Пользовательские типы данных

Лекция №10

Исходные предпосылки

- Базовые типы данных позволяют представлять простые объекты
- На практике приходится работать со сложными объектами, имеющими много свойств
  - Студент
  - Товар
- Представление информации о сложных объектах в виде набора простых переменных ведет к ошибкам

Составные типы данных языка Си

- Структуры (struct)
  - Последовательное хранение данных
- Объединения (union)
  - Параллельное хранение данных
- Классы (class)
  - Объединение данных и функций

- Составные типы данных приближают программирование к реальной жизни
- Они позволяют обойтись без СУБД в небольших приложениях

Структурированный тип данных

- Имеет уникальное имя
- Состоит из нескольких полей
- Для каждого поля указываются имя и тип
- Используется для описания программных объектов:
  - переменных,
  - массивов
  - указателей
- !!! Описание типа и описание объекта – разные вещи

Составной тип данных содержит

- элементы простых типов данных
  - int a;
  - float *p;
  - char s[5];
- элементы ранее созданных составных типов данных
  - date dd;
  - address *ap;
- прототипы функций (в классах)

Формат описания структуры

```
struct имя_типа
   {тип_поля1 имя_поля1;
    тип_поля2 имя_поля2;
    ……….
     тип_поляN имя_поляN;
    }; //в конце описания типа обязательна ;
```

//описание типа и описание переменной – разные вещи!!!

Пример описания структурированного типа

```
struct student
{
   char fio[32];
   int age;
   float rating;
}

//Память выделяется при создании переменных!
student st1, Ms[3], *pstud;
```

Способы заполнения структур

- Путем инициализации

```
student St1={“Ivanov”,19,4.5};
student St[]={{“Sidorov”,20,4.7}, {“Petrov”,18,4.6}};
```

- В процессе работы программы

```
strcpy(St1.fio,”Antonov”);
St1.age=21;
St1.rating=4.3;
```

Операции над структурами

- = - для всей структуры
  - mS[0]=St1;
- . – обращение к полю путем прямой адресации
- -> - обращение к полю путем косвенной адресации через указатель

- Все операции арифметические и логические операции выполняются только над отдельными полями.
- Переопределение операций над сложными объектами доступно в классах

Работа с массивами структур

- Заполнение с клавиатуры
- Печать на экране
- Поиск полей с определенным свойством.

Ввод массива структур

```
student S[3];
int n=3, i;
for(i=0;i<n;i++)
{
printf(“fio? “);
scanf(“%s”,S[i].fio);
printf(“age? “);
scanf(“%d”,&S[i].age);
printf(“rating? “);
scanf(“%f”,&S[i].rating);
}
```

Вывод массива структур

```
for(i=0;i<n;i++)
{
printf(“fio =%s age=%d rating=%f \n“,
S[i].fio, S[i].age, S[i].rating);
}
```

Операции над полями массивов структур

- Усреднение
- Поиск минимума/максимума
- Поиск элемента с заданным значением
- Подсчет элементов, соответствующих признаку
- Упорядочение по одному из полей

Функция вычисления среднего возраста

```
float fSrVozr(student S[], int n)
{int sm=0,i;
 for (i=0;i<n;i++)
    sm=sm+S[i].age;
return((float)sm/n);
}
```

Функция нахождения указателя на элемент с минимальным возрастом

```
Student* fMinVozr(student *p, int n)
{student *qmin,*q;
 for (qmin=q=p;q<p+n;q++)
   if(q->age<qmin->age)
          qmin=q;
return(qmin);
}
```

Упорядочение студентов по фамилиям

```
void fSortFio(student *p, int n)
{student *t,*q,tmp,*tm;
 for (q=p;q<p+n-1;q++)
    {tm=q+1;
    for (t=q+1;t<p+n;t++)
            if(strcmp(t->fio, tm->fio)<0)
                    tm=t;
     tmp=*t;
    *t=*tm;
     *tm=tmp;
       }
}
```

Вызовы функций по обработке структур

```
#include<stdio.h>
#include<string.h>
void main(void)
{student *p,*q,*t;
int n,s,m;
printf(“n=”);
scanf(“%d”,&n);
p=new student[n];
If(p==NULL)
 return;
```

ПРОДОЛЖЕНИЕ

```
for(i=0;i<n;i++)
{
printf(“fio? “);
scanf(“%s”,p[i].fio);
printf(“age? “);
scanf(“%d”,&p[i].age);
printf(“rating? “);
scanf(“%f”,&p[i].rating);
}
s=fSrVozr(p,n);
t=fMinVozr(p,n);
fSortFio(p,n);
delete p;
}
```

Объединения

```
union field
{int a;//1 переменная 4 байта
 char c[4];// 4 элемента по 1 байту
};

field f;
char b;
f.a=1;
b=f.c[0];
```

Объединение для представления объектов с разными параметрами

- union figure
- {float radius;
- float sides[2];
- };

- figure c,d, Sc, Sd;
- c.radius=3;
- s.side[1]=3;
- s.side[2]=4;
- Sc=3.14*c.radius*c.radius
- Sd=s.side[1]*s.side[2];

Выводы

- Структуры позволяют в удобной форме представлять информацию о свойствах сложных объектов
- Тип данных и объект - разные вещи
- Целиком объект можно только присваивать
- Остальные операции только над отдельными полями
- Структуры позволяют организовывать простые базы данных, когда использование специализированных СУБД нецелесообразно