Указатели на функции при обработке массивов структур

Использование указателей на функции

Пример с файлом данных по студентам с соответствующим массивом структур (лекции ранее).

В структуре ***student*** есть 3 числовых поля — год рождения (**`year_of_birth`**), курс (**`year`**) и средний балл (**`average`**).

Задача: отсортировать массив структур по заданному числовому полю.

Вариант 1: написать соответствующие функции сортировки и общую функцию **`SortKind()`**, которая будет в качестве одного из параметров принимать ***указатель (имя) функции сортировки***.

См. пример lect-12-02.c

Использование указателей на функции

```
typedef struct student studs;
…
void SortRating(studs **str0, int n0);
void SortYear(studs **str0, int n0);
void SortCourse(studs **str0, int n0);

void SortKind(int n, studs **str0, void (*funcName)(studs**, int))
{
   funcName(str0, n);
}
```

Использование (в `main()`)

```
SortKind(n,stud0,SortCourse);
```

`stud0` — динамический массив указателей на элементы типа `studs`, `n` — количество элементов.

СПбГЭТУ «ЛЭТИ»
ПЕРВЫЙ ЭЛЕКТРОТЕХНИЧЕСКИЙ

Использование указателей на функции

Вариант 2: написать универсальную функцию сортировки, которая в качестве параметра получает значение поля для сортировки (пример lect-12-03.c).

```
void SortKind(int n, studs **str0, float (*funcName)
(studs**, int)) {
   studs *tmp_struct;
   int i, j;
    for(i=0;i<n;i=i+1)
    {
        for(j=0;j<n-i-1;j=j+1)
        {
            if(funcName(str0,j)>funcName(str0,j+1))
            {
                tmp_struct=str0[j];
                str0[j]=str0[j+1];
                str0[j+1]=tmp_struct;
            } } } }
```

Использование указателей на функции

```
float RatingValue(studs **str0, int i0)
{
    return str0[i0]->average;
}

float YearValue(studs **str0, int i0)
{
    return str0[i0]->year_of_birth;
}

float CourseValue(studs **str0, int i0)
{
    return str0[i0]->year;
}
```

Использование (в `main()`) `SortKind(n,stud0,CourseValue);`

Служебные функции возвращают тип float, но при выводе результата можно сделать требуемый формат

СПбГЭТУ «ЛЭТИ»
ПЕРВЫЙ ЭЛЕКТРОТЕХНИЧЕСКИЙ

Массив указателей на функции

Поскольку есть несколько однотипных функций, можно создать массив указателей на функции, тогда выбор функции можно оформить через ввод переменной (номера функции) и выбор элемента массива по номеру (индексу) (пример lect-12-04.c).

Объявления и реализации — те же, что и в предыдущем случае.

В `main()`:

```
float (*kind[3])(studs**, int); // описание массива
…
kind[0] = YearValue;
kind[1] = CourseValue;
kind[2] = RatingValue;
```

Массив указателей на функции

В main():

```
do
{
   CLS;
   printf("Kinds of sort:\n");
   printf("1 - by year of birth\n");
   printf("2 - by year of education\n");
   printf("3 - by rating\n");
   printf("Enter your choice: ");
   scanf("%d",&option);
} while((option<1)||(option>3));
…
SortKind(n,stud0,kind[option-1]);
```

Динамический массив указателей на функции

Массив функций можно сделать динамическим.

Это полезно, если есть расширяемая библиотека функций (`.h`-файл и соответствующий `.c`-файл).

Тогда можно посчитать количество прототипов в `.h`-файле (`m`) и как-то получить их имена (обработать строки объявления прототипов).

Для динамического массива указателей на функции тоже требуется очистка памяти (пример lect-12-05.c).

В `main()`:

```
float (**kind)(studs**, int)=NULL; //описание массива
…
m=3; // нужно откуда-то получить количество элементов!
kind=(float(**)(studs**,int))malloc(m*sizeof(float(*)
(studs**,int))); // проверка на NULL – обязательна!
```


СПбГЭТУ «ЛЭТИ»
ПЕРВЫЙ ЭЛЕКТРОТЕХНИЧЕСКИЙ

Динамический массив указателей на функции

Если ввести новый тип — указатель на функцию сортировки (через **typedef**), то несколько упрощается оператор для выделения памяти (пример lect-12-05a.c).

```
typedef float(*sorting)(studs**,int);
```

В **main()**:

```
sorting *kind=NULL;
…
m=3;
kind=(sorting*)malloc(m*sizeof(sorting));
```