۱) C یک زبان میانی است – مستقیما به حافظه دسترسی دارد – خوانایی بالایی دارد – قابل انعطاف است – زبان برنامه نویسی سیستم است – قابل حمل است – زبان کوچک اما قدرتمندی است – نسبت به حروف حساس است

۲) گاهی برای خوانده شدن کدها و برای اینکه به چه علت یکسری از کدها را نوشته ایم نیاز است در میان برنامه های نوشته شده یکسری از توضیحات را به عنوان یادداشت درج کنیم تا بعدا خود ما یا سایر برنامه نویسان بتوانند متوجه مفهوم کدها بشوند. گاهی اوقات هم نیاز هست برخی از خطوط از کدهای نوشته شده زمان کامپایل برنامه اجرا نشوند در این مواقع از کامنت ها استفاده می نماییم. در زبان C کامنت ها بین /* و */ و یا بعد از // قرار می گیرند.

۳) هدف از برنامه نویسی ورود داده ها به کامپیوتر ، پردازش و استخراج نتایج آنهاست. نوع داده تعیین کننده مقدار حافظه مورد نیاز و عملگرهای مجاز بر روی آنها می باشد. در زبان C پنج نوع داده وجود دارد که عبارتند از:

Int برای ذخیره اعداد صحیح

Float برای ذخیره اعداد اعشاری

Double برای ذخیره اعداد اعشاری بزرگتر از float

Char برای ذخیره کاراکتر

Void در زبان C از void برای نشان دادن اینکه یک تابع (function) هیچ نوع پارامتری نمی پذیرد، استفاده می شود

۴) متغیر نامی برای کلمات حافظه است که داده ها در آنها قرار می گیرند و ممکن است در طول اجرای برنامه تغییر کنند. متغیر ها محل ذخیره داده ها هستند. برای مقدار دهی به متغیر به سه روش می توان عمل کرد:

۱. هنگام تعریف (تعیین نوع) متغیر
۲. پس از تعریف نوع متغیر و با دستور انتساب (=)
۳. دستورات ورودی

۵) ثوابت مقادیری هستند که در برنامه وجود دارند ولی قابل تغییر نیستند. ثوابت به دو صورت تعیین می شوند:

#define <نام ثابت> <مقدار>

const <نوع داده> <نام ثابت> = <مقدار>

مزیت استفاده از ثوابت این است که اگر بخواهیم در هر بار اجرای برنامه، مقداری را عوض کنیم، لازم نیست که کلیه دستورات برنامه که با آن مقدار سروکار دارند را تغییر دهیم و کافیست آن مقدار را در ثابت تعیین شده مورد نظر تغییر دهیم.

۶)
```
1   int x,y;
2   double ch,d;
3   const unsigned int k = 20;
```

۷) عملگرها نمادهایی هستند که اعمال خاصی را انجام می دهند. باعث راحتی و ساده شدن کدنویسی می گردد.

۸) تقدم عملگرها مشخص می کند که در محاسباتی که بیش از دو عملوند دارند ابتدا کدام عملگر اثرش را اعمال کند.

M = x + y / ۲ * ۳;

۹)
```c
// a = 5(00000101)
int a = 5;

// The result is 10 (00001010)
printf("a is %d\n", a);
printf("a<<1 is %d\n", a<<1);
```

۱۰) این عملگر، عبارتی را ارزیابی کرده، بر اساس ارزش آن عبارت (درستی یا نادرستی) نتیجه عبارت دیگر را در متغیر قرار می دهد.

۱۱)
```c
// Example A
int x;
char ch;
float f;

ch = x;
x = f;
f = ch;
f = x;

// Example B
char ch;
int i;
float f;
double d;

// result1 = int
result1= ch/i;

// result2 = double
result2= f/d;

// result3 = float
result3= f/i;
```

۱۲)
- اگر یکی از عملوندها long double باشد، عملوند دیگر به long double تبدیل می‌شود.
- وگرنه، اگر یکی از عملوندها double باشد، عملوند دیگر به double تبدیل می‌شود.
- وگرنه، اگر یکی از عملوندها float باشد، عملوند دیگر به float تبدیل می‌شود.
- وگرنه، اگر یکی از عملوندها unsigned long باشد، عملوند دیگر به unsigned long تبدیل می‌شود.
- وگرنه، اگر یکی از عملوندها long باشد، عملوند دیگر به long تبدیل می‌شود.
- وگرنه، اگر یکی از عملوندها unsigned int باشد، عملوند دیگر به unsigned int تبدیل می‌شود.

۱۳) اگر داده های ذخیره شده در متغیر صحیح، به اندازه ای کوچک باشد که در بایت کم ارزش قرار گیرد در انتقال مقادیر صحیح به کاراکتری، اطلاعاتی را از دست نخواهیم داد. ولی اگر مقدار موجود در متغیر صحیح، در دو بایت ذخیره شده باشد، بخشی از عدد که در بایت با ارزش آن قرار دارد، از بین می رود. لذا برنامه نویس باید در تبدیل انواع دقت کند تا اطلاعات مفید و ارزشمند خود را از دست ندهد.

۱۴)
```
1   // Example 1
2   int x,y,m,p;
3   x = (y*2) / (m*p);
4
5   // Example 2
6   int x,y,m,r,k;
7   y = (x+m^2) - (k/(r+2));
```

۱۵) در عبارت اول ابتدا مقدار داخل پرانتز محاسبه می شود و در عبارت دوم ابتدا عبارت های ++ و -- محاسبه می گردد سپس ضرب و تقسیم و در آخر جمع

۱۶)
```
1   x = 8
2   m = 6
3
4   y = x*2 < m+4 ? 4*m : 8*m
5   // 8*2=16 < 6+4=10 => False
6   // Result          y=8*6=48
```

۱۷) تعیین نیازمندیهای مسئله – تحلیل مسئله – طراحی الگوریتم حل مسئله – پیاده سازی الگوریتم – تست و کنترل برنامه – نگهداری و نوسازی برنامه

۱۸) شروع ایجاد برنامه

الف. نیازمندی مسئله: بدست آوردن معدل

ب. تحلیل مسئله: گرفتن ۵ ورودی عدد اعشاری از دانش آموز و خروجی معدل دانش آموز

(نمره درس ۱ + نمره درس ۲ + نمره درس ۳ + نمره درس ۴ + نمره درس ۵) / ۵ = معدل دانش آموز

پ. طراحی الگوریتم:

Float Nomreh۱ = input

Float Nomreh۲ = input

Float Nomreh۳ = input

Float Nomreh۴ = input

Float Nomreh۵ = input

Float Moadel = (Nomreh۱+Nomreh۲+Nomreh۳+Nomreh۴+Nomreh۵) / ۵

Print Moadel

ت. پیاده سازی اگوریتم:

```c
#include "stdio.h"

int main() {

    float tedad_nomarat[5] ;
    float jame_nomarat = 0;
    float moadel = 0;
    int i;

    for (i = 0; i < 5; i++)

    {
        printf("\nEnter Score: ");

        scanf("%f", &tedad_nomarat[i]);
    }

    printf("\n\n----------------------------\n\n");

    for (i = 0; i < 5; i++)
    {
        jame_nomarat = jame_nomarat + tedad_nomarat[i];
    }
    moadel = (jame_nomarat / 5);

    printf("The sum is : %.2f \n", jame_nomarat);
    printf("The average is : %.2f \n", moadel);


   getchar();
   getchar();
   return 0;
}
```

لینک GitHub

ث. تست برنامه:

برای تست برنامه داده های خودمان (نمرات) را وارد می کنیم تا ببینیم برنامه درست کار می کند یا خیر

```
[redacted]/mnt/f/university-programming-practice/average-5-score$ ./a.out

Enter Score: 20

Enter Score: 18

Enter Score: 16.5

Enter Score: 15

Enter Score: 19


------------------------------

The sum is : 88.50
The average is : 17.70
```

می بینیم که برنامه درست کار می کند و با وارد کردن ۵ نمره دانش آموز جمع نمرات و معدل آن را برای ما محاسبه کرده و برای ما بر چاپ می کند

ج. نگهداری برنامه:

برنامه در حال حاضر بدون خطا و باگی عمل می کند و اجرا می شود باید در آینده ببینیم مشکلات برنامه چیست و آیا به نوسازی نیاز دارد یا خیر

تهیه کننده : حسین (امیر) اسماعیلی

Www.Iamiresmaeili۷.ir

GitHub