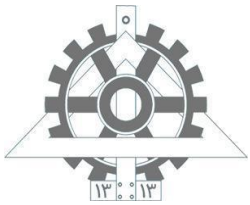


UT Trading Language (UT^2L)

Documentation

Nazanin Torabi Goudarzi
Mohammad Ali Zamani

فهرست مطالب

۱ - مقدمه ۳

۱ - ۱ -کنترل اجرای برنامه ۸

۲ - ساختار کلی ۱۰

۲ - ۱ - قواعد کلی نحو ۱۰

۲ - ۲ - کامنت‌ها ۱۰

۲ - ۳ - قواعد نامگذاری کلاس ها، متدها و متغیرها ۱۱

۳ - نشانه‌گذاری زمانبند (Schedule) ۱۲

۴ - متغیرها ۱۲

۴ - ۱ - متغیرهای از پیش تعریف شده ۱۲

۴ - ۲ - متغیرهای استاتیک ۱۳

۴ - ۳ - متغیرهای External ۱۳

۵ - انواع داده ۱۴

۵ - ۱ - انواع داده پایه ۱۴

۵ - ۲ - لیست ۱۴

۵ - ۳ - معامله ۱۴

۵ - ۴ - سفارش ۱۵

۵ - ۵ - خطا ۱۵

۵ - ۶ - ستون ۱۶

۶ - عملگرها ۱۶

۶ - ۱ - عملگرهای حسابی ۱۷

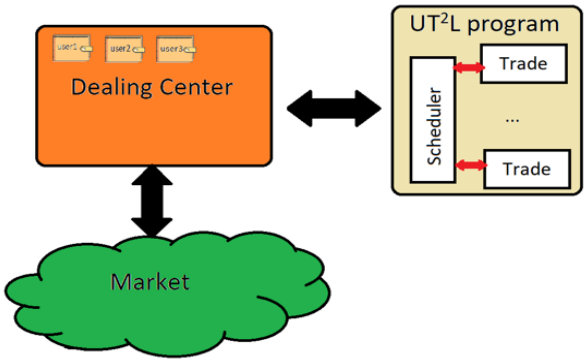
۶ - ۲ - عملگرهای مقایسه‌ای ۱۷

۶ - ۳ - عملگرهای منطقی ۱۸

۶ - ۴ - عملگرهای تخصیص ۲۰
۶ -۵ - اولویت عملگرها ۲۱
۷ - ساختار تصمیم‌گیری ۲۲
۸ - ساختار خطا ۲۲
۹ - ساختار تکرار ۲۳
۹ - ۱ - کلیدواژه break ۲۳
۹ - ۲ - کلیدواژه continue ۲۳
۱۰ - قوانین گستره ۲۴
۱۱ - توابع پیش فرض ۲۴

۱ – مقدمه

زبان UT^2L یک زبان جامع برای انجام معاملات بازار[۱] ارزهای دیجیتال[۲] و فارکس[۳] است که در ادامه UTL و UT^2L به جای یکدیگر مورد استفاده قرار می‌گیرند. این زبان به منظور انجام معاملات الگوریتمی[۴] طراحی و توسعه یافته است و انجام معاملات را بر اساس استراتژی‌های معامله‌گر آسان می‌نماید. برنامه‌هایی که در این زبان نوشته می‌شوند، برای اجرا به یک واسط معاملاتی[۵] متصل می‌گردند. شکل ۱، نحوه ارتباط برنامه و واسط معاملاتی را نشان می‌دهد. هر فرد برای انجام معامله[۶] لازم است که یک حساب در واسط معاملاتی داشته باشد که ثبت نام در واسط معاملاتی از طریق سازمان‌ها و یا برنامه‌های مشخصی انجام می‌گیرد. هم‌چنین، هر فرد در حساب خود کیف پولی[۷] دارد که ارزهای خریداری شده را در آن نگهداری می‌کند. هر برنامه در هنگام اتصال به واسط معاملاتی، درخواست خود را با نام کاربری و رمز عبور حساب کاربری مربوط به خود که از قبل ثبت شده است، ارسال می‌کند (دستور Connect در بخش ۱۰ را مشاهده کنید). در هنگام انجام معاملات، ارز مورد نیاز، از کیف پول حساب کاربری کم می‌شود و ارز خریداری شده به کیف پول افزوده می‌گردد. در واقع، کیف پول برای ارزهای مختلف، مقادیر نظیر را نگهداری می‌نماید.



شکل ۱: شمای ارتباط برنامه UT^2L و واسط معاملاتی

[1] Market
[2] Cryptocurrency
[3] FOREX
[4] Algorithmic trading
[5] Broker
[6] Trade
[7] Wallet

واسط معاملاتی به صورت دوره‌ای اطلاعات شاخص‌های بازار معاملاتی را برای برنامه ارسال می‌کند که زمان ارسال اطلاعات با رویداد[8] تیک[9] بیان می‌گردند. این اطلاعات در برنامه، در متغیرهای مشخصی ذخیره می‌شوند (در واقع مقدار متغیرها به‌روزرسانی می‌گردند) و برنامه می‌تواند به این متغیرها دسترسی داشته باشد.

هر بازار، معادل Market در شکل ۱، متشکل از تعدادی زیربازار[10] است که با نماد ارزهای تبادل شده مانند BTC/TMN (بیت‌کوین/تومن) نشان داده می‌شوند. هر برنامه، در زمان اتصال به واسط معاملاتی مشخص می‌کند که برای معامله مایل است اطلاعات کدام زیربازار را مشاهده نماید. بنابراین، در هر تیک، اطلاعات مربوط به آن زیربازار توسط واسط معاملاتی به برنامه ارسال می‌گردد. هر زیربازار را می‌توان به صورت نموداری مشابه شکل ۲ نشان داد. در این نمودار، در هر زمان، مقدار کمترین قیمت، بیشترین قیمت و حجم معامله انجام شده در قالب مستطیل مشخص می‌گردد که اصلاحاً به آن Candle گفته می‌شود. به عنوان مثال، در تاریخ ۸ نوامبر و در ساعت ۹:۰۲:۲۵ (زمان تهیه این مستند)، مقدار بیشترین قیمت معاملهٔ هر بیت‌کوین برابر مقدار ۱۸۲۴۹۹۹۸۰۱ تومان بوده است. به عبارت بهتر این ستون‌ها گویای اطلاعات لحظه‌ای معامله هستند. در هر زیر بازار هر معامله مدت زمانی دارد و این گونه نیست که در لحظه انجام شود؛ بلکه، یک زمان باز شدن و یک زمان بسته شدن دارد. هر ستون یا هر Candle در این نمودار، حاوی اطلاعات مجموعه معاملاتی می‌باشد که بین دو تیک انجام شده است. همانطور که در شکل ۲ بزرگنمایی شده است، قسمت پایین مستطیل نشان‌دهندهٔ کمترین قیمت یک سفارش می‌باشد که بین دو تیک باز شده است (۱) و قسمت بالای مستطیل بالاترین قیمت یک سفارش را نشان می‌دهد که در فاصله بین دو تیک بسته شده است (۲). هم‌چنین، خطوط بالا و پایین مستطیل بیانگر بیشترین و کمترین قیمت سفارشات انجام شده بین دو تیک هستند (خطوط ۳ و ۴). دقت شود که رنگ‌بندی‌های این نمودار بر اساس قیمت باز شدن و بسته شدن معامله هستند؛ بدین ترتیب که اگر قیمت باز شدن از قیمت بسته شدن بیشتر باشد، ستون آن قرمز یا سیاه رنگ خواهد بود و اگر قیمت باز شدن کم‌تر از قیمت بسته شدن باشد، ستون آن سبز یا سفید خواهد بود.

[8] Event
[9] Tick
[10] Sub-market

شکل ۲: زیربازار مربوط به معامله BTC/TMN از سایت والکس[11]

هر برنامهٔ UT²L، شامل تعدادی معامله و توصیفی برای زمانبندی معاملات است. توصیفات مربوط به زمانبندی مشخص می‌کنند که زمانبند[12] برنامه بایستی معاملات برنامه را با چه ترتیبی (یا حتی به صورت موازی) اجرا نماید. در شکل ۳، یک نمونه کد برنامه نشان داده شده است.

در هر رویداد تیک، ابتدا متغیرهای مربوطه به روز رسانی می‌شوند. سپس، کنترل به زمانبند منتقل می‌گردد تا زمانبند بتواند با توجه به نحوهٔ اجرای معاملات، کنترل را به برنامه‌ها منتقل نماید. برای اطلاعات بیشتر بخش ۱-۱ را مطالعه فرمایید.

[11] https://wallex.ir/app/trade/BTCTMN
[12] Schedule

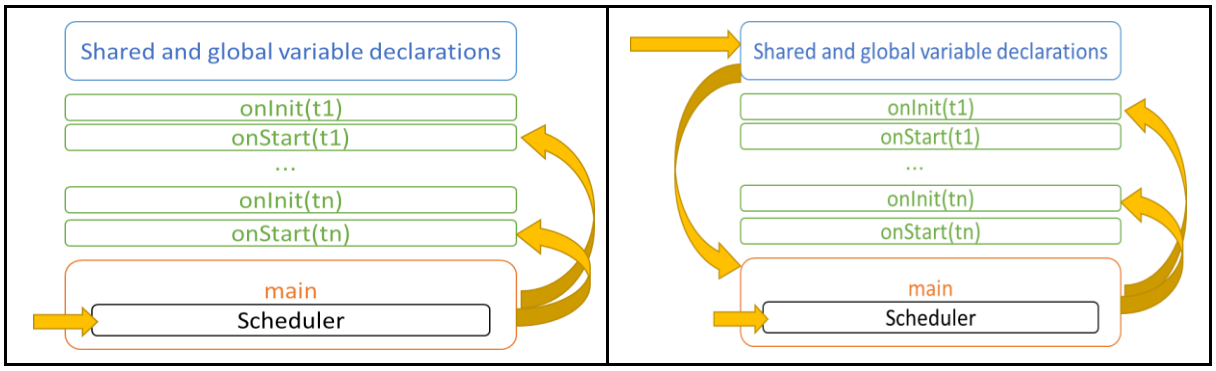
```
static int balance;
shared int tick_counts = 0;

void OnInit(Trade t3)
{
    // Just set some variables
    Order o3 = Order(SELL, 100, 100, 10);
}
void OnStart (Trade t1) throw Exception
{
    /*Update predefined variables in each tick, and based on policies
    that you have set the Orders are generated. */
    if (tick_counts > 5)
    {
        RefreshRate();
        tick_counts = 0;
    }

    float low = t1.Bid;
    double high = t1.Ask;
    float[2] predict;
    Candle [100] samples = GetCandle(100);
    float[100] maxSamples;
    for(int i=0; i==100 ;i++)
    {
        maxSamples[100-i] = samples[i].High;
    }
    //linear regression algorithm
    float Stoploss = 100;
    float TakeProfit = 250;
    float amount = 20;
    Order o1 = Order(BUY, Stoploss, TakeProfit, amount);
    if (predict[0] * 101 + predict[1] > 50000)
        o1.Open();

    float profit_ratio = TakeProfit/amount;
    while(profit_ratio > 5)
```

```
38      {
39          if (profit_ratio < 7)
40          {
41              o1.Close();
42              break;
43          }
44          else{
45              continue;}
46      }
47  }
48  void OnStart(Trade t2)
49  {
50      tick_counts ++;
51  }
52  void OnStart(Trade t3) throw Exception; //throws
53  {
54      GetCandle(100); // will not throw an exception if 100<tick_counts
55
56  }
57
58  void Main(){
59      try{
60          Connect("username", "password");
61          Trade t1 = Observe("USDETH");
62          Trade t2 = Observe("");
63          Trade t3 = Observe("IRRETH");
64      catch Exception e {
65          if (e.Type == 1)
66              print("Login Failure!");
67      }
68      @schedule (t1 preorder t3) parallel t2;
69      //@schedule (t1 parallel t2) preorder (t3 parallel t4);
70  }
```

شکل ۳: نمونه برنامه در UT^2L

۱ - ۱ - کنترل اجرای برنامه

در زمان شروع اجرای برنامه، ابتدا متغیرهای مشترک[۱۳] و سراسری[۱۴] تعریف می‌شوند و مقداردهی اولیهٔ آن‌ها صورت می‌گیرد. سپس تابع Main() برنامه اجرا می‌شود و در آن معاملات کاربر تعریف می‌گردند. منظور از معامله، شروع به رصد اطلاعات شاخص‌های یک زیربازار مشخص به منظور انجام معامله خرید و فروش است. پس از آن، اجرا، به بخش توصیف زمانبندی منتقل می‌گردد که براساس ترتیب تعریف شده در آن، تابع OnInit() معاملات اجرا می‌شوند (شکل ۴-الف). در هر تیک از سمت واسط معاملاتی، کنترل، مجدداً به بخش زمانبندی در تابع Main() منتقل می‌گردد تا براساس ترتیب تعریف شده در آن، تابع OnStart() معاملات اجرا شوند (شکل ۴-ب).



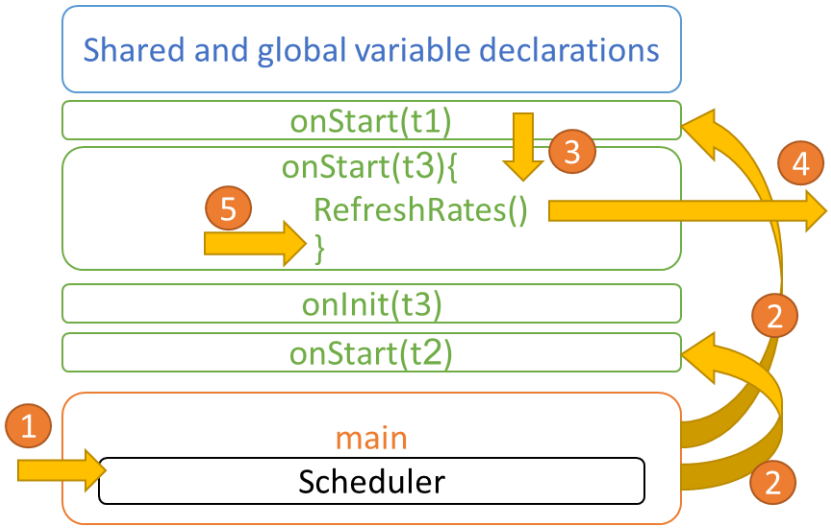
شکل ۴: روند کنترل اجرای ابتدایی برنامه (الف) روند کنترل اجرای برنامه در طول زمان (ب)

در زمان اجرا، واسط معاملاتی با ارسال رویداد تیک، اطلاعات زیربازارهای خواسته شده را به برنامهٔ ما ارسال می‌کند. در صورتی که اجرای تمام معاملات ترتیبی و موازی در تیک قبلی تمام شده باشد، برنامه، تیک بعدی را دریافت خواهد کرد و درغیراینصورت، تیک بعدی را از دست خواهد داد. برای مثال، در بدنهٔ OnStart() معامله t3 در شکل ۳، یک حلقهٔ بی پایان وجود دارد. پس از اجرای برنامه، هنگامی اولین تیک را دریافت می‌نماید که اجرای تابع OnInit() تمام معاملات پایان یافته باشد. پس از دریافت اولین تیک، دو معاملهٔ t1 و t2 به صورت موازی اجرا خواهند شد. پس از اجرای t1، کنترل برنامه به t3 می‌رود که اجرای آن پایان نمی‌پذیرد. پس برنامهٔ ما، تیک‌های بعدی را از دست خواهد داد و در واقع OnStart() معاملات t1 و t2 یکبار اجرا می‌شوند. اگر حلقهٔ while را در بدنهٔ

[13] Shared
[14] Global

t3 کامنت کنیم، اجرای t3 زمان زیادی نخواهد گرفت. توجه کنید که حلقه‌های طولانی یا فراخوانی‌های بازگشتی باعث زمان بر شدن اجرای بدنهٔ ()OnStart یک معامله خواهند شد و اطلاعات متغیرها پس از مدتی کهنه خواهند شد. درصورتی که تصمیمات معاملهٔ ما به مقادیر به روز رسانی شده بستگی داشته باشد، با دستور ()RefreshRates، اجرای معاملهٔ در حال اجرا متوقف می‌شود و مقادیر جدید از واسط معاملاتی دریافت می‌گردند. توجه کنید معاملاتی که به صورت موازی اجرا می‌شوند درصورت دسترسی به مقدار متغیرها، مقادیر به روز شده را دریافت خواهند کرد. هم‌چنین، توجه نمایید که برنامه قابلیت تغییر متغیرهای مربوط به شاخص زیربازارها را ندارند و بنابراین مشکل همروندی نخواهیم داشت.

برای مثال، ترتیب کنترل اجرای برنامه در شکل ۳ در زیر نشان داده شده است. در هنگام آمدن تیک، براساس ترتیب در توصیف زمانبندی، دو معاملهٔ t1 و t2 به صورت موازی اجرا می‌شوند که پس از اتمام اجرای t1، معاملهٔ t3 اجرا خواهد شد (حلقه بی پایان در خط ۵۴ را کامنت کنید). در صورت اجرای t3، اجرای t3 و t2 (اگر هنوز تمام نشده باشد) متوقف می‌گردد تا مقادیر جدید از واسط معاملاتی دریافت گردند. پس از دریافت مقادیر، اجرای t3 و t2 از ادامه، شروع می‌شود.



شکل ۵: جریان کنترلی برنامهٔ شکل ۳

توجه داشته باشید که برنامه هنگامی تمام می‌شود که (۱) اجرای تمامی معاملات تمام شده باشد و یا (۲) اتصال برنامه با واسط معاملاتی قطع شده باشد و برنامه دیگر قادر به دریافت تیک‌ها نباشد.در حالت دوم، پیام خطایی مبنی بر قطع شدن از واسط معاملاتی به کاربر داده می‌شود. هم‌چنین، واسط معاملاتی، سفارشات باز معاملاتی که جریان تیک به آن‌ها ندارد را می‌بندد. با استفاده از دستور ()Terminate، یک معامله بسته خواهد شد. در نتیجه، در تیک بعدی، درصورتی تابع ()OnStart یک معامله اجرا می‌گردد که قبلاً بسته نشده باشد.

۲ - ساختار کلی

در این زبان، کد برنامه، درون یک فایل با پسوند utl. قرار می‌گیرد و شامل موارد زیر می‌باشد:

- یک یا چند تابع توصیف شده توسط کاربر
- یک تابع اصلی (Main)
- تعدادی تابع ()OnStart و ()OnInit به ازای هر معامله

۲- ۱ - ساختار کلی نحو

زبان UTL به بزرگ و کوچک بودن حروف حساس است و در این زبان، وجود کاراکتر های tab و space در خروجی برنامه تاثیری ندارند. هم‌چنین، جزئیات مربوط به scope و خطوط برنامه در ادامه به تفصیل توضیح داده خواهند شد.

۲ - ۲ - کامنت‌ها

در این زبان، کامنت‌ها به دو صورت تک‌خطی و چندخطی می‌توانند نوشته شوند. تمامی کاراکترهای بعد از کاراکتر //، تا انتهای خط، کامنت به حساب می‌آیند. از سوی دیگر، برای کامنت چندخطی، بدین صورت عمل می‌شود که تمامی کاراکترهای موجود، چه در یک خط نوشته شوند و چه در چند خط نوشته شده باشند، بین کاراکترهای /* تا */، کامنت تلقی می‌شوند. به خطوط ۱۱ و ۱۲ از شکل ۳ توجه فرمایید.

۲ - ۳ - قواعد نام‌گذاری کلاس‌ها، توابع و متغیرها

تمامی اسامی انتخابی در متن برنامه، بایستی که از قواعد نام‌گذاری زیر پیروی کنند:

- تنها از کاراکترهای A...Z، a...z، _ و ارقام تشکیل شده باشند (محدودیتی بر روی تعداد کاراکترهای یک اسم در زبان UTL وجود ندارد).
- با ارقام شروع نگردند.
- نام هر متغیر در هر scope یکتاست ولی در scopeهای درونی، از نام متغیر بیرونی می‌توان استفاده نمود. در نتیجه، به هنگام استفاده از آن scope، متغیر درونی ارجحیت دارد.
- امکان تعریف دو تابع با نام یکسان ولی با prototype های مختلف وجود دارد.
- معادل کلید واژه‌های زبان نباشند.

در جدول زیر تمامی کلیدواژه‌های زبان UTL خلاصه شده است:

int	string	for	while	else	if
continue	try	false	true	float	bool
OnInit	OnStart	throw	return	catch	break
static	shared	void	schedule	double	Main
Digits	BUY	SELL	Bid	Ask	type
Volume	Low	High	Close	Open	Time
Text	Trade	Order	Candle	Exception	RefreshRate
GetCandle	Terminate	Connect	Observe	Print	Preorder
parallel					

۳ - نشانه‌گذاری زمانبند (Schedule)

در زبان UTL، یک نشانه‌گذاری schedule@ وجود دارد که در خطوط ۶۸ و ۶۹ نمونه کد شکل شماره ۳ مشاهده می‌شود. این نشانه‌گذاری برای ایجاد ترتیب بین معاملات می‌باشد. لازم به ذکر است که به صورت پیش فرض، معاملات به صورت سریالی اجرا می‌گردند و در هر لحظه، تنها دو معامله امکان اجرای موازی دارند. این موضوع با عبارت parallel بین دو معامله مشخص می‌شود.

اولویت‌بندی بین دو معامله را می‌توان با دستور preorder مشخص نمود. نکته مهم این است که همواره، تنها یک عبارت schedule در تابع ()main تعریف می‌گردد.

۴ - متغیرها

۴ - ۱ - متغیرهای از پیش تعریف شده

در زبان UTL، تعدادی متغیر سراسری از پیش تعریف شده وجود دارند که می‌توان از هر نقطه از برنامه به آن‌ها دسترسی داشت. مقدار این متغیرها توسط واسط معاملاتی در زمان اجرا مشخص می‌شود؛ به عبارت دیگر، در هر تیک، این مقادیر به روز رسانی می‌گردند و توسط روش‌ها و روند برنامه تغییر نمی‌کنند. متغیرهای از پیش تعریف شده، وضعیت نمودار فعلی زیربازار را در لحظهٔ شروع برنامه، در نتیجهٔ اجرای تابع ()RefreshRate و یا پس از آمدن هر تیک (Tick) منعکس می‌کنند.

لیست متغیرهای از پیش تعریف شده به شرح زیر است:

- Ask - آخرین قیمت فروش ارز کنونی
- Bid - آخرین قیمت خرید ارز کنونی
- Candles - تعداد ستون‌ها (Candle) زیربازار کنونی
- Digits - تعداد ارقام بعد از اعشار در قیمت ارز کنونی

لیست سری‌ها و آرایه‌های از پیش تعریف شده نیز به شرح زیر می‌باشد:

- Time - زمان باز شدن هر ستون در چارت کنونی
- Open - قیمت باز شدن هر ستون در چارت کنونی
- Close - قیمت بسته شدن هر ستون در چارت کنونی
- High - بیشترین قیمت هر ستون در چارت کنونی
- Low - کمترین قیمت هر ستون در چارت کنونی
- Volume - حجم تیک مربوط به هر ستون در چارت کنونی

مقادیر تمام متغیرهای از پیش تعریف شده، در لحظه‌ای که توابع ویژه مثل ()OnStart و ()RefreshRate اجرا می‌شوند، به طور خودکار توسط تابع ()main کاربر به روزرسانی می‌گردد.

متغیرهای از پیش تعریف شده هیچ گاه در سمت راست - قرار نمی‌گیرند و در چنین حالتی کامپایلر بایستی پیغام خطا برگرداند.

۴ - ۲ - متغیرهای استاتیک[15]

در این زبان نیز همانند زبان C++، متغیرهای ایستاتیک قابلیت تعریف دارند و تمامی قوانین عملکردی و نحوی آن‌ها همانند قوانین مربوط به این متغیرها در زبان C++ می‌باشد. خط ۱ از شکل شماره ۳ مثال مربوطه را نشان می‌دهد.

۴ - ۳ - متغیرهای External

معاملات می‌توانند با یکدیگر با استفاده از متغیرهای مشترک ارتباط داشته باشند و بر روی تصمیمات یکدیگر اثر بگذارند. برای این منظور، با تعریف متغیر External، مطابق خط ۲ شکل شماره ۳، می‌توان به این مهم دست یافت. این متغیرها توسط تمامی معاملات قابل دسترسی هستند. نوع این گروه از متغیرها، می‌تواند از هریک از انواع داده‌های قابل تعریف در UTL باشد.

[15] Static

۵ – انواع داده

۵ – ۱ – انواع داده پایه

در زبان UTL، همواره انواع داده‌های پایهٔ int، float، double، string، bool و void وجود دارند. متغیرهایی که از نوع دادهٔ پایه می‌باشند، به جای اشاره‌گری[۱۶] که به خانه‌ای از حافظه اشاره نماید، مقادیر مربوطه مستقیماً ذخیره می‌گردند. تمامی قوانین مربوط به عملکرد، نوع و اندازهٔ تخصیص داده شده به هر یک از این انواع داده، همانند زبان C++ می‌باشد. در ادامه، انواع داده‌هایی که مختص به زبان UTL هستند، بیان شده است.

۵ – ۲ – لیست

لیست در زبان UTL، مشتمل بر تعداد مشخصی داده است که نوع آن‌ها نیز یکسان می‌باشد. نمونه‌های مربوط به تعریف یک لیست در زبان UTL، در خطوط شمارهٔ ۲۲ و ۲۳ نمونه کد شکل شماره ۳ نشان داده شده است. دقت شود که مقداردهی و دسترسی به المان‌های یک لیست، مطابق خط شمارهٔ ۲۶ نمونه کد، به صورت مستقیم انجام می‌گیرد. در صورت مقداردهی المان یا المان‌هایی از لیست با مقادیری که با نوع داده‌ای لیست همخوانی ندارد، با خطای شماره ۵ مواجه خواهیم شد. (مجموعهٔ کامل کدهای مربوط به خطاها در زبان UTL، در بخش شناسایی و پاسخ به خطا بیان شده است).

۵ – ۳ – معامله

در زبان UTL، یک نوع متغیر به نام Trade وجود دارد که اطلاعات بازگشتی از تابع ()Observe را نگهداری می‌کند. لازم به ذکر است که برای یک معامله، این قابلیت وجود دارد که بر روی سفارش[۱۷] متناظر با آن، دو تابع ()Open. و ()Close. فراخوانی گردند. این توابع، به ترتیب، معامله را شروع می‌کنند و به آن خاتمه می‌دهند. دقت

[16] Pointer
[17] Order

شود هر معامله‌ای که شروع شده است، الزاماً بایستی در برنامه خاتمه یابد؛ در غیر اینصورت، با خطا مواجه خواهیم شد.

به ازای هر Trade، باید تابع ()OnStart آن نوشته شود، حتی اگر این تابع فاقد کد باشد. یک Trade، دارای نشانوندهای[18] از پیش تعریف شده است که در بخش ۴ - ۱ شرح داده شده‌اند. این مقادیر، مطابق خطوط ۱۹ و ۲۰ نمونه کد شکل شماره ۳ قابل دسترسی هستند.

۵ - ۴ - سفارش

در زبان UTL، یک متغیر Order وجود دارد که اطلاعات یک سفارش را در خود نگهداری می‌کند. این سفارش، همواره به یک معامله وابسته می‌باشد. یک Order، قابلیت باز (شروع) یا بسته شدن (خاتمه) دارد؛ اما، در صورتیکه شروع گردد، بایستی حتماً خاتمه یابد. مطابق خطوط شمارهٔ ۷ و ۳۲ نمونه کد شکل شماره ۳، سفارش مربوط به یک Trade، در تابع ()OnStart یا ()OnInit مربوط به آن Trade می‌تواند ایجاد گردد. نکتهٔ حائز اهمیت این است که این متغیر، ۴ پارامتر مختلف TakeProfit، Stoploss، type و amount را دریافت می‌نماید و سفارش متناظر با آن‌ها را بر روی Trade مورد نظر ایجاد می‌کند. نوع دادهٔ مربوط به هر یک از پارامترهای TakeProfit، Stoploss و amount می‌تواند از نوع داده‌ای double، float و int باشد و پارامتر type، نوع سفارش، از قبیل خرید یا فروش بودن را تعیین می‌کند که این خود از نوع int است. مترجم زبان UTL به صورت پیش فرض برای BUY (خرید) مقدار 0 و برای SELL (فروش) مقدار 1 را در نظر می‌گیرد. بنابراین، برای خرید، از دستور BUY مطابق خط شماره ۳۲ و برای فروش از دستور SELL مطابق خط شماره ۷ نمونه کد شکل شماره ۳ استفاده می‌شود.

۵ - ۵ - خطا

در زبان UTL یک نوع دادهٔ خطا یا Error داریم که همواره می‌توان به اطلاعات خطاهایی که رخ می‌دهند، از طریق آن‌ها دسترسی پیدا کرد. لازم به ذکر است که ساختار شناسایی و پاسخگویی به خطا نیز در ادامهٔ این مستند

[18] Parameter

و در بخش ۸ ذکر شده است. نوع داده ای خطا مشتمل بر اطلاعاتی از قبیل نوع و متن خطا می باشد. جدول مربوط به انواع خطا در زبان UTL، در بخش ۸ بیان شده است.

۵ – ۶ – ستون[۱۹]

نوع داده‌ای Candle، یکی دیگر از انواع داده در زبان UTL است. این نوع داده‌ای برای نگهداری اطلاعات مربوط به هر Bar یا همان Candle در نمودار Candle stick به کار برده می‌شود. هر ستون، شامل ۶ پارامتر مختلف است که در قسمت ۴ - ۱ نیز به آن‌ها اشاره گردید. به این پارامترها می‌توان همانند خط شمارهٔ ۲۶ شکل ۳ دسترسی پیدا کرد. به بیان دقیق‌تر، ساختار پارامترهای قابل استفاده در این نوع داده‌ای همانند شکل زیر است:

```
Candle{
    string Time;
    float Open;
    float Close;
    float High;
    float Low;
    double Volume;
}
```

۶ – عملگرها[۲۰]

زبان UTL چهار گروه عملگر اصلی دارد که در ادامه هر یک به تفصیل شرح داده خواهند شد.

[19] Candle
[20] Operator

۶ – ۱ – عملگرهای حسابی[۲۱]

این گروه از عملگرها، تنها بر روی اعداد عمل می‌کنند. لیست این عملگرها در جدول زیر آمده است. در مثال‌های بیان شده، A را برابر با ۲۰ و B را برابر با ۱۰ در نظر بگیرید.

عملگر	شرکت پذیری	توضیح	مثال
+	چپ	جمع	A + B = 30
-	چپ	تفریق	A - B = 10
*	چپ	ضرب	A * B = 200
/	چپ	تقسیم	A / B = 2 B / A = 0
%	چپ	باقی مانده	A % B = 10
-	راست	منفی تک عملوندی پیشوندی	-A = -20
-- و ++	راست	پیشوندی	--A
-- و ++	چپ	پسوندی	A ++

۶ – ۲ – عملگرهای مقایسه‌ای[۲۲]

این عملگرها وظیفهٔ مقایسه را بر عهده دارند؛ بنابراین، نتیجهٔ آن‌ها باید برابر مقدار درست[۲۳] و یا نادرست[۲۴] باشد. به عبارت دیگر، خروجی آن‌ها یک مقدرا دودویی[۲۵] است. توجه داشته باشید که عملوندهای مربوط به عملگرهای > و < تنها از جنس اعداد صحیح هستند. دو عملگر == و =! بر روی تمامی انواع داده‌ها تعریف شده‌اند

21 Arithmetic Operators
22 Comparison Operators
23 True
24 False
25 Boolean

و باید نوع عملوندهای آن‌ها یکسان باشد. در مورد لیست‌ها، باید تعداد اعضای دو لیست و نوع اعضای متناظر آن‌ها با هم برابر باشد؛ در غیر این صورت، بایستی خطای کامپایل در نظر گرفته شود. لیست عملگرهای مقایسه‌ای، در جدول زیر بیان شده است. مقادیر A و B را همانند قبل درنظر بگیرید.

عملگر	شرکت پذیری	توضیح	مثال
==	چپ	تساوی	(A == B) = false
=!	چپ	عدم تساوی	(A != B) = true
<	چپ	کوچکتر	(A < B) = false
>	چپ	بزرگتر	(A > B) = true

۶ - ۳ - عملگرهای منطقی[۲۶]

در این زبان، عملگرهای منطقی، هم بر روی نوع bool و هم بر روی نوع عدد (به صورت بیتی) قابل اعمال هستند. این عملگرها در جدول زیر ذکر شده‌اند. در مثال‌های bool، مقادیر A و B به ترتیب برابر true و false می‌باشند.

نوع bool (منطقی)

عملگر	شرکت پذیری	توضیح	مثال
&&	چپ	عطف منطقی (logical AND)	(A && B) = false
||	چپ	فصل منطقی (logical OR)	(A || B) = true
!	راست	نقیض منطقی (logical NOT)	(!A) = false

[26] Logical Operators

نوع عدد

بر روی انواع عددی، این عملگرهای منطقی به این صورت عمل می‌کنند که نمایش بیتی آن عدد را در نظر می‌گیرند و سپس، بر روی نمایش بیتی آن‌ها اعمال می‌گردند. مجدداً یادآوری می‌شوند که در مثال‌های عددی ارائه شده در زیر، A را برابر با ۵ که در مبنای دو برابر است با ۱۰۱ و B را برابر با ۲ که در مبنای دو برابر است با ۱۰ در نظر بگیرید.

عملگر	شرکت پذیری	توضیح	مثال
~	چپ	نقیض بیتی (Bitwise Not)	~ A = 010
>>	چپ	شیفت به راست بیتی، به تعداد پارامتر دوم (در اینجا B) پارامتر اول را به سمت راست شیفت می دهیم.	$A >> B = (001)_2$
>>	چپ	شیفت به چپ بیتی، به تعداد پارامتر دوم (در اینجا B) پارامتر اول را به سمت چپ شیفت می دهیم.	$A << B = (10100)_2$
&	چپ	عطف بیتی (Bitwise AND)	$A \& B = (000)_2$
|	چپ	فصل بیتی (Bitwise OR)	$A \mid B = (111)_2$
^	چپ	نقیض فصل بیتی (Bitwise XOR)	$A \wedge B = (111)_2$

۶ - ۴ - عملگرهای تخصیص[۲۷]

عملگرهای تخصیص UTL، در جدول زیر بیان شده‌اند. لازم به ذکر است که بایستی نوع پارامترهای هر دو سمت عملگرهای تخصیص یکسان باشند؛ در غیر اینصورت، با خطا مواجه می‌شویم. هم‌چنین، توجه داشته باشید که تنها عملگر = بر روی لیست قابل اعمال است و بدین نحوه عمل می‌نماید که تمامی مقادیر عناصر لیست سمت راست را به عناصر لیست سمت چپ تخصیص می‌دهد.

عملگر	شرکت پذیری	توضیح	مثال
=	راست	مقدار عملوند سمت چپ را برابر مقدار عملوند سمت راست قرار می دهد.	X = Y
+=	راست	مقدار عملوند سمت چپ را به اندازه متغیر سمت راست افزایش می دهد و مقدار جدید را جایگزین می کند.	x+=y , x= x + y
=-	راست	مقدار عملوند سمت چپ را به اندازه متغیر سمت راست کاهش می دهد و مقدار جدید را جایگزین می کند.	x-=y, x = x - y
=*	راست	مقدار عملوند سمت چپ را در مقدار عملوند سمت راست ضرب می کند و مقدار جدید را جایگزین می کند.	x*=y, x = x * y
=/	راست	مقدار عملوند سمت چپ را بر مقدار عملوند سمت چپ تقسیم می کند و مقدار جدید را جایگزین می کند.	x/=y, x = x/ y
%=	راست	مقدار باقی مانده عملوند سمت چپ را بر عملوند سمت راست حساب می کند و مقدار جدید را جایگزین مقدار عملوند سمت چپ می کند.	x%=y, x = x%y

[27] Assignment Operators

۶ - ۵ - اولویت عملگرها

اولویت عملگرها به طور کامل و به ترتیب اولویت در جدول زیر آمده است.

اولویت	دسته	عملگر ها	شرکت پذیری
1	پرانتز	()	چپ به راست
2	دسترسی به اعضا، اجرای متد	.	چپ به راست
3	دسترسی به عناصر لیست	[]	چپ به راست
4	تک عملوندی پسوندی	++ و --	چپ به راست
5	تک عملوندی پیشوندی	++ و -- و ! و - و ~	راست به چپ
6	ضرب و تقسیم	% * /	چپ به راست
7	جمع و تفریق	+ -	چپ به راست
8	شیفت بیتی	<< و >>	چپ به راست
9	رابطه و مقایسه	< و >	چپ به راست
10	مقایسه تساوی	=! و ==	چپ به راست
11	عطف، فصل، نقیض فصل بیتی	& و | و ^	چپ به راست
12	عطف منطقی	&&	چپ به راست
13	فصل منطقی	||	چپ به راست
14	تخصیص	=	راست به چپ
15	کاما (ورودی متدها)	,	چپ به راست

۷ - ساختار تصمیم‌گیری[۲۸]

در زبان UTL، تنها ساختار تصمیم‌گیری موجود، ساختار if then else می‌باشد که نحوهٔ استفاده از آن در خطوط شمارهٔ ۱۳، ۳۳، ۳۹ تا ۴۵ در شکل ۳ قابل مشاهده است. دقت شود که این ساختار بدون else نیز می‌تواند به کار رود.

۸ - ساختار خطا

ساختار شناسایی و پاسخ‌گویی به خطا به صورت try … catch می‌باشد. این ساختار در خطوط ۵۹ تا ۶۷ کد نمونهٔ شکل ۳ قابل مشاهده است. دقت شود که Exception یک ساختار داده است که مشتمل بر پارامترهای Text و Type می‌باشد. پارامتر Text، یک رشته کاراکتر است که حاوی پیام خطا می‌باشد و پارامتر Type یک متغیر عددی است که بر اساس آن می‌توان نوع خطا را مشخص نمود. لیست خطاهای موجود در این زبان، در جدول زیر ذکر شده است. توجه کنید که خطایی برای قطع شدن از واسط معاملاتی پس از اتصال نداریم و در صورت بروز چنین خطایی برنامه خاتمه می‌یابد.

Type	نوع خطا
1	خطای ورود
2	خطای کمبود ارز
3	خطای سرریز
4	خطای خارج از بازه آرایه
5	خطای اتصال
6	خطای تقسیم بر صفر

[28] Decision Making Structure

۹ – ساختار تکرار

ساختارهای تکرار در زبان UTL، شامل for و while می باشند که در خطوط ۳۷ تا ۴۶ و ۲۴ تا ۲۷ قابل مشاهده هستند. اصول اساسی for و while به جهت نحو همانند زبان C++ است.

۹ – ۱ – کلید واژه[۲۹] break

در هر کدام از حلقه‌های تعریف شده در زبان UTL، می‌توان با استفاده از کلیدواژهٔ break از حلقه خارج شد و دستورات بعد از حلقه را اجرا نمود. بدیهی است که در صورت استفاده از چند حلقهٔ تودرتو، دستور break بر روی درونی‌ترین حلقه عمل می‌کند و اجرای حلقه‌های بیرونی ادامه پیدا می‌یابد.

۹ – ۲ – کلید واژه continue

در هر کدام از حلقه‌های تعریف شده در زبان UTL، می‌توان با استفاده از کلیدواژهٔ continue تکرار کنونی حلقه را متوقف نمود و تکرار بعدی را آغاز کرد. بدیهی است که در صورت استفاده از چند حلقهٔ تودرتو، دستور continue بر روی درونی‌ترین حلقه عمل می‌کند.

[29] Keyword

۱۰ - قوانین گستره[۳۰]

قوانین مربوط به گستره به صورت زیر است:

- گستره مربوط به توابع ()OnStart، ()OnInit و تابع تعریف شده توسط کاربران سراسری است.
- گستره مربوط به متغیرهای external سراسری است.
- گستره مربوط Trade ها سراسری است.
- گستره مربوط به متغیرهای غیر external از جایی که تعریف شده‌اند است.
- گستره پارامترهای توابع تعریف شدهٔ کاربران، بدنه تابع است.

۱۱ - توابع پیش فرض

- ()Connect - برای اتصال به واسط معاملاتی مورد استفاده قرار می‌گیرد و ورودی‌های این تابع، username و password می‌باشند.
- ()Observe - به منظور مشاهدهٔ زیر بازارها مورد استفاده قرار می‌گیرد و چنانچه زیر بازار مد نظر وجود نداشته باشد، نشان دهندهٔ خطا است.
  - لیست زیر بازارهای موجود در زبان UTL به شرح زیر است:
    - USD/ETH
    - USD/BNB
    - USD/ADA
    - USD/XRP
    - USD/IRR
    - USD/EUR
    - BTC/ETH
    - BTC/BNB
    - BTC/ADA
    - BTC/XRP
    - BTC/IRR
    - BTC/EUR

[30] Scope

- در میان نمادهای بالا، USD دلار، IRR ریال ایران، EUR یورو، BTC بیت‌کوین، ETH اتریوم، BNB بی ان بی، ADA کاردانو و XRP ریپل می‌باشند.

- ()RefreshRate - کاربر می‌تواند به صورت دستی و هر زمان که بخواهد با فراخوانی این تابع، مقادیر متغیرهای از پیش تعریف شده در معامله‌ای که به آن متصل است را پیش از دریافت تیک بعدی به روز رسانی نماید.
- ()Main - برای اتصال به واسط معاملاتی و مشاهدهٔ زیر بازارها مورد استفاده قرار می‌گیرد. لازم به ذکر است که توابع ()Connect و ()Observe و نشانه‌گذاری Schedule تنها در این تابع فراخوانی می‌شوند.
- OnStart(Trade t) - یک تابع اجباری برای هر معامله است که وظیفهٔ انجام معاملات را برعهده دارد.
- OnInit(Trade t) - تابعی است که تنها یک بار در هر برنامه اجرا می‌گردد. تعدادی از متغیرها در این تابع مقداردهی اولیه می‌شوند. بعلاوه برخی Order های مد نظر کاربر می‌توانند در این قسمت تعریف گردند.

- ()Terminate - تابعی است اجباری که پس از فراخوانی، کل سفارش‌های مربوط به معامله متناظر آن را خاتمه می‌دهد و در تیک‌های بعدی، دیگر ()OnStart آن فراخوانی نمی‌شود. هم‌چنین، در صورتی که در تابع ()OnStart فراخوانی شده باشد، معامله مربوط به آن تابع ()OnStart را نیز خاتمه می‌دهد. به بیان بهتر، معامله‌ای که تابع ()Terminate در اثر اجرای تابع ()OnStart متناظر آن فراخوانی شده است، خاتمه می‌یابد.