به نام خدا

دانشکده‌ی مهندسی برق و کامپیوتر دانشکده فنی دانشگاه تهران

مبانی کامپیوتر و برنامه نویسی

نیمسال اول	عنوان:	اساتید:
1402-03	ساختمان داده و لیست‌های پیوندی ۱	دکترمرادی، دکتر هاشمی

در این آزمایشگاه ابتدا ساختمان داده‌ها در زبان C را بررسی می‌کنیم و سپس به بررسی لیست‌های پیوندی می‌پردازیم.

ساختمان داده:

همانطور که می‌دانید، ساختمان داده‌ها در C مجموعه‌ای از متغیرها (حتی از انواع متفاوت) هستند و به صورت زیر تعریف می‌شوند:

```
struct structure_name {
    data_type member_name1;
    data_type member_name2;
    ....
    ....
};
```

برای استفاده از struct، لازم داریم تا پس از تعریف ساختار آن، به تعداد نیاز struct ایجاد کنیم. برای این کار به صورت زیر عمل می‌کنیم:

```
struct structure_name Name;
```

با این کار، یک ساختمان به نام Name و از نوع ساختار structure_name تعریف کرده‌ایم. برای دسترسی به عناصر ساختمان، می‌توان از عملگر . پس از نام ساختمان استفاده کرد. همچنین می‌توان مفهوم struct را با اشاره‌گرها پیاده‌سازی کرد.

پس از مطالعه مطالب لینک‌های زیر و مشاهده فیلم‌هایی که در اختیارتان قرار گرفته است، بخش‌های بعد را انجام دهید.

۱- ساختمان داده در C
۲- اشاره‌گرها و ساختمان داده
۳- توابع و ساختمان داده

۱- انجام دهید!(Structs)

برنامه‌ی زیر را در یک پروژه‌ی جدید اجرا کنید.

```
#include <stdio.h>
#include <stdlib.h>
struct information
{
   float weight;
   float height;
};
int main()
{
    struct information person_1;
    printf("Please enter weight: \n");
    scanf("%f", person_1.weight);
    printf("Please enter height: \n");
    scanf("%f", person_1.height);
    return 0;
}
```

- آیا برنامه به‌درستی اجرا می‌شود؟ تغییرات لازم را جهت اجرای صحیح برنامه اعمال کنید.
- حال این برنامه را با استفاده از typedef بازنویسی کنید.
- عملگر . در عبارت person_1.weight به چه معناست؟ این عملگر چه تفاوتی با -> دارد؟
- حال عبارت زیر را جایگزین عبارت هایلایت شده کنید. سپس برنامه را طوری تغییر دهید تا برنامه به‌درستی اجرا گردد.

```
Information *person_ptr;
```

(راهنمایی: ابتدا باید تخصیص حافظه انجام دهید.)

پاسخ سوالات بالا را با دستیاران آموزشی درمیان بگذارید.(قسمت ۱)

فرض کنید تعداد افرادی که نیاز است تا اطلاعات‌شان را ذخیره کنیم بیشتر باشد و لازم است که از آرایه استفاده کنیم.(این تعداد در قطعه کد زیر مشخص شده است.) حال توابع getInformation و printInformation را به نحوی تکمیل کنید که عملیات گرفتن داده‌ها از ورودی و نمایش آنها در خروجی، توسط این توابع انجام گردد.

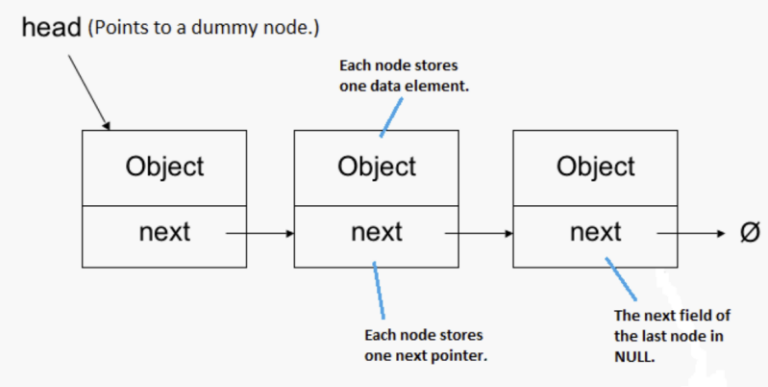
```
#define NUM_OF_PARTICIPANTS ۳

void getInformation (information *person_ptr)
{
    for(int i=0; i<NUM_OF_PARTICIPANTS; i++){
        printf(“Please enter weight for participant %d: \n”, i+1);
        ...
        printf(“Please enter height for participant %d: \n”, i+1);
        ...
    }
}
void printInformation (information *person_ptr)
{
   for(int i=0; i<NUM_OF_PARTICIPANTS; i++){
        printf(“participant %d’s information:\n”, i+1);
        printf(“weight: %f\n”, ...);
        printf(“height: %f\n”, ...);
    }
}
```

* توجه داشته باشید که پس از ایجاد این توابع، باید تغییراتی را در تابع **main** نیز ایجاد کنید!

نتیجه را به دستیاران آموزشی نشان دهید.(قسمت ۲)

لیست‌های پیوندی[1]:

با مفهوم struct در بخش قبلی آشنا شدید. از ترکیب مفاهیم اشاره‌گر، تخصیص حافظه پویا [2] و struct ها می‌توان لیست‌های پیوندی را ساخت. لیست‌های پیوندی از تعدادی گره[3] تشکیل می‌شوند که هرکدام خود یک struct بوده و علاوه بر داده‌های خود، شامل اشاره‌گری به گره بعدی است. تصویر زیر نمایی از ساختار کلی لیست‌های پیوندی ارائه می‌دهد.



برای پیاده سازی لیست های پیوندی نیاز داریم تا از جایی شروع کنیم، پس برای سادگی یک گره بیهوده[4] ایجاد کرده و اشاره‌گری که به آن اشاره می کند را سر[5] لیست پیوندی در نظر می‌گیریم. در قسمت Object این گره بیهوده هیچ اطلاعات معتبری وجود ندارد و در واقع لیست ما از عنصر بعد از این گره (عنصر دوم در شکل بالا) شروع می‌شود.

در قطعه کد زیر ساختار یک گره از یک لیست پیوندی را در زبان C مشاهده می‌کنید:

```
struct structure_name {
    data_type member_name1;
    data_type member_name2;
    ....
    ....
    struct structure_name *next;
};
```

- **توجه:** همانند struct ها، برای ایجاد هرگره از یک لیست پیوندی، از تخصیص حافظه پویا استفاده می‌کنیم.
- **توجه:** لیست‌های پیوندی انواع مختلفی دارند که با آنها در آزمایشگاه بعدی آشنا خواهیم شد.

---

[1] Linked lists
[2] Dynamic memory allocation
[3] Node
[4] Dummy
[5] Head

- اما چرا از لیست‌های پیوندی استفاده می‌کنیم؟ از مزایای لیست‌های پیوندی می‌توان به موارد زیر اشاره کرد:

✓ **پیاده‌سازی:** ساختارهای داده خطی مانند پشته‌ها و صف‌ها اغلب به راحتی با استفاده از یک لیست پیوندی پیاده‌سازی می‌شوند.

✓ **عملیات درج و حذف:** عملیات درج و حذف در لیست پیوندی بسیار ساده‌تر است. برخلاف آرایه‌ها در لیست‌های پیوندی، پس از درج یا حذف یک عنصر، نیازی به جابجایی دیگر عناصر نداریم، فقط باید آدرس موجود در اشاره‌گر بعدی به‌روز شود.

✓ **ساختار داده پویا:** یک لیست پیوندی یک آرایش پویا است که می‌تواند با تخصیص حافظه در زمان اجرا، رشد کرده و کوچک شود. بنابراین نیازی به دادن اندازه اولیه لیست نیست.

- توجه داشته باشید که استفاده از لیست‌های پیوندی معایبی نیز دارد. برای مثال:

✓ **استفاده از حافظه:** در مقایسه با آرایه، به حافظه بیشتری در لیست پیوندی نیاز است. زیرا در یک لیست پیوندی، یک اشاره گر نیز برای ذخیره آدرس عنصر بعدی مورد نیاز است و برای خود به حافظه اضافی نیاز دارد.

✓ **پیمایش:** در یک لیست پیوندی، پیمایش در مقایسه با یک آرایه زمان‌برتر است. دسترسی مستقیم به یک عنصر در لیست پیوندی مانند آرایه بر اساس فهرست امکان پذیر نیست. به عنوان مثال، برای دسترسی به یک گره در موقعیت n، باید تمام گره های قبل از آن را طی کرد.

✓ **پیاده سازی پیچیده:** پیاده سازی لیست پیوندی در مقایسه با آرایه پیچیده تر است.

✓ **اشتراک گذاری مشکل داده ها:** به این دلیل است که دسترسی مستقیم به آدرس حافظه یک عنصر در یک لیست پیوندی امکان پذیر نیست.

! به موارد بالا فکر کنید. به نظرتان لیست‌های پیوندی چه مزایا و معایب دیگری دارند؟ در این مورد تحقیق کنید.

۲- انجام دهید!

قصد داریم تا یک لیست پیوندی متشکل از ۳ گره ایجاد کنیم. قطعه کد زیر را بدین منظور کامل کنید.

```
#include <stdio.h>
#include <stdlib.h>
typedef struct node
{
   int data;
   struct node *next;
}node;
int main()
{
    node *first_node , *second_node , *third_node , *head;
    // Allocating memory
    head = ...;
    ...;
    ...;
    ...;
    // Assigning data values
    first_node->data = 1;
    second_node->data = 2;
    third_node->data = 3;
    // Connecting nodes
    head->next = ...;
    ...;
    ...;
    ...;
    return 0;
}
```

نتیجه را به دستیاران آموزشی نشان دهید.(قسمت 3)

حال می‌خواهیم عملیات تخصیص حافظه در برنامه قسمت قبل، توسط یک تابع که خروجی آن از جنس اشاره‌گر به یک گره است، انجام شود. تابع زیر را طوری تکمیل کنید که هربار یک گره را ایجاد کرده و اشاره‌گر به آن گره را بازگرداند.(توجه کنید که این تابع ورودی نمی‌گیرد!)

```
node* CreateNode(){
    node *the_node;
    // Allocating memory
    the_node= ...;
    the_node->next=NULL;
    return ...;
}
```

پس از تعریف این تابع، چه تغییراتی باید در تابع main ایجاد کنیم؟ برای دستیاران آموزشی توضیح دهید. (قسمت۴)

3 – انجام دهید!

حال فرض کنید تعداد گره‌ها را در اختیار داشته باشیم. برنامه زیر را به طوری تکمیل کنید که یک لیست پیوندی با تعداد مورد نیاز یعنی NUM_OF_NODES گره ایجاد شود.

```
#include <stdio.h>
#include <stdlib.h>
#define NUM_OF_NODES 3
typedef struct node
{
   int data;
   struct node *next;
}node;
node* CreateNode(){
    //قطعه کد قسمت قبل را در اینجا قرار دهید
}
int main()
{
    node *head , *temp;
    // Allocating memory
    head = CreateNode();
    temp = head;
    for(int i=0; i<NUM_OF_NODES; i++){
        ...;
        ...;
    }
    return 0;
}
```

فکر کنید و پاسخ دهید:

- چرا نیازی نیست اشاره‌گر next در گره آخر را در پایان برابر با NULL قرار دهیم؟

- چرا temp را تعریف کردیم؟

نتیجه را به دستیاران آموزشی نشان داده و روند اجرای برنامه را توضیح دهید.(قسمت 5)

برای آنکه مقدار data در هر گره برابر با شماره گره باشد، چه‌کار باید بکنیم؟ قطعه کد زیر را بدین منظور تکمیل کرده و آن‌را به برنامه قسمت قبل اضافه کنید.

```
// Assigning data values
 temp = ...;
 for(int i=0; i<NUM_OF_NODES; i++){
     ...;
     temp->data = i+1;
 }
```

حال قطعه کدی را به برنامه اضافه کنید تا مقدار data را برای همه‌ی گره‌ها به‌ترتیب چاپ کند. در داخل حلقه for که بدین منظور نوشته‌اید، از عبارت زیر استفاده کنید:

```
printf("data for node %d = %d\n",...
```

نتیجه را به دستیاران آموزشی نشان دهید.(قسمت 6)

۴- انجام دهید!

اکنون قصد داریم برنامه‌ای را که در بخش اول نوشتیم، با لیست‌های پیوندی پیاده سازی کنیم. برای این کار تغییرات لازم را در آن برنامه اعمال کنید.

- **راهنمایی:**

- لازم است همانند قسمت قبل ابتدا توسط یک حلقه، گره‌ها را ایجاد کنید.(بدیهی‌ست به تابع CreateNode که در قسمت های قبل نوشتیم نیاز دارید.)

- عبارت زیر را به ساختار information اضافه کنید:

```
struct information *next;
```

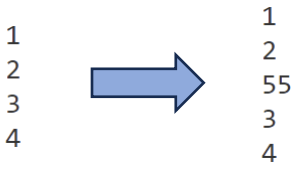
- در تابع main، از توابع getInformation و printInformation به صورت زیر استفاده کنید:

```
getInformation (head);
printInformation (head);
```

نتیجه را به دستیاران آموزشی نشان دهید.(قسمت ۷)

۵- فکر کنید!

در برنامه زیر یک آرایه از اعداد به طول SIZE ایجاد کرده‌ایم. حال می‌خواهیم یک خانه با مقدار VALUE را به این آرایه به صورتی اضافه کنیم که این خانه‌ی جدید خانه‌ی شماره‌ی POSITION در آرایه نهایی باشد. برای مثال آرایه‌ی اولیه و آرایه نهایی به ازای مقادیر SIZE=4 و POSITION=3 و VALUE=5 به صورت زیر باید باشد:

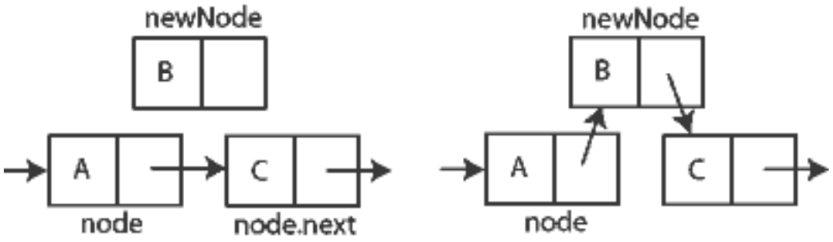



```c
#include <stdio.h>
#include <stdlib.h>
#define SIZE 4
#define POSITION 3
#define VALUE 55
int main()
{
    int *array;
    array = (int*)malloc(SIZE * sizeof(int));
    // Assigning values
    for (int i=0; i<SIZE; i++){
        *(array+i)=i+1;
    }
    // Printing the array
    for (int i=0; i<SIZE; i++){
        printf("%d\n",*(array+i));
    }
    printf("********\n");
    // Reallocating memory
        array = realloc(array , ... );
    // Shifting
    ...
    // Assigning value
    ...;
    // Printing the array again
    for (int i=0; i<SIZE+1; i++){
        printf("%d\n",*(array+i));
    }
    return 0;
}
```

این برنامه را تکمیل نموده و نتیجه را به دستیاران آموزشی نشان دهید.(قسمت ۸)

- به نظرتان در قسمت قبل اگر به جای آرایه از یک لیست پیوندی استفاده می‌کردیم، چه اتفاقی می‌افتاد؟ عملیات اضافه کردن راحت‌تر انجام نمی‌شد؟
- حال فرض کنید به جای آرایه، یک لیست پیوندی داریم که می‌خواهیم یک خانه به وسط آن اضافه کنیم؛ چه گام هایی را باید انجام دهیم؟ چه خطایی ممکن است در این حالت رخ دهد؟

راهنمایی: می‌توانید از تصویر زیر کمک بگیرید.



- **توجه:** مفهوم اضافه کردن[6] و باقی مفاهیم لیست‌های پیوندی را در آزمایشگاه بعدی به‌طور مفصل بررسی خواهیم کرد.

پاسخ سوالات بالا را به دستیاران آموزشی توضیح دهید.(قسمت ۹)

۶ – انجام دهید!

برای هر قطعه کد زیر به سوالات جواب دهید. هر مرحله را به دستیاران نشان دهید

۱. فایل LinkedList_Lab9.1.c را دانلود کرده و اجرا کنید. آیا خطا میدهد؟ چرا؟ راه حل چیست؟

۲. فایل LinkedList_Lab9.2.c را دانلود کرده و اجرا کنید. آیا خطا میدهد؟ چرا؟ راه حل چیست؟

a. در این مورد دقت نمایید که دو اشکال وجود دارد. یک اشکال ممکن است که خطای ظاهر نداشته باشد.

b. خطا میتواند در مواقع و حالات مختلف اتفاق بیفتد. مثلا زمانی که لیست خالی است یا ... در این موارد دقت کنید.

موفق باشید.

تهیه و تنظیم: امیرمرتضی رضائی

[6] Insertion