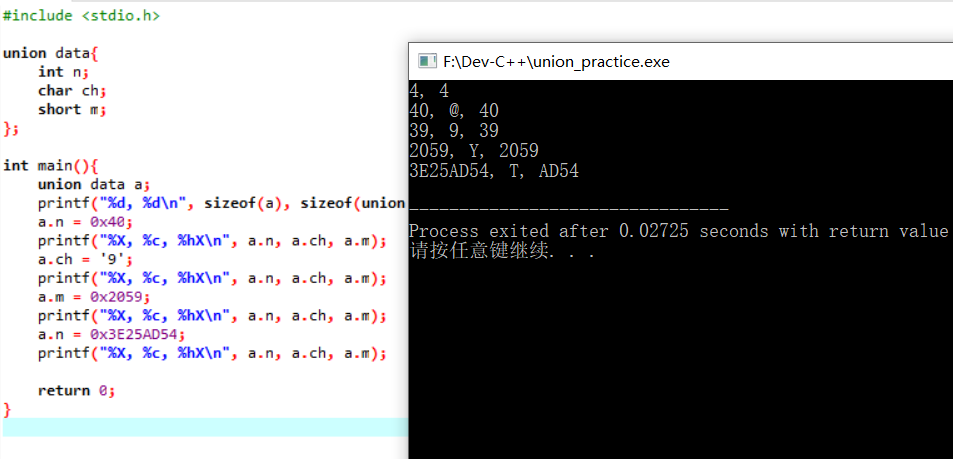
# 1、



1. 对于第一行输出，分析：共用体所占空间即为其所含最大类型变量所占空间；
2. 对于第二行输出，分析：0x40存在4个byte中的第一个，故in和short都可输出，其中%hx是将short类型按16进制输出的格式控制符。且0x40对应的10进制是64，以64为ASCII码的是@;
3. 对于第三行输出，分析：‘9’所对应的ASCII码是57，换算成16进制就是0x39，存在第一个byte中，故short也可输出；
4. 对于第四行输出，0x2059将0x59（也就是低位）存在第一个byte中，将0x20存在第二个byte中，故short也可输出，但char仅对应0x59，换算成10进制就是89，也就是ASCII码中的Y；
5. 对于第五行输出，从第一个byte到第四个byte，依次存储 0x54,0xAD,0x25,0x3E，故short仅输出AD54，char仅对应0x54，换算成10进制就是84，也就是ASCII中的T。

# 2、

#include <stdio.h>

#include <stdlib.h>

/\* run this program using the console pauser or add your own getch, system("pause") or input loop \*/

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

struct address\_format

{

int srtsubnet;

int srcnode;

union{

int dstsubnet;

int dstgroup;

};

union{

int dstnode\_2a;

struct{

int dstnode\_2b;

int group;

int grpmemb;

};

int unique\_node\_id[6];

};

};

return 0;

}

# 3、

//node\_practioce\_main.c

#include <stdio.h>

#include <stdlib.h>

#include "node\_practice\_operation.h"

/\* run this program using the console pauser or add your own getch, system("pause") or input loop \*/

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

printf("-------create with 1 node-------\n");

INFO info1={2000};

node \*head=create(info1);

printf("length=%d\n",length(head));

print(head);

printf("\n");

printf("--------Insert after 1st Node-----\n");

INFO info2={2001};

insert(head,info2,1);

printf("length=%d\n",length(head));

print(head);

printf("\n");

printf("-------Insert after 2nd Node-----\n");

INFO info3={2002};

insert(head,info3,2);

printf("length=%d\n",length(head));

print(head);

printf("\n");

printf("-----------Delete 3rd Node------\n");

delete(head,2);

printf("length=%d\n",length(head));

print(head);

printf("\n");

printf("--------Destroy all nodes-------\n");

destory(head);

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

printf("-------create several nodes -----\n");

head=create(info1);

insert(head,info2,1);

insert(head,info3,2);

printf("length=%d\n",length(head));

print(head);

printf("\n");

// printf("------After bubble\_sort1 ----\n");

// bubble\_sort1(head);

// print(head);

// printf("\n");

// printf("------After bubble\_sort2-----\n");//wrong

// bubble\_sort2(head);

// print(head);

// printf("\n");

// printf("------After selection\_sort------\n");

// selection\_sort(head);

// print(head);

// printf("\n");

printf("-----After insertion\_sort-------\n");

head =insertion\_sort(head);

print(head);

printf("\n");

// printf("------After quick\_sort---------\n");\\amazing, it works

// quick\_sort(head);

// print(head);

// printf("\n");

}

//node\_practice\_operation.c

#include "node\_practice\_operation.h"

node \* create(INFO info)//only used to create and return a head node

{

node \* head;

head=malloc(sizeof(node));

if(NULL==head)

{

printf("Error: malloc failed in create \n");

}

head->info=info;

head->next=NULL;

return head;

}

int length(node \* head)//include head

{

int len=0;

while(NULL!=head)

{

head=head->next;

len++;

}

return len;

}

node\* insert(node \* head,INFO info,int position)//插入到head之后的第position个位置

{

if(position<0 || position>length(head))

{

printf("Illegal position in insert");

return;

}

node\* new\_node=malloc(sizeof(node));

if(NULL==new\_node)

{

printf("Error: malloc failed in insert \n");

}

new\_node->info=info;

if(position==0)

{

new\_node->next=head;

head=new\_node;

}

else

{

while(--position)head=head->next;

new\_node->next=head->next;

head->next=new\_node;

}

return new\_node;

}

void delete(node\* head,int position)//删除head之后的第position个节点

{

if(position<1 || position>length(head))

{

printf("Illegal poisition in delete");

return;

}

while(--position)head=head->next;

head->next=head->next->next;

}

void print(node \*head)

{

while(NULL!=head)

{

printf("the score of the node is: ");

printf("%d\n",head->info.score);

head=head->next;

}

}

void destory(node\* head)

{

int count=length(head);

while(--count)

{

delete(head,count);

}

free(head);

}

void swap(int \*a,int \*b)

{

int temp=\*a;

\*a=\*b;

\*b=temp;

}

void bubble\_sort1(node\* head)//soga,in my opinion...in fact,it's selection sort

{

for(node\* p=head;NULL!=p; p=p->next)

{

for(node\* q=p;NULL!=q;q=q->next)

{

if(q->info.score > p->info.score)

{

swap(&p->info.score,&q->info.score);//jiao huan liang jie dian weizhi

}

}

}

}

void bubble\_sort2(node\* head)//bubble sort is j & j+1, but not i & j(selection) emmm maybe

{

int len=length(head);

int flag;

node \*c1,\*c2;

for(int i=0;i<len-1;i++)

{

flag=0;

for(int j=0;j<len-1-i;j++)

{

c1=get\_node(head,j);

c2=c1->next;

if(c1->info.score < c2->info.score)

{

swap\_info(&c1->info,&c2->info);

flag=1;

}

}

if(flag==0)

{

break;

}

}

}

void swap\_info(INFO\* a,INFO\* b)

{

INFO temp=\*a;

\*a=\*b;

\*b=temp;

}

node\* get\_node(node\* head, int pos)//return NO.pos node after head, just like a arrray

{

if(pos<0 || pos>=length(head))

{

printf("Illegal position in get\_node");

return NULL;

}

while(NULL!=head && pos--)head=head->next;

return head;

}

void selection\_sort(node \* head)//from big to small

{

node \*c1,\*c2;

int len=length(head);

for(int i=0;i<len;i++)

{

int flag=i;

for(int j=i+1;j<len;j++)

{

c1=get\_node(head,j);

c2=get\_node(head,flag);

if(c1->info.score > c2->info.score)

{

flag=j;

}

}

if(i!=flag)

{

c1=get\_node(head,i);

c2=get\_node(head,flag);

swap\_info(&c1->info,&c2->info);

}

}

}

node\* insertion\_sort(node\* head)

{

int len=length(head);

node \*c1,\*c2;

for(int i=1;i<len;i++)

{

c1=get\_node(head,i);

int flag=i;

for(int j=i-1;j>=0;j--)

{

c2=get\_node(head,j);

if(c1->info.score > c2->info.score)

{

node\* p=insert(head,c1->info,j);

if(j==0)

{

delete(head,flag);

head=p;

}

else

{

delete(head,flag+1);

}

}

else

{

c2->next=c1;

}

flag--;

}

}

return head;

}

void quick\_sort\_recursive(node\* head,int start,int end)//just like a array

{

if(start>=end)return;

int left=start,right=end;

INFO mid=head->info;

INFO com1,com2;

while(left<right)

{

do

{

left++;

com1=get\_node(head,left)->info;

}while(com1.score <= mid.score && left<end);

do

{

com2=get\_node(head,right)->info;

right--;

}while(com2.score >= mid.score && right>start);

right++;//使right与com2对应

if(left<right)

{

swap\_info(&com1,&com2);

}

}

swap\_info(&mid,&com2);

quick\_sort\_recursive(head,start,left-1);//left is the first number of the higher area

quick\_sort\_recursive(head,left,end-1);

}

void quick\_sort(node\* head)

{

int len=length(head);

quick\_sort\_recursive(head,0,len-1);

}

//node\_practice\_operation.h

#include <stdio.h>

#include <stdlib.h>

#include "node\_practice\_struct.h"

node \* create(INFO info);//only used to create and return a head node

int length(node \* head);//include head

node\* insert(node \* head,INFO info,int position);//插入到head之后的第position个位置

void delete(node\* head,int position);//删除head之后的第position个节点

void print(node \*head);

void destory(node\* head);

void swap(int \*a,int \*b);

void bubble\_sort1(node\* head);//soga,in my opinion...in fact,it's selection sort

void bubble\_sort2(node\* head);//bubble sort is j & j+1, but not i & j(selection) emmm maybe

void swap\_info(INFO\* a,INFO\* b);

node\* get\_node(node\* head, int pos);//return NO.pos node after head, just like a arrray

void selection\_sort(node \* head);//from big to small

node\* insertion\_sort(node\* head);

void quick\_sort\_recursive(node\* head,int start,int end);//just like a array

void quick\_sort(node\* head);

//node\_practice\_struct.h

typedef struct {

int score;

}INFO;

typedef struct mystruct{

INFO info;

struct mystruct\* next;

}node;