기초컴퓨터프로그래밍 10분반 25\_이규하 (2021039057)

15주차 FINAL Report

* 함수

- 함수의 선언 : 사용자 정의 함수를 사용하겠다는 것을 컴파일러에게 미리 알려주는 역할

- 함수의 정의 : 함수가 처리해야할 기능들을 구체적으로 작성 (함수의 결과 값(반환 값)이 있는 경우에는 return문과 함께 변수 또는 연산 식 사용.

|  |  |
| --- | --- |
| 함수 선언 | 데이터 형 함수이름 (함수인자들); |
| 함수 정의 | 데이터 형 함수이름 (함수인자들);  {  변수선언;  문장;  [return 변수 또는 수식;]  } |

|  |  |  |
| --- | --- | --- |
| - 결과 값(반환 값)이 없는 void 형 함수 | | #include <stdio.h>  void line\_print(void);  int main(void)  {  line\_print();  return 0;  }  void line\_print(void)  {  함수 세부사항  } |
|  |  | |
| - 결과 값(반환 값)이 있는 함수 | | #include <stdio.h>  int add(int x);  int main(void)  {  int n, sum;  printf(“1부터 n까지의 합\n:”);  printf(“정수 n 입력 : ”);  scanf(“%d”, &n);  sum=add(n); // 함수 호출  printf(“정수합 : %d\n”, sum);  return 0;  }  int add(int x)  {  int i, result=0;  for(i=1; i<=x; i+=1)  result+=i;  return result;  } |

- 매크로 상수와 함수 (ex. #define START 50)

: 프로그램 작성단계에서 결정할 수 없는 상수를 정의하는 경우에 사용

: 매크로의 이름은 일반적인 변수와 구별하기 위해 대문자 사용

: 매크로 상수로 정의된 값은 변수로 사용할 수 없으며 프로그램 내에서 저장된 값의 변경이 불가.

- 재귀 함수

: 호출된 함수가 실행 중에 자기 자신을 다시 호출하는 것.

: 재귀 호출에서는 무한 반복이 이루어지므로 이를 해결하기 위해 적절한 조건을 주어 함수를 벗어나도록 해 주어야 함.

- 변수

|  |  |  |
| --- | --- | --- |
| **변수 종류** | **의미** | **사용방법** |
| 지역변수 | 그 변수의 사용 범위가 어떤 한 함수 안에서만 사용 | 함수 또는 복합문 안에서 변수를 선언 |
| 전역변수 | 범위의 제한 없이 프로그램의 모든 함수에서 접근가능 | 함수 밖에서 변수를 선언 |

- 자동(auto)변수

: 함수 안에서 선언하는 지역 변수들은 모두 자동(auto) 변수에 해당.

: 함수가 호출되어 실행될 때만 기억 공간이 확보(유지)되고 함수의 실행이 종료됨과

동시에 자동으로 소멸되는 변수

: 자동 변수를 선언하기 위해서 auto라는 기억 클래스 지정자를 사용하지만 생략가능.

- 정적(static)변수

: static 지정자를 사용

: 프로그램이 종료될 때까지 변수의 수명이 지속

: 초기화를 하지 않아도 0으로 초기화(한번 초기화된 정적 변수는 더 이상 초기화X)

: 함수 내부에서 선언될 경우=내부 정적 변수

: 함수 외부에서 선언될 경우=외부 정적 변수

* 포인터 : 포인터(pointer)는 메모리에 저장된 데이터의 위치를 가리킴.
* 포인터 변수

: 오직 주소로만을 저장하고 사용하기 전에 선언해야 하며, 저장된 값을 다른 주소로

변경시킬 수 있음.

: 포인터 변수를 선언할 경우 간접(indirection)연산자 \*를 사용.

|  |
| --- |
| 데이터 형 \*포인터 변수 |

(데이터 형은 포인터 변수가 가리키게 될 주소에 저장된 데이터 형을 가리킴)

* 포인터와 배열

: 배열 요소들은 메모리 상에 연속된 기억 공간에 저장됨.

: 배열의 이름은 그 배열의 첫 번째 요소가 저장되어 있는 주소를 가리키므로 주소를

가리킨다는 점에서는 배열과 포인터는 유사.

- 포인터와 문자열

배열 또는 포인터를 이용하여 문자열을 초기화

|  |  |  |
| --- | --- | --- |
| 방법1 | char str1[]=“copy”; | |
| char str1[9]=“copy”; //방법1의 또다른 쓰임 | |
|  | |  |
| 방법2 | char str2=“copy”; | |

- 포인터에 대한 연산

: 포인터 변수에는 주소가 저장되고 주소는 4byte(Visual C++)로 표현된 숫자이므로

포인터 변수나 주소에 대한 연산이 가능

: 주소는 양의 정수로 표현하므로 실수 연산은 불가

- 포인터를 인자로 사용하는 함수 호출

|  |  |
| --- | --- |
| 방법1 | 값의 의한 호출(call by value)  : 인자를 전달할 때 항상 그 값(value)만을 전달해 줌. |
| 방법2 | 주소에 의한 호출(call by address)  : 포인터(주소)를 함수의 인자로 사용하여 호출. |

- 포인터 배열

: 포인터 배열이란 여러 개의 포인터 변수를 배열로 사용.

: 문자형 포인터 배열을 사용할 때 유용

|  |
| --- |
| char na[4][17]={“Korea”, “Brazil”, “Germany”, “Papua New Guinea”};  char \*ct[]={“Korea”, “Brazil”, “Germany”, “Papua New Guinea”}; |

- 포인터와 함수

: 함수의 주소를 가리킴

: C언어에서 포인터 상수(문자열 상수, 배열의 이름, 함수의 이름)

: 함수의 이름은 그 함수가 시작되는 주소를 나타내는 포인터 상수이므로 함수

포인터를 사용하여 함수를 인자로 전달 가능

|  |
| --- |
| 데이터 형 \*함수포인터(함수 인자들) |

* + 구조체 : 서로 다른 데이터 형을 갖는 변수들을 대표 이름으로 묶어 놓는 것.

|  |
| --- |
| struct 구조체 태그  {  데이터 형 변수;  데이터 형 변수;  ...  } |

구조체 태그(구조체를 대표하는 이름)

멤버(구조체를 구성하는 항목들)

구조체 멤버의 이름은 구조체 태그와 다른 이름을 사용.

- 구조체 변수와 데이터 입력

|  |  |  |
| --- | --- | --- |
| **[방법 1]**  **구조체를 정의한 후에**  **구조체 변수를 선언** | **[방법 2]**  **구조체 정의와 동시에**  **구조체 변수를 선언** | **[방법 3]**  **typedef를 이용하여**  **구조체 변수를 선언** |
| struct user  {  char name[20];  int year;  };  struct user d; 또는  user d; | struct user  {  char name[20];  int year;  } d; | typedef struct {  char name[20];  int year;  } user;  user d; |

- 구조체 배열

: 구조체 변수와 다르게 첨자가 사용

: 저장된 내용을 출력하는 경우 도트 연산자 . 를 이용

- 구조체 포인터

: 구조체 포인터에 대해서는 간접 연산자 ‘->’을 사용.

(ex. (pt+i)->name은 pt[i].name과 동일)

- 구조체와 함수

: 함수의 인자로 구조체 변수를 사용하는 것은 일반 변수를 함수의 인자로 사용하는

것과 차이가 없음

- 구조체를 헤더파일로 저장하여 사용하는 방법

: 메모장(notepad)프로그램을 이용하여 구조체 정의 부분만 따로 **헤더파일명.h**라는

이름의 텍스트 파일로 저장

: #include **헤더파일명.h** 이용하여 구조체를 불러드릴 수 있음.

- 구조체 속의 구조체(중첩된 구조체)

: 구조체는 모든 데이터 형을 포함할 수 있기 때문에 구조체의 멤버로 또 다른 구제를

사용 할 수 있음.