به نام خدا

دستور کار کارگاه مبانی کامپیوتر و برنامه‌نویسی

جلسه یازدهم

# اشاره‌گرها و آرگومان‌ها

‍‍۱. برنامه‌ی ساده زیر را در نظر بگیرید:

#include <stdio.h>

int main(int argc, const char \*argv[]) {

int i = 0;

printf("argc: %d\n", argc);

for (i = 0; i < argc; i++) {

printf("argv[%d]: %s\n", i, argv[i]);

}

}

آن را کامپایل کرده و اجرا کنید. خروجی‌ها را بررسی کرده و آنچه فکر می‌کنید این برنامه انجام می‌دهد را با مدرس کارگاه خود در میان بگذارید.

مثال اول

./a.exe

مثال دوم

./a.exe 1

مثال سوم

./a.exe 1 2

به آنچه در اجرای یک برنامه در محیط command line داده می‌شود command line argument می‌گویند. دقت داشته باشید که argc تعداد این آرگومان‌ها و argv آرایه‌ای از رشته‌ها می‌باشد که هر رشته آن یک آرگومان را نمایندگی می‌کند. نکته‌ی قابل توجه بعدی اولین آرگومان است که همواره اسم برنامه می‌باشد.

۲. با استفاده از command line argumentها برنامه‌ای بنویسید که عدد n را به عنوان یک آرگومان از کاربر گرفته و حاصل جمع اعداد ۱ تا n را چاپ می‌کند.

برای مثال:

./a.exe 3

6

./a.exe 4

10

۳. فرض کنید جدول زیر نمایی از حافظه باشد:

|  |  |
| --- | --- |
| Address | Value |
| 1 | 20 |
| 2 | ‘H’ |
| 3 | ‘I’ |
| 4 | 10.2 |
| 5 | ‘S’ |
| 6 | 0 |

در این صورت خروجی قطعه کد زیر چه خواهد بود؟

int \*p = 1;

int \*q = p;

\*p = 20

\*q++;

printf("%d\n", \*p);

q++;

pritnf("%d\n", \*q);

p = 5;

printf("%s\n", p);

p = 2

printf("%s\n", p);

راهنمایی: می‌دانید که رشته‌ها در C می‌بایست Null Terminated باشند از این مفهوم برای کشف یک خطای زمان اجرا در کد فوق استفاده کنید.

۴. خطای برنامه زیر را پیدا کرده و آن را اصلاح کنید. به نظر شما برنامه‌ی فعلی (دارای خطا) درست کار می‌کند؟ فکر می‌کنید علت این اتفاق چیست؟

#include <stdio.h>

int f(int\* p) {

printf("a = %d\n", p); // a = 10?

}

int main() {

int a = 10;

f((int \*) a);

}

۵. تا به حال چگونگی چاپ کردن انواع داده‌ای مختلف با printf را دیده‌اید، اما یک نوع داده‌ای جدید که تازه با آن آشنا شده‌اید اشاره‌گرها می‌باشند. اشاره‌گرها نوع جدیدی نمی‌باشند بلکه عددهایی هستند که آدرسی را در حافظه سیستم شما نمایندگی می‌کنند. آن‌ها را می‌توانید مانند عددها نیز چاپ کنید اما مشکل اصلی آنجاست که آدرس‌های حافظه با توجه به پردازنده می‌توانند ۳۲ بیتی یا ۶۴ بیتی باشند و به همین دلیل بهتر است آن‌ها را به گونه‌ای دیگر چاپ کنیم.

#include <stdio.h>

int g = 5;

int main() {

int a = 10;

int b = 100;

printf("g: %p\n", &g);

printf("a: %p\n", &a);

printf("b: %p\n", &b);

}

برنامه فوق را اجرا کنید و نتیجه را با مدرس کارگاه خود در میان بگذارید.