به نام خدا

داک آموزشی رویداد گلابی

مرحله اول

سوال سوم

دانشکده مهندسی کامپیوتر

دانشگاه صنعتی شریف

نیم سال اول ۰۱ـ۰۰

دبیر رویداد:

محمدطه جهانی نژاد

مسئول مرحله اول:

ایمان محمدی

طراحان داک آموزشی سوال سوم:

نگار باباشاه

بهار دیبایی‌نیا

ایمان محمدی

محمد صادقی

ویراستاران داک آموزشی سوال سوم:

نگار باباشاه

ایمان محمدی

شایان صالحی

مسئول لتک داک:

حسین علی‌حسینی

فهرست

مطالب

بخش ۱. printf, putchar, puts

در زبان C تابع‌های مختلفی برای نمایش خروجی وجود دارند. پراستفاده‌ترین آن‌ها printf، putchar و puts می‌باشد که برای استفاده از آن‌ها، لازم است هدر stdio.h اینکلود شده باشد. prototype این توابع به صورت زیر می‌باشد:

- **int printf(const char *format, variables …)**
- **int putchar(int char)**
- **int puts(const char *str)**

printf

در بین این‌ها، تابع printf بیشترین استفاده را دارد. ورودی اول آن رشته‌ای است که قصد داریم آن را چاپ کنیم. می‌توان در رشته از یک سری format specifiers مانند %d یا %f استفاده کرد که با مقدارهایی که در ورودی‌های اضافی می‌آیند جایگزین می‌شوند. اگر عملیات چاپ موفقیت‌آمیز باشد، تعداد کاراکترهایی که چاپ شده‌اند برگردانده می‌شود و در غیر این صورت عددی منفی خروجی این تابع خواهد بود.
شکل کلی format specifiers به صورت زیر است:

- **%[flags][width][.precision][length]specifier**

لیست قسمت‌های مختلف در جدول صفحه بعد آمده است:

f	E	e	d/i	c
Decimal floating point	Scientific notation using E character	Scientific notation using e character	Signed decimal integer	Character
u	s	o	G	g
Unsigned decimal integer	String of characters	Signed octal	Uses the shorter of %E or %f	Uses the shorter of %e or %f
%	n	p	X	x
Prints % character	Nothing printed	Pointer address	Unsigned hexadecimal integer (capital letters)	Unsigned hexadecimal integer

- flags

- (mines sign) : مقدار را چپ‌چین می‌کند (پیش‌فرض راست‌چین است).
+ (plus sign) : علامت اعداد (+ یا -) را نشان می‌دهد.
space : اگر قرار نباشد علامتی نوشته شود، یک فاصله قبل مقدار نوشته خواهد شد.
: اگر همراه با o، x یا X استفاده شود، پیش از مقدارهای به جز صفر، به ترتیب 0، 0x یا 0X نوشته می‌شود. اگر همراه با e، E، f، g یا G استفاده شود، مقدار پرینت‌شده حتما شامل decimal point خواهد بود.
0 : در صورتی که با padding همراه باشد، به جای فاصله، کاراکتر 0 قرار می‌گیرد.

- width

عدد : حداقل تعداد کاراکترهایی که باید چاپ شوند را نشان می‌دهد. اگر تعداد کاراکترها کمتر از این عدد باشد، خروجی با 0 یا فاصله پر می‌شود.
* : در رشته فرمت عرض مشخص نخواهد شد و به جای آن در ورودی‌های بعدی تابع عرض داده می‌شود.

- precision

عدد : برای d، i، o، u، x و X، دقت حداقل تعداد رقم‌ها را مشخص می‌کند. برای e، E و f تعداد رقم‌های اعشار را مشخص می‌کند. برای g و G حداکثر تعداد رقم‌ها را نشان می‌دهد و برای s برابر حداکثر تعداد کاراکترها برای چاپ می‌باشد.
* : در رشته فرمت دقت مشخص نخواهد شد و به جای آن در ورودی‌های بعدی تابع دقت داده می‌شود.

- length

h : ورودی برای integer specifiers به صورت short int یا unsigned short int تفسیر می‌شود.
l : برای integer specifiers، به صورت long int یا unsigned long int تفسیر می‌شود.
L : برای floating point specifiers، به صورت long double تفسیر می‌شود.

putchar

تابع putchar یک کاراکتر به عنوان ورودی دریافت می‌کند و آن را چاپ می‌کند. تابع putchar اگر بتواند کاراکتر را چاپ کند، کست‌شده آن به int و در غیر این صورت مقدار EOF که معادل ۱ ـ است را برمیگرداند.

puts

تابع دیگری که برای نمایش خروجی به کار می‌رود puts می‌باشد که به عنوان ورودی یک رشته دریافت می‌کند و آن را چاپ می‌کند و در انتهای رشته یک کاراکتر خط جدید (\n) اضافه می‌کند. در صورت ارور، این تابع مقدار EOF را برمی‌گرداند و در غیر این صورت مقداری نامنفی خروجی این تابع است.
تابع puts به دلیل سادگی از تابع printf سریع‌تر است. همچنین اگر ممکن است رشته شامل formatting characters مانند %s باشد در حالی که قصدی برای استفاده از آن‌ها نداریم (مثلا رشته‌ای از ورودی دریافت کرده‌ایم که در آن %s وجود دارد)، استفاده از تابع puts ایمن‌تر است.
اگر نخواهیم در انتهای رشته خط جدیدی چاپ شود، می‌توانیم از سینتکس زیر استفاده کنیم:

- **fputs(str, stdout)**

بخش ۲. getch

برای گرفتن کد اسکی مقدار خوانده شده از کیبورد، از تابع getch() که در کتابخانه conio تعریف شده است، استفاده می‌شود. این تابع پارامتر ورودی ندارد و خروجی آن مقدار اسکی خوانده شده از کیبورد است.

ویژگی های تابع

- تابع getch() کنسول خروجی را متوقف می کند تا زمانی که یک کلید فشار داده شود.
- از بافری برای ذخیره کاراکتر ورودی استفاده نمی‌کند.
- کاراکتر وارد شده بلافاصله بدون انتظار برای زدن کلید enter برگردانده می‌شود.
- کاراکتر وارد شده در کنسول نمایش داده نمی‌شود.

از این تابع می‌توان برای پذیرش ورودی‌های مخفی مانند رمز عبور، شماره پین ATMو ... استفاده کرد.
به مثال زیر توجه کنید:

```c
#include <conio.h>
#include <stdio.h>
#include <string.h>

int main()
{
    char pwd[9];
    printf("Enter Password: ");
    for (int i = 0; i < 8; i++) {
        pwd[i] = getch();
        printf("*");
    }
    pwd[i] = '\0';
    printf("\n");
    printf("Entered password: ");
    for (i = 0; pwd[i] != '\0'; i++)
        printf("%c", pwd[i]);
    getch();
    return 0;
}
```

همان طور که مشخص است، ابتدا یک رشته به طول ۸ تعریف می‌کنیم. سپس برنامه با ۸ بار انجام دستور ()getch رشته ای به طول ۸ دریافت کرده و به ازای هر دریافت ، یک * چاپ می‌کند. سپس در انتهای رشته کاراکتر 0 را گذاشته و نهایتاً کل رشته‌ی ورودی را چاپ می‌کند.
فرض کنید ورودی Abcd1234 باشد. در این صورت خوجی این کد به صورت زیر است:

خروجی نمونه

```
Enter Password: ********
Entered password: Abcd1234
```

همان طور که می دانید، برخلاف تابع ()getch، برای استفاده از تابع های ()getchar و ()getc بعد از واردن کردن کاراکتر باید enter وارد شود.
تابع ()getche هم مانند ()getch عمل می‌کند و تابع استاندارد نیست و بدون زدنenter ورودی را دریافت می‌کند ولی ورودی را نمایش نیز می‌دهد.

بخش ۳. rand and srand

بعضی از الگوریتم ها (مانند الگوریتم های لاس وگاس یا مونته‌کارلو) نیاز به یک عدد یا متغیر تصادفی دارند تا خروجی مناسب بدهند.
یکی از روش‌های به دست آوردن یک عدد شبه رندوم (pseudo random number)، استفاده از تابع rand() است. این تابع وقتی اولین بار در برنامه صدا زده می‌شود، از مقداری ثابت به نام seed شروع می‌کند و عددی از روی آن به دست می‌آورد. در دفعات بعدی، با انجام الگوریتم خاصی روی عدد رندوم قبلی تولید شده، عدد جدیدی را خروجی می‌دهد.
خروجی تابع rand()، در بازه‌ی[0, RAND_MAX] است. ROUND_MAX یک ثابت (constant) است که مقدار آن ممکن است در پیاده‌سازی‌های مختلف متفاوت باشد، اما تضمین می‌شود که حداقل برابر با 32767 است.
حال فرض کنید می‌خواهیم با استفاده از این تابع، عددی تصادفی در بازه‌ی [0,100] بیابیم. به این منظور کافیست باقی‌مانده‌ی مقداری که rand()خروجی می‌دهد به 100 را بدانیم.
نکته‌ی مهم درباره تابع rand() این است که اگر در یک برنامه فقط با استفاده از اعداد تصادفی تولید کنیم، بعد از اجرای مجدد برنامه، اعداد تولید شده توسط rand() فرقی با اعداد تولید شده در دفعه‌ی قبل ندارند.
دلیل این اتفاق این است که مقداری که تابع rand() با استفاده از آن شروع به ساخت اعداد شبه تصادفی میکند (که آن را seed می‌نامیم)، در هر بار اجرای برنامه ثابت (و در واقع برابر با مقدار 1) است.
به همین دلیل از تابع دیگری به نام srand() استفاده می‌کنیم تا در هنگام شروع برنامه seed مربوط به تابع rand را توسط آن تعیین کنیم. برای تعیین seed، باید از مقداری که در هر بار اجرای برنامه با دفعات متفاوت باشد استفاده کنیم. یک روش مناسب و استاندارد، استفاده از تابع time() است که در بخش ۸ با آن آشنا می‌شوید! در این صورت با هر بار اجرای برنامه، seed متفاوتی داریم و بنابراین اعداد شبه تصادفی تولید شده به احتمال بالا مشکل یکسان بودن seed را نخواهند داشت.
کد صفحه بعد، شیوه تولید عدد تصادفی به شیوه ذکر شده را نشان می‌دهد.

```c
#include <stdio.h>
#include <stdlib.h>
#include<time.h>
int main(void){
   srand(time(0));
   printf("Randomly generated numbers are: ");
   for(int i = 0; i<5; i++)
      printf(" %d ", rand());
   return 0;
}
```

بخش ۴. system

تابع system، تابعی است که به کمک آن می‌توان در یک برنامه‌ی C یا C++ ، یک دستور مربوط به سیستم عامل را اجرا کرد. هدری که این تابع در آن تعریف شده است stdlib.h می‌باشد و برای استفاده از تابع system، باید آن را include کرد. تعریف تابع system به این صورت است:

- **int system(const char *command)**

پارامتر ورودی آن، یک رشته است که در حقیقت همان دستوری‌ست که به سیستم عامل داده می‌شود. به کمک این تابع، می‌توان هر دستوری که سیستم عامل فراهم کرده است را روی ترمینال اجرا کرد. به عنوان مثال، می‌توان system(”dir”) یا system(”ls”) را اجرا کرد. برخی کاربردهای رایج این تابع، دستور system(”pause”) (که اجرا را تا زمانی که یک کلید فشار داده شود متوقف می‌کند) و دستور system(”cls”) (که نوشته‌های روی صفحه یا ترمینال را پاک می‌کند) هستند.
با این حال، بهتر است از استفاده‌ی بیش از اندازه‌ی تابع system خودداری کنید. چرا که اولا این اجرای این تابع هزینه‌بر است و منابع سخت افزاری زیادی مصرف می‌کند . علاوه بر آن، استفاده از system، برنامه را اصطلاحاً non-portable می‌کند. یعنی ممکن است برنامه‌ای که روی سیستم شما اجرا می‌شود، روی سیستمی دیگر اجرا نشود. به عنوان مثال، دستور system(”pause”) فقط روی سیستمی عاملی اجرا می‌شود که چنین دستوری برایش تعریف شده باشد (مثل ویندوز)، اما ممکن است روی سیستم عامل‌های دیگر مانند مک یا لینوکس اجرا نشود.

بخش ۵. sleep

گاهی مواقع نیاز داریم که برای مدتی برنامه متوقف شود و هیچ کاری انجام نشود یا بعبارتی تاخیری در اجرا به وجود بیاید.تابع ()sleep برای این منظور به کار می آید.

```c
#include<stdio.h>
int main()
{
   printf("Sleeping for 1 second.\n");
   sleep(1);
   return 0;
}
```

همان طور که مشخص است ابتدا رشته داده شده به printf چاپ می‌شود سپس تابع sleep(1) فراخوانی می‌شود. چون ورودی آن 1 است پس برنامه به مدت 1 ثانیه متوقف می‌شود و سپس خاتمه می‌یابد.

بخش ۶. kbhit

این تابع در کتابخانه conio.h وجود دارد پس حتما باید در ابتدای برنامه include شود. از این تابع برای تعیین اینکه آیا یک کلید فشرده شده است یا نه استفاده می‌شود. اگر یک کلید فشار داده شده باشد ، مقدار غیر صفر بر می‌گرداند و در غیر این صورت صفر را بر می‌گرداند.

```
#include <iostream.h>
#include <conio.h>
int main()
{
  while (!kbhit())
    printf("Press a key\n");
  return 0;
}
```

در کد بالا تا وقتی که کاربر کلیدی روی کیبورد را فشار نداده باشد Press a key چاپ می‌شود.
این تابع نیز در کتابخانه توابع استاندارد C موجود نیست.

بخش ۷. Fflush (stdin)

از این تابع برای پاک کردن بافر خروجی استریم استفاده می‌شود و در صورت موفقیت صفر و در غیر این صورت خطای EOF را بر می‌گرداند. هنگام گرفتن یک رشته از ورودی، بافر برای ورودی بعدی پاک نمی‌شود و همان ورودی قبلی در نظر گرفته می‌شود. برای حل این مشکل، باید از fflush(stdin) استفاده کنیم.
به مثال زیر توجه کنید:

```c
#include <stdio.h>
#include<stdlib.h>
int main()
{
  char str[20];
  int i;
  for (i=0; i<2; i++)
  {
    scanf("%[^\n]s", str);
    printf("%s\n", str);
  }
  return 0;
}
```

همان طور که مشخص است، این برنامه ۲ بار از کاربر ورودی می‌گیرد و هر بار آن را چاپ می کند. اما در برخی کامپایلر ها فقط یک ورودی گرفته می‌شود و خروجی یکسان تحویل می‌دهد. علت این است که رشته قبلا در بافر ذخیره شده و استریم هنوز پاک نشده اکنون اگر همین کد را با fflush(stdin) بزنیم برنامه مورد انتظارمان رفتار می‌کند.کد ذکرشده در صفحه بعد آمده است. (البته در برخی کامپایلر ها این رفتار، تعریف نشده است و ممکن است حتی بعد از اضافه کردن دستور هم این مشکل حل نشود.)

```c
#include <stdio.h>
#include<stdlib.h>
int main()
{
  char str[20];
  int i;
  for (i = 0; i<2; i++)
  {
    scanf("%[^\n]s", str);
    printf("%s\n", str);
    fflush(stdin);
  }
  return 0;
}
```

در واقع در اینجا از fflush برای پاک کردن بافر و گرفتن ورودی جدید استفاده کرده‌ایم.

بخش ۸. time.h, clock_t, CLOCK_PER_SECOND, clock()

فرض کنید می‌خواهیم مدت زمانی که طول می‌کشد یک بخش از کد اجرا شود را اندازه بگیریم. برای این کار، کافیست زمان شروع و پایان آن بخش از کد را بدانیم.
به همین منظور، میتوانیم از تابع clock() که در فایل هدر time.h تعریف شده است، استفاده کنیم. این تابع، متغیری از جنس clock_t برمی‌گرداند که مربوط به زمان حال (زمان صدا شدن تابع) در پردازنده است. clock_t در واقع دیتا تایپی برای نمایش زمان است.
ال می‌توانیم تابع clock() را در ابتدا و انتهای آن قسمت از کد صدا بزنیم و مقداری که برمی‌گرداند را در دو متغیر (مانند start، end) ذخیره کنیم. سپس همان طور که گفتیم، باید end – start را محاسبه کنیم. می‌دانیم این مقدار، از جنس clock_t است. کافیست آن را به double کست کنیم تا مقداری از جنس double به دست بیاید. حال برای این که این مدت زمان را به ثانیه تبدیل کنیم، باید حاصل را بر ثابت CLOCK_PER_SEC[1] تقسیم کنیم. نمونه‌ای از اجرای این عملیات را در مثال صفحه بعد می‌بینید:

[1] این ماکرو در هدر time.h تعریف شده است. در حقیقت به تعداد clock tickهای پردازنده در واحد ثانیه اشاره می‌کند.

```c
/* Program to demonstrate time taken by function fun() */
#include <stdio.h>
#include <time.h>

// A function that terminates when enter key is pressed
void fun()
{
  printf("fun() starts \n");
  printf("Press enter to stop fun \n");
  while(1)
  {
    if (getchar())
      break;
  }
  printf("fun() ends \n");
}

// The main program calls fun() and measures time taken by fun()
int main()
{
  // Calculate the time taken by fun()
  clock_t t;
  t = clock();
  fun();
  t = clock() - t;
  double time_taken = ((double)t)/CLOCKS_PER_SEC; // in seconds

  printf("fun() took %f seconds to execute \n", time_taken);
  return 0;
}
```

بخش ۹. difftime

یک روش دیگر برای پیدا کردن اختلاف زمانی بین دو نقطه از برنامه، استفاده از تابع difftime() است. difftime نیز در فایل هدر time.h تعریف شده است. prototype این تابع به این صورت است:

- **double difftime(time_t time1, time_t time2)**

که در آن time1 مربوط به زمان پایان و time2 مربوط به زمان شروع است. خروجی تابع، همان اختلاف دو زمان بر حسب ثانیه و از جنس double است.
همانطور که می‌بینید، متغیرهای time_1 و time_2 ، از نوع time_t هستند. time_t نیز دیتا تایپی برای نمایش زمان است.
حال برای به دست آوردن time_1 و time_2، باید از تابع time استفاده کنیم. Prototype این تابع به این صورت است:

- **time_t time (time_t *second)**

کافیست در قسمت مورد نیاز از کد، time(NULL) را صدا بزنیم. خروجی آن متغیری از جنس time_t است که در واقع زمان اکنون را بر حسب ثانیه (با احتساب این که مبدا زمان UTC, January 1, 1970 00:00:00 است) نشان می‌دهد.
به این طریق نیز می‌توان مانند مثال زیر، مدت زمان اجرای قطعه‌ای از کد را به دست آورد.

```c
#include <stdio.h>
#include <time.h>

int main () {
   time_t start_t, end_t;
   double diff_t;

   printf("Starting of the program...\n");
   start_t = time(NULL);

   printf("Sleeping for 5 seconds...\n");
   sleep(5);

   end_t = time(NULL);
   diff_t = difftime(end_t, start_t);

   printf("Execution time = %f\n", diff_t);
```

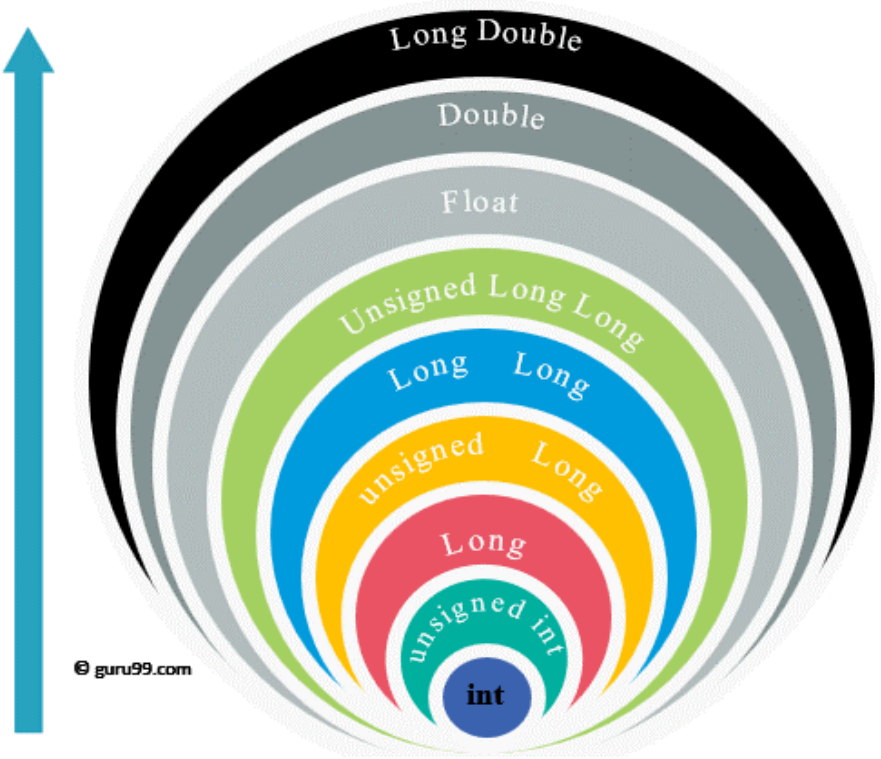
```
18      printf("Exiting of the program...\n");
19
20      return(0);
21  }
```

بخش ۱۰. Type Casting

تبدیل یکdata type به data type ای دیگر، type casting نامیده می‌شود. به عنوان مثال، اگر می‌خواهیم متغیری از جنس long به int تبدیل شود، باید آن را از long به int کست کنیم. به طور کلی می‌توان گفت دو نوع type cast در زبان C وجود دارند:

- **implicit یا ضمنی**

کست کردن به صورت ضمنی توسط کامپایلر انجام می‌شود. وقتی بیش از یک نوع data type در یک عبارت استفاده می‌شوند، کامپایلر آن‌ها را به یک type کست می‌کند تا از از دست رفتن داده‌ها جلوگیری کند. به این منظور data typeها اولویت‌بندی می‌شوند. وقتی دو عملوند با data type متفاوت در یک عبارت ظاهر شده‌اند، عملوندی که data type آن اولویت پایین‌تری دارد، به data type عملوند دیگر کست می‌شود.
کامپایلر ابتدا کاراکترها را به int کست می‌کند. اگر همچنان عملوندهایی با data type متفاوت وجود داشتند، همه‌ی آن‌ها را به data type با اولویت بالاتر در شکل زیر کست می‌کند:



کد زیر را در نظر بگیرید:

```c
#include <stdio.h>
int main() {
   int a = 10;
   char b = 'S';
   float c = 2.88;
   a = a+b;
   printf("Implicit conversion from character to integer : %d\n",a);
   c = c+a;
   printf("Implicit conversion from integer to float : %f\n",c);
   return 0;
}
```

خروجی نمونه

Implicit conversion from character to integer : 93
Implicit conversion from integer to float : 95.879997

توجه کنید که کست implicit نباید روی data typeهای ناسازگار انجام شود. به عنوان مثال تبدیل ضمنی float به int مقدار آن را به کل تغییر می‌دهد و معنی خود را از دست می‌دهد. یا تبدیل long int به int، باعث از دست رفتن بیت‌های پرارزش عدد خواهد شد.

- **explicit یا غیرضمنی**

کست کردن مقادیر از یک نوع به نوعی دیگر به صورت غیر ضمنی (یا explicit)، توسط برنامه‌نویس و با استفاده از cast operator به این صورت انجام می‌شود:

- **(type_name) expression**

کد صفحه بعد و خروجی آن را در نظر بگیرید:

```c
#include <stdio.h>
int main() {
   float c = 5.55;
   int s = (int)c+1;
   printf("Explicit Conversion : %d\n",s);
   return 0;
}
```

خروجی نمونه

```
Explicit Conversion : 6
```

بخش ۱۱. گرفتن فایل خروجی (.exe) در ویندوز

۱. cmd را باز کرده و عبارت gcc –version را وارد کنید. اگر عبارت ‘gcc’ is not recognized را دریافت کردید ، cmd را بسته و به مسیر نصب mingw بروید (اگر از codeblocks استفاده می‌کنید احتمالا در همان مسیر نصب codeblocks وجود دارد) سپس فولدر mingw را باز کرده و سپس فولدر bin را باز کنید.

۲. از URL section بالای صفحه مسیر را کپی کنید.

۳. روی ایکون This PC راست کلیک کرده و properties را انتخاب کنید.

۴. به قسمت advanced system settings رفته و از منوی advanced پنجره باز شده دکمه Environment Variables را بزنید. در قسمت system variables روی Path کلیک کرده و دکمه Edit را بزنید.

۵. از پنجره باز شده دکمه New را زده و در محل ایجاد شده ادرس کپی شده را پیست کنید.

۶. همه در همه پنجره های باز دکمه ok را بزنید.

۷. مجددا cmd را باز کرده وgcc –version را وارد کنید . این بار باید ورژن به درستی شناسایی شود.

۸. اکنون به فولدری بروید که در آن فایل c. مورد نظر وجود دارد. روی قسمتURL section کلیک کرده تا کل آن انتخاب شود و عبارت cmd را در ان تایپ کنید و enter را بزنید. اکنون در پنجره cmd باز شده آدرس فولدر را می‌بینید.

۹. اکنون دستور gcc filename.c را بزنید.(برای مثال gcc main.c)

۱۰. حال در فولدر برنامه فایل a.exe را مشاهده می‌کنید که همان خروجی برنامه شما است.

بخش ۱۲. گرفتن فایل خروجی (.exe) در mac OS

۱. در قسمت ترمینال عبارت gcc -v را وارد کنید اگر اروری مبنی بر معتبر نبودن این دستور دریافت کردید به مراحل بعد و در غیر این صورت به مرحله آخر بروید.

۲. به سایت brew.sh بروید و دستوری که در وسط این صفحه می‌بینید را کپی کنید. (در شکل با رنگ آبی مشخص شده است.)

۳. اکنون این دستور را در ترمینال paste کنید و چند ثانیه صبر کنید تا homebrew نصب شود.

۴. اکنون در ترمینال دستور xcode-select –install را وارد کنید.

۵. اگر با این دستور xcode command tools روی سیستم نصب نشد، نرم افزارXcode را ازmac app store دریافت کنید.(حجمش چند گیگه :))

۶. اکنون دستور brew install gcc را در ترمینال بزنید.

۷. دوباره عبارت gcc -v را در ترمینال وارد کنید. دیگر نباید اروری مبنی بر معتبر بودن دستور ببینید.

۸. با استفاده از دستورات ترمینال وارد دایرکتوری برنامه‌ای که می‌خواهید از آن خروجی بگیرید شوید و دستور gcc -o filename.out filename.c را وارد کنید. (به عنوان مثال: gcc -o hello.out hello.c)

۹. حال در فولدر برنامه فایل خروجی را مشاهده می‌کنید.

بخش ۱۳. گرفتن فایل خروجی (**exe.**) در لینوکس

۱. در ترمینال لینوکس عبارت sudo apt install gcc را بزنید. (در صورت نصب نداشتن gcc)

۲. با استفاده از دستورات ترمینال به دایرکتوری که فایل c. در آن قرار دارد بروید و دستور gcc -o filename filename.c را بزنید. (مثال: gcc -o hello hello.c)

۳. حال در دایرکتوری مورد نظر فایل خروجی را دارید.

بخش ۱۴. ناپدید کردن نمایشگر کنسول

برای اینکه در برنامه اجرا شده خود، نشانگر کنسول (Console Cursor) را ناپدید کنید (برای مثال در سوال بازی دایناسور کروم، با space بازی را انجام دهید و کاری به نشانگر نداشته باشید)، می‌توانید تکه کُد زیر را به برنامه اضافه کنید:

```
#include <windows.h>
void hidecursor()
{
   HANDLE consoleHandle = GetStdHandle(STD_OUTPUT_HANDLE);
   CONSOLE_CURSOR_INFO info;
   info.dwSize = 100;
   info.bVisible = FALSE;
   SetConsoleCursorInfo(consoleHandle, &info);
}
```