرید. هنگامی که بخواهیم دو عدد را در مبنای دو جمع کنیم، از بیت سمت راست آن‌ها شروع می کنیم. فرض کنید اسم آن دو x و y باشد. پس از جمع کردن آن دو، بیت سمت راست حاصل جمع (sum) به دست می‌آید. هم‌چنین، ممکن است نیاز باشد که یک رقم «انتقالی» (یا carry) برای محاسبه‌ی جمع در مرحله‌ی بعد ایجاد شود:

x	y	carry	sum
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

با نگاهی به این جدول، مشخّص می‌شود که s برابر x XOR y بوده و c برابر and آن دو است. حال اگر به کمک قطعاتی به نام «gate های منطقی»، بتوانیم مداری الکتریکی بسازیم که x و y را ورودی بگیرد و s و c را خروجی دهد، می‌توانیم به صورت «سخت‌افزاری»، حاصل جمع دو بیت (دو عدد) را حساب کنیم. در آینده در درس «مدارهای منطقی»، مثال‌های جالب‌تر و پیچیده‌تری از همین موضوع را مشاهده خواهید کرد!

عکس کانال «مبانی پلاس»، چه شخصی است؟



دانشمندی که عکس او به عنوان عکس کانال انتخاب شده، **دنیس مک‌آلستر ریچی**، دانشمند کامپیوتر و خالق زبان برنامه‌نویسی سی و هم‌چنین سیستم‌عامل Unix است که به عنوان پایه و اساس اکثر سیستم‌های عامل کنونی، از جمله Linux، macOS، Android و iOS، استفاده می‌شود.

دنیس ریچی به همراه کن تامپسون، زبان C را که در این نیم‌سال تحصیلی با آن سر و کار داریم، اختراع کرده و جوایز متعددی – از جمله جایزه‌های تورینگ و همینگ – را مشترکا دریافت کرده‌اند. امروزه، زبان C، به صورت گسترده‌ای در بسیاری از سیستم‌های عامل و نرم‌افزارهای کاربردی مختلف در جهان، استفاده می‌شود. پیشنهاد می‌شود برای آشنایی بیش‌تر با این دانشمند، به این‌جا مراجعه کنید.

حرفه‌ای فلوچارت بکشیم

سید پارسا نشایی، محمدمهدی برقی

چرا فلوچارت؟

ممکن است برای شما این سوال به وجود آمده باشد که «می‌توان هر الگوریتمی را به زبان فارسی یا انگلیسی و یا حتی به شکل کدهای برنامه‌نویسی توصیف کرد، پس چه نیازی به رسم فلوچارت وجود دارد؟»

احتمالا بسیاری از شما، در حل مسائل مربوط به حوزه‌هایی از ریاضی هم‌چون نظریه‌ی مجموعه‌ها، جهت آمادگی برای کنکور سراسری، از ابزارهای ترسیمی مثل نمودار ون یا اشکالی شبیه به آن استفاده کرده اید. وقتی اعضای چند مجموعه را به صورت تعریف ریاضی و یا حتی کلامی بیان می‌کنید، یافتن اشتراک و اجتماع آن‌ها در یک نگاه آسان نیست، اما اگر اعضا را در نمودارهای ون رسم کنید، با یک نگاه می‌توانید اعمال روی مجموعه‌ها را در ذهن خود انجام داده و در کسری از ثانیه، گزینه‌ی درست را روی پاسخ‌برگ علامت بزنید. نه تنها در نظریه‌ی مجموعه‌ها، بلکه در بسیاری از علوم و مهارت‌های گوناگون – از جمله برنامه‌نویسی – رسم شکل می‌تواند به درک بهتر مسئله کمک کرده و هم‌چنین از به کار بردن فعل‌های تکراری جلوگیری کند.

چگونه فلوچارت بکشیم؟

در تمرینات و امتحانات درس مبانی برنامه‌سازی، مکررا به رسم فلوچارت نیازمند خواهید شد، بنابراین لازم است با اصول اولیه‌ی رسم یک فلوچارت که قادر به دریافت نمره باشد، آشنا شوید. برای رسم فلوچارت، لازم است قدم‌های زیر را طی کنید:

۱. در ذهن خود – و یا اگر زمان کافی دارید، بر روی کاغذ – الگوریتم مسئله را بنویسید.

۲. یک نماد start در ابتدای فلوچارت قرار دهید تا نقطه‌ی شروع فلوچارت را مشخص کند.

۳. به ترتیب از خط اول الگوریتم پیش‌روی کنید و خطوط را با نمادهایی که در کلاس درس آموخته‌اید، جای‌گزین کنید. هر جا که پرش یا رفتن به دستور دیگری موجود است، به ازای آن، یک فلش در فلوچارت قرار دهید.

۴. در پایان، نماد end را قرار دهید.

نکات مهم و اشتباهات متداول

۱. ترتیب منطقی اجرا را در الگوریتم و فلوچارت، کاملا رعایت کنید.

۲. سعی کنید فلوچارت خود را به گونه‌ای رسم کنید که کم‌ترین پیچیدگی و تلاقی خطوط را داشته باشد تا تصحیح آن برای دستیاران آموزشی مسئول تمرین‌ها، آسان‌تر شود.

۳. فلوچارت رسم شده، **حتما باید** شروع و پایان مشخصی داشته باشد. عدم رعایت این موضوع، ممکن است منجر به **کسر نمره** شود.

۴. می‌توانید به جای قلم و کاغذ، از وب‌سایت‌ها و نرم‌افزارهایی هم‌چون draw.io، edraw و یا creately برای رسم فلوچارت تمرین‌ها استفاده نمایید. از مزایای استفاده از این نرم‌افزارها، اصلاح کردن فلوچارت به آسانی، بدون نیاز به استفاده مکرر از پاک‌کن و یا رسم دوباره‌ی فلوچارت از اول است.

یک فلوچارت حرفه‌ای

فلوچارت، نقشه‌ای است که گام به گام، شما را به حل مسئله، نزدیک و نزدیک‌تر می‌کند و راه را قبل از دست به کار شدن و صرف هزینه‌ی زیاد، به شما نشان می‌دهد؛ به همین دلیل لازم است «حرفه‌ای فلوچارت بکشیم» و اصول مهمی را در کشیدن فلوچارت‌هایمان، رعایت کنیم.

نظم را برقرار کنید

نه لازم است از انواع و اقسام نمادهای ناشناخته با تعداد زیاد استفاده کنید و نه از نمادهای محدود با تعداد کم. لازم است هر بیننده، در یک نگاه به هدف الگوریتم پشت آن فلوچارت، پی ببرد.

در هر کاری، نظم خوب است، اما در رسم فلوچارت، نظم واجب است! سعی کنید تمام متغیرها را درون یک نماد مقداردهی نکنید تا تراکم فلوچارت کم‌تر شود. هم‌چنین باید به جهت‌های اصلی ترسیم، توجه داشته باشید. جهت‌های اصلی فلش‌ها، «به سمت پایین» و «به سمت بالا» هستند. رعایت این نکته، از پراکندگی فلوچارت، جلوگیری می‌کند. هم‌چنین، سعی کنید اندازه‌ی نمادها متناسب با هم باشند.

فلوچارت یا بوم نقاشی؛ مسئله این است!

فلوچارت، مکانی برای نمایش هنر نقاشی و رنگ‌آمیزی شما نیست. رنگارنگ بودن بیش از حد فلوچارت، از حدی به بعد، آزاردهنده می‌شود.

علاوه بر رعایت نکات فوق، سعی کنید خطوط فلوچارت، کم‌ترین تلاقی ممکن را داشته باشند. هم‌چنین، استفاده از دایره‌های شماره‌دار و شماره‌گذاری را برای چند قسمتی کردن فلوچارت خود، حتما در نظر داشته باشید تا فلوچارت شما تمیزتر رسم شود و در یک نگاه، هدف از رسم فلوچارت، راحت‌تر قابل تشخیص بوده و تصحیح آن نیز برای دستیاران آموزشی، راحت‌تر باشد.

فلوچارت زندگی‌تان، عالی و بی‌نقص باد!