CPROUC/2021-2022

Jesse op den Brouw

CPROUC

Structures

DE HAAGSE HOGESCHOOL

Structure

- Een structure is een samengesteld datatype.
- De algemene gedaante van een structure is:

```
struct struct-identifier {
    datatype member1;
    datatype member2;
    ...
};
```

- De variabelen in de structure worden *members* genoemd.

Structure – definitie

- Voorbeeld met directe definitie:

```
struct student_struct {
    int idcode;
    char achternaam[30];
    char klas[10];
    double resultaat;
} student1, student2;
```

- Bij directe definitie van variabelen mag *struct-identifier* weggelaten worden.

Structure – definitie

- Voorbeeld met indirecte definitie:

```
struct student_struct student1, student2;
```

- Nu mag *struct-identifier* niet weggelaten worden.

Structure – gebruik members

- De members zijn te benaderen met de *member operator* .

```
struct student_struct student1;


student1.idcode = 19001234;
strcpy(student1.achternaam, "Alberts");
strcpy(student1.klas, "ep13");
student1.resultaat = 7.3;
```

- Nu mag *struct-identifier* niet weggelaten worden.

Structure – initialisatie

- Een structure mag bij definitie gelijk geïnitialiseerd worden:

```
struct student_struct student2 =
  { 19001235, "Blokland", "ep14", 6.9 };
```

- Let op de juiste volgorde bij initialisatie.
- Vergeet de ; niet aan het einde.
- Gehele structure op 0 zetten:

```
struct student_struct student2 = { 0 };
```

- Integers worden met 0 geïnitialiseerd, floats/doubles worden met 0.0 geïnitialiseerd, strings worden met null-bytes geïnitialiseerd.

Structure – afdrukken

- Afdrukken van een structure:

```
printf("%08d ", student1.idcode);
printf("%-29s ", student1.achternaam);
printf("%-9s ", student1.klas);
printf("%4.1f\n", student1.resultaat);
```

- %08d → acht cijfers met leidende nullen
- %-29s → maximaal 29 karakters, links uitgelijnd
- %-9s → maximaal 9 karakters, links uitgelijnd
- %4.1f → 4 cijfers, met 1 decimaal en punt

Structure – inlezen

- Inlezen van een structure met `scanf`:

```
struct student_struct student3;

printf("Geef idcode, achternaam, klas en resultaat: ");

scanf("%d", &student3.idcode); /* gebruik & */
scanf("%29s", student3.achternaam); /* geen & */
scanf("%9s", student3.klas); /* geen & */
scanf("%lf", &student3.resultaat); /* gebruik & */
```

Structure – vergelijken

- Twee structures kunnen alleen vergeleken worden door alle members paarsgewijs te vergelijken:

```
if (s1.idcode == s2.idcode &&
    strcmp(s1.achternaam, s2.achternaam) == 0 &&
    strcmp(s1.klas, s2.klas) == 0 &&
    s1.resultaat == s2.resultaat) {
     /* true */
 }
```

- Dus niet: `if (s1 == s2) { } /* FOUT! */`

Structure – array

- Array van structures met initialisatie:

```
struct student_struct studenten[50] =
{
   { 19001234, "Alberts", "ep13", 7.3 },
   { 19001235, "Blokland", "ep14", 6.9 },
   { 19001236, "Cornelie", "ep15", 4.9 }
};
```

- De overige structures worden met 0 geladen. Het aantal (50) mag weggelaten worden voor exacte allocatie.

Structure – afdrukken array

- Afdrukken van de array:

```
for (int i = 0; i < 3; i++)
{
   printf("%08d %s\n", studenten[i].idcode,
                       studenten[i].achternaam);
}
```

- De overige members op vergelijkbare wijze.

Structure – grootte array bepalen

- De aantal elementen in de array kan bepaald worden met `sizeof`:

```
int nr_elem = sizeof studenten / sizeof studenten[0];

for (int i = 0; i < nr_elem; i++)
{
   printf("%08d %s\n", studenten[i].idcode,
                       studenten[i].achternaam);
}
```

Functies – argument/parameter

- Een structure mag in zijn geheel dienen als argument/parameter.

```
void print_student(struct student_struct stud)
{
    printf("%08d ", stud.idcode);
    printf("%-29s ", stud.achternaam);
    printf("%-9s ", stud.klas);
    printf("%4.1f\n", stud.resultaat);
}
…
print_student(student3);
```

Functies – teruggave

- Een structure mag in zijn geheel dienen als returnwaarde:

```
struct student_struct lees_student(void) {
    struct student_struct stud;

    scanf("%d", &stud.idcode);
    scanf("%29s", stud.achternaam);
    scanf("%9s", stud.klas);
    scanf("%lf", &stud.resultaat);

    return stud;
}
```

Functies – teruggave

- Een structure mag in zijn geheel dienen als returnwaarde:

```
struct student_struct lees_student(void) {
  /* zie vorige slide */
}
```

- Aanroepen met:

```
student4 = lees_student();
```

Functies – gebruik array

- Een array van structures mag ook dienen als argument/parameter, maar het aantal elementen moet meegegeven worden:

```
double maximum(struct student_struct studenten[], int aantal)
{
  /* definitie van de functie */
}
```

- Array, dus adres van het eerste element wordt meegegeven.

Functies – gebruik array

- Voorbeeld: vind hoogste resultaat:

```
double maximum(struct student_struct studs[], int aantal) {
double max = 0.0;

for (int i = 0; i < aantal; i++) {
    if (studs[i].resultaat > max) {
        max = studs[i].resultaat;
     }
}

return max;
}
```

Typedef

- Met `typedef` kan een nieuw datatype aangemaakt worden.
- Handig bij het gebruik van structures:

```
typedef struct student_struct student_t;
```

- Nu kan `student_t` gebruikt worden:

```
student_t student6, student7, heleklas[50];
```

Typedef

- Met `typedef` kan een nieuw datatype aangemaakt worden.
- Dit kan samen met de declarative van de structure:

```
typedef struct student_struct {
    int idcode;
    char achternaam[30];
    char klas[10];
    double resultaat;
} student_t;
```

Gebruik typedef

- Voorbeeld van gebruik `typedef`:

```
double gemiddelde(student_t studs[], int aantal)
{
   double som = 0.0;

   for (int i = 0; i < aantal; i++) {
      som = som + studs[i].resultaat;
   }

   return som / aantal;
}
```

Typedef – again

- Met typedef kan een nieuw datatype aangemaakt worden.
- Handig bij gebruik van exacte grootte van een variabele.

```
typedef signed char int8_t;
typedef signed long int int32_t;
typedef signed long long int int64_t;

typedef unsigned char uint8_t;
typedef unsigned long int uint32_t;
typedef unsigned long long int uint64_t;
```

Enum

- Met `enum` kan een enumeratie gemaakt worden. Dit wordt dan een nieuw datatype. Voorbeeld:

```
enum boolean { FALSE, TRUE };
```

- `FALSE` krijgt de waarde 0 en `TRUE` krijgt de waarde 1. Gebruik van het nieuwe datatype:

```
boolean x = TRUE;
…
if (x == TRUE) { … }
```

Enum

- Voorbeeld van enum met toegekende waarden:

```
enum filecode { MAP=1, REGULAR=2, PIPE=4, SPECIAL=8 };
```

- `MAP` krijgt de waarde 1, `REGULAR` krijgt de waarde 2 etc. Gebruik van het nieuwe datatype:

```
filecode file = REGULAR;
…
if (file == MAP) { … }
```

Opdracht

Schrijf een functie met de volgende declaratie:

```
void verhoog(student_t studenten[],
             int aantal, double hoeveelheid);
```

die het resultaat van alle (via `aantal`) studenten verhoogt met `hoeveelheid`. Let erop dat een resultaat nooit groter kan zijn dan 10,0 en nooit lager kan zijn dan 1,0.

Uitwerking

```
void verhoog(student_t studenten[], int aantal, double hoeveelheid)
{
    for (int i = 0; i < aantal; i++)
    {
        studenten[i].resultaat += hoeveelheid;
        if (studenten[i].resultaat > 10.0)
        {
            studenten[i].resultaat = 10.0;
        } else if (studenten[i].resultaat < 1.0)
        {
            studenten[i].resultaat = 1.0;
        }
    }
}
```

let’s change